COMPUTER-AIDED DESIGN,
ENGINEERING, AND MANUFACTURING
SYSTEMS TECHNIQUES AND APPLICATIONS

VOLUME
I

SYSTEMS TECHNIQUES AND COMPUTATIONAL METHODS

COMPUTER-AIDED DESIGN,
ENGINEERING, AND MANUFACTURING
SYSTEMS TECHNIQUES AND APPLICATIONS

VOLUME
I

SYSTEMS TECHNIQUES AND COMPUTATIONAL METHODS

EDITOR
CORNELIUS LEONDES

CRC Press
Boca Raton London New York Washington, D.C.

Library of Congress Cataloging-in-Publication Data

Catalog record is available from the Library of Congress.

Direct all inquiries to CRC Press LLC, 2000 N.W. Corporate Blvd., Boca Raton, Florida 33431.

International Standard Book Number 0-8493-0993-X
Printed in the United States of America 1 2 3 4 5 6 7 8 9 0
Printed on acid-free paper

Preface

A strong trend today is toward the fullest feasible integration of all elements of manufacturing, including maintenance, reliability, supportability, the competitive environment, and other areas. This trend toward total integration is called concurrent engineering. Because of the central role information processing technology plays in this, the computer has also been identified and treated as a central and most essential issue. These are the issues that are at the core of the contents of this volume.

This set of volumes consists of seven distinctly titled and well-integrated volumes on the broadly significant subject of computer-aided design, engineering, and manufacturing: techniques and applications. It is appropriate to mention that each of the seven volumes can be utilized individually. In any event, the great breadth of the field certainly suggests the requirement for seven distinctly titled and well-integrated volumes for an adequately comprehensive treatment. The seven volume titles are:

1. Systems Techniques and Computational Methods
2. Computer-Integrated Manufacturing
3. Operational Methods in Computer-Aided Design
4. Optimization Methods for Manufacturing
5. The Design of Manufacturing Systems
6. Manufacturing Systems Processes
7. Artificial Intelligence and Robotics in Manufacturing

The contributors to this volume clearly reveal the effectiveness and great significance of the techniques available and, with further development, the essential role that they will play in the future. I hope that practitioners, research workers, students, computer scientists, and others on the international scene will find this set of volumes to be a unique and significant reference source for years to come.

Cornelius T. Leondes
Editor

Editor

Cornelius T. Leondes, B.S., M.S., Ph.D., is an Emeritus Professor at the School of Engineering and Applied Science, University of California, Los Angeles. Dr. Leondes has served as a member or consultant on numerous national technical and scientific advisory boards. He has served as a consultant for numerous Fortune 500 companies and international corporations, published over 200 technical journal articles, and edited and/or co-authored over 120 books. Dr. Leondes is a Guggenheim Fellow, Fulbright Research Scholar, and Fellow of IEEE. He is a recipient of the IEEE Baker Prize, as well as its Barry Carlton Award.

Contributors

Sanjeev Bedi
University of Waterloo
Waterloo, Ontario, Canada

M. Cargian
Syracuse University
Syracuse, New York

Francesco Crenna
University of Genova
Genova, Italy

Mohamed Dekhil
Rain Infinity
San Jose, California

Ruxu Du
University of Miami
Coral Gables, Florida

Simon Dunstall
University of Melbourne
Victoria, Australia

Paul Hoskins
CAMplete Solutions, Inc.
Waterloo, Ontario, Canada

Fathy Ismail
University of Waterloo
Waterloo, Ontario, Canada

Kug Weon Kim
Chonan National Technical College
Chungnam, South Korea

Rinaldo C. Michelini
University of Genova
Genova, Italy

Jonathan Owen
University of Bridgeport
Bridgeport, Connecticut

Henk Jan Pels
Eindhoven University of Technology
Eindhoven, The Netherlands

G.B. Rossi
University of Genova
Genova, Italy

Utpal Roy
Syracuse University
Syracuse, New York

Hyo-Chol Sin
Seoul National University
Seoul, Korea

Tarek Sobh
University of Bridgeport
Bridgeport, Connecticut

Andrew Warkentin
University of Dalhousie
Halifax, Nova Scotia, Canada

Andrew Wirth
University of Melbourne
Victoria, Australia

Arian Zwegers
Baan Development
Apeldoorn, The Netherlands

Contents

Chapter 1 Tele-Manufacturing: Rapid Prototyping on the Internet/Intranets
Utpal Roy and M. Cargian

Chapter 2 Models and Algorithms for Machine Scheduling with Setup Times
Simon Dunstall and Andrew Wirth

Chapter 3 Computer Aided 5-Axis Machining
Andrew Warkentin, Paul Hoskins, Fathy Ismail, and Sanjeev Bedi

Chapter 4 Finite Element Method and Thermo-Viscoplastic Cutting Model in Manufacturing Systems
Kug Weon Kim and Hyo-Chol Sin

Chapter 5 Diagnostics for Monitoring Maintenance and Quality Manufacturing
Rinaldo C. Michelini, Francesco Crenna, and G.B. Rossi

Chapter 6 Reverse Engineering and Inspection of Machined Parts in Manufacturing Systems
Tarek Sobh, Jonathan Owen, and Mohamed Dekhil

Chapter 7 Application of Reference Architectures for Enterprise Integration
Arian Zwegers and Henk Jan Pels

Chapter 8 Engineering Monitoring and Diagnosis Using Wavelet Transforms
Ruxu Du

1

Tele-Manufacturing: Techniques and Applications for Rapid Prototyping on the Internet/Intranets

1.1 Introduction
Rapid Prototyping Background • Current Access to Prototyping • Slicing a Model for Rapid Prototyping • Objective and Motivation

1.2 Preliminaries
Rapid Prototyping Representation • Current Slicing Algorithms • Java • Collaborative Design

1.3 Review of Related Works
Slicing Algorithms • Cusp Height and Accuracy • Decreasing Processing and Build Time • Observations from the Reviewed Research

1.4 Computational Aspects and Procedures
Preparing the STL File for Slicing • Searching for the Start of a Contour (Trigger Point) • Determining the Trigger Point Edge • Tracing Out the Contour

1.5 Prototype Implementation and Case Studies
System Architecture of the Prototype • Program Initialization • Data Input and Manipulation • Search Method to Find Trigger Points • Contour Tracing Implementation • World Wide Web Access • Results of Enhanced Algorithm

1.6 Conclusions

Utpal Roy
Syracuse University

M. Cargian
Syracuse University

The fundamental aspects of rapid prototyping systems have steadily matured since their inception in the 1980s. Intricacies of CAD model slicing for layered manufacturing are compounded by construction of a 3-D model from 2½-D slices. Surface finish problems, processing time, and accuracy are within the scope of this work, which entails the development of a slicing algorithm that works through the World Wide Web. Additionally, this process extends the current slicing algorithms to one that is more apt to deal with file processing time and connections for collaborative design. The use of the World Wide Web will enable this software to be utilized by others without the need for software at their sites. The software is developed with tools such as Java, HTML, and the Web.

1.1 Introduction

Manufacturing in today's world has developed and changed due to new technology and worldwide competition. As a consequence, a need for quicker and faster development times for products has arisen. Other design and manufacturing technologies such as concurrent engineering, design for manufacture, just-in-time production, and computer-aided design (CAD) have pushed the design envelope further. To reduce the development times, companies need to be able to create prototypes of their new products—quickly and cost effectively.

Rapid Prototyping Background

In today's competitive business arena, companies must continually strive to create new and better products, faster and more efficiently than their competitors. The design and manufacture process is continually enhanced to be more responsive to changes, as well as quicker to market. Many technologies and business practices, such as concurrent engineering, just-in-time production, and design for manufacture, have been utilized to decrease design time. In addition to these enhancements, over the past decade, rapid prototyping has evolved to improve the product design cycle. Rapid prototyping is a system for creating immediate prototypes of a new design or change that is used to evaluate it or in actual application.

There are many CAD environments available for creating new engineering designs or concepts. As a new design or modification to a current design is developed in a package such as CATIA or Pro/Engineer, this model can then be created in a short time to have an actual prototype for further testing. This prototype, along with analysis tools, helps to quickly define the success and failures of the new design.

Previously, prototypes could be costly and take a long time to create. Once the prototype is complete, further modifications may be needed, and again the cost and time increase. With rapid prototyping, there are costs and time associated with making the object, but at much lower expense to the designer. Where prototypes may have previously been measured in days and weeks, they now can be measured in hours.

Rapid prototyping is also known as layered manufacturing (LM) as well as several other names. The LM process takes the CAD model of an object and virtually slices the object into many two-dimensional (2-D) patterns of the cross section of the object. This stack of slices is then created by an LM machine, one layer at a time. Each layer adheres to the layer below it, eventually creating the final prototype.

Current Access to Prototyping

After creating an object, a designer who wishes to have a prototype made has a few options. The file can be sent to an outside prototyping agency for price quoting and possible creation. Part of speed of *rapid* prototyping is lost due to the time spent communicating with the vendor, sending the object, and having it processed. Alternatively, the company can choose to invest in its own prototyping hardware, software, and people to staff it, which can be expensive. Additionally, there are several other factors to consider in the prototyping equipment, all of which have unique features and attributes that result in different prototype quality and finish.

Slicing a Model for Rapid Prototyping

A great deal of work and research has been done in the area of slicing the CAD model for rapid prototyping. These slicing algorithms take a CAD model and, based on a user-defined layer height, slice the model. Prototypes created in a layer-by-layer process exhibit a staircase effect on some surfaces. As shown in Fig. 1.1, the difference at the edge of the CAD model and the sliced model is termed the cusp

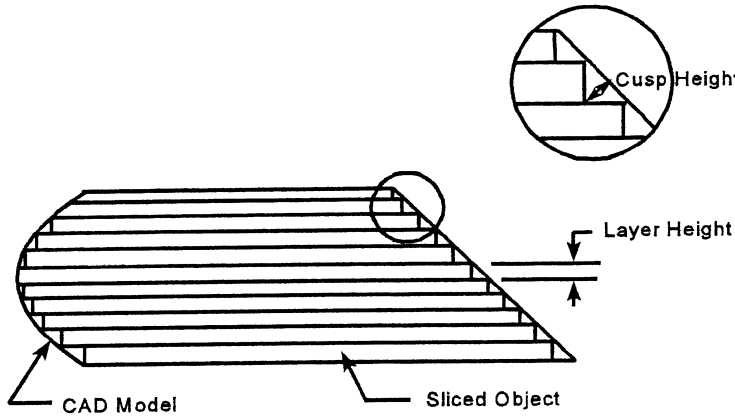


FIGURE 1.1 Comparison of the CAD model to the sliced prototype.

height. Ideally, the cusp height should be a minimum for any object. Although the slice height can be changed by the user of most software packages, and will reduce the cusp height, a decreased fixed slice height results in longer computing and processing time on the machine. In general, it would be advantageous for the slicing software to minimize the cusp height while performing the slicing procedure as quickly as possible.

Objective and Motivation

With this brief introduction to rapid prototyping, it can be seen that there are several areas that can be improved. Two of these improvements are in the accessibility of rapid prototyping software and the slicing algorithm.

Currently, any institution wishing to implement rapid prototyping on a project must invest a large amount of capital in a free-form fabrication apparatus, software, and training, or have their model created by an outside rapid prototyping agency.

This slicing module is aimed at being one of the design services for the collaborative product design environment (CPD) developed in the Knowledge Based Engineering Laboratory at Syracuse University. This CPD framework allows product design across the Internet by virtual collocation of people and services. Once the design work is complete within this environment, there are several Web-based services for testing and analysis of the model. Some of the services available include process planning for the part, casting analysis, and NC path planning.

The objective of this work is to develop a module that will allow slicing for rapid prototyping from this environment. Internet prototyping could allow a required model to be sent instantly over the Internet and added to the queue of the layered manufacturing machine. The reduction in processing time of the sliced object needs to be addressed by utilizing a modified slicing algorithm. These objectives are to be met using a number of tools including, Java and the Internet.

1.2 Preliminaries

The essence for rapid prototyping is the need to obtain fast, cost-effective parts to aid in the design of a new product. There are several criteria that describe the layer-by-layer manufacturing process, including file format, slicing algorithms, and manufacturing systems. File format defines the current standards for object and feature information transfer. The prototype finish is directly affected by the slicing algorithm and manufacturing process used.

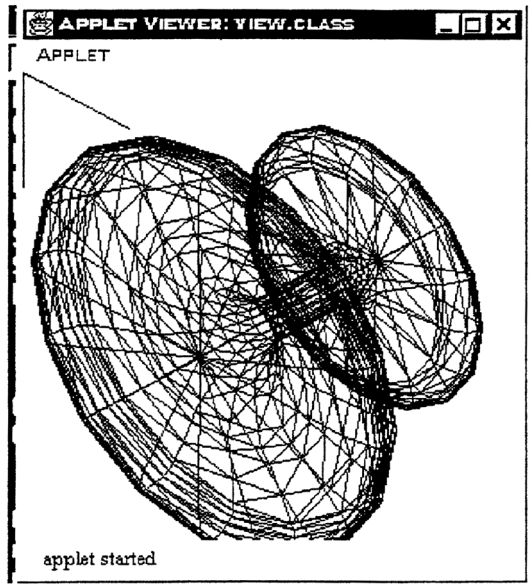


FIGURE 1.2 Display of the STL file of a faceted object.

Rapid Prototyping Representation

The current exchange standard for rapid prototyping is the stereolithography (STL) file. The STL file is a triangulation of the surface of the object that creates a set of triangles (facets) that represent the surface of the original model. The triangulation results in a set of nonoverlapping triangles that do not extend outside of the object's boundaries but completely cover the entire surface of the CAD object. A sample tesselated object is shown in Fig. 1.2.

The tessellation results in an unordered, feature- and topology-deficient file. This file does not have any information on features such as holes, cuts, or protrusions. Additionally, there is no surface or contour data about the object. The STL file does not contain any information to drive the layered manufacturing process, leaving the slicing algorithm to find all of the contours. Although the STL file does not help the slicing process with information, it is the industry standard that most CAD packages can create and is the neutral format utilized for this Web project.

Current Slicing Algorithms

Many current slicing systems utilize geometric intersections of a plane (horizontal to the stereolithography platform) with the object to define each slice. Each new slice that needs to be defined is created by moving the intersecting plane further up the object as in Fig. 1.3. Once the line segments are found from these intersections, they must be sorted in a head-to-tail fashion, as in Fig. 1.4, to form the contour.

Java

A new tool for the Web is the Java programming language developed by Sun Microsystems, Inc., in 1991 (Cornell and Horstmann, 1996). It is becoming the next language of the Internet, and changes the way the World Wide Web operates. With Java, Web pages can come alive with graphics, animations, video, and sound. Java is a full-featured, object-oriented programming language that is adaptable to many situations.

Within Java, the programmer can create applets (programs that run over the Internet or intranets) or applications. Applets are stored on an Internet/intranet server, and users download them when they

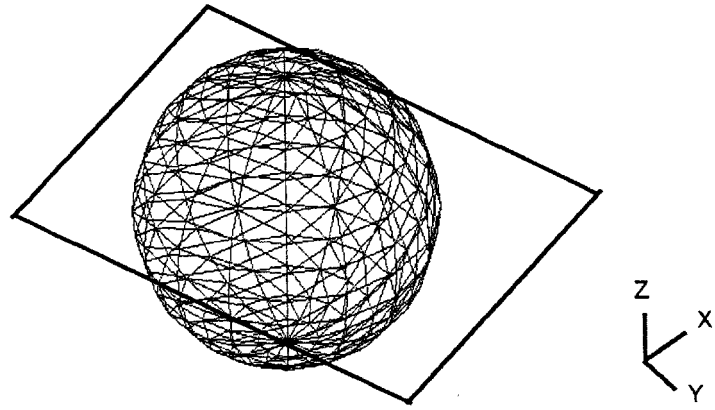


FIGURE 1.3 Previous slicing algorithms that intersect a plane with the STL object to create each contour.

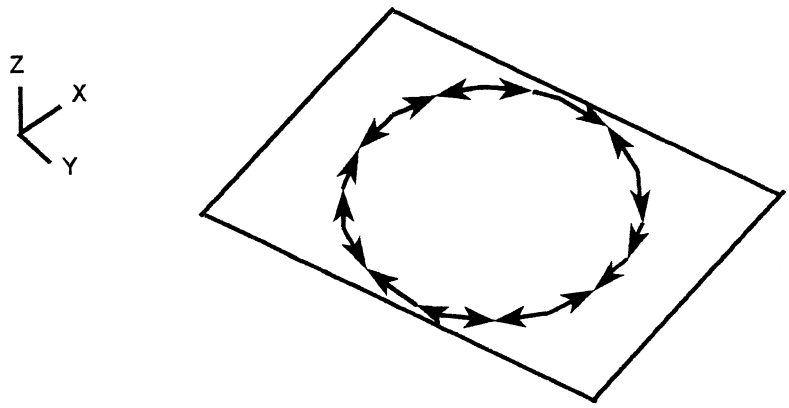


FIGURE 1.4 Resulting unsorted, unordered contour that needs to be arranged to form a complete contour.

arrive on a Web page which then runs locally on the user's machine. An interpreter usually found in the Web browser is required to read and process the neutral Java file. Applications, on the other hand, are stand-alone programs that do not run inside a browser.

Java programs are stored as bytecodes, intermediary files that allow the Java program to run on any computer that has an interpreter. There is no longer a barrier caused by differences among operating systems run on a computer. The code for a Java program can be written once, stored on a Web server, and then downloaded by the user onto any machine. Because popular browsers now support Java, and many operating systems are adding Java as a central feature, it is easy to run Java applets or applications.

One limitation of the current system is a question of security. Most browsers disable Java's access to read or write to the local user's machine. The security feature in the browser limits the Java program to only access files on the server from which it came. By doing so, a malicious program is prevented from erasing or damaging the user's computer or data. If access is required to read and write on the local computer, Java can develop stand-alone applications. These applications are full-blown applications such as those created in C++ and can be run directly from the command line.

The possibilities for Java-enabled Web pages are endless. Utilizing this programming language opens the possibility to the "diskless" computer, where all applications are stored on Internet/intranet servers.

As programs are needed for specific tasks, they would be downloaded from a Web page. Whether or not Internet-centric computing becomes a reality, Java will have an important role in creating dynamic, useful Web pages.

Collaborative Design

The Internet and corporate Intranets have become a powerful tool for communication. Engineers and designers now have the opportunity to work on projects with people all over the world through the Web. Research in this area is being done at Syracuse University to make the collaborative design process possible for product design. Systems are being implemented to connect feature and part information with a sharable object using tools like VRML. This collaborative product design framework allows designers from all over the world to work together on a design.

Once the design takes place in this networked enterprise, an Internet/intranet-based rapid prototyping module could then be used to process the CAD model to create the object. This program could be linked directly to a layered manufacturing machine to create the prototype immediately.

1.3 Review of Related Works

To make the rapid prototyping process more accurate and create better parts, a number of possible enhancements have been studied. Research suggests several main areas that help to improve the rapid prototyping process, including slicing algorithms, cusp height, part accuracy, and decreasing both processing and build time. Each of these areas relies heavily on the others because a solution in one area may help or hinder another.

Slicing Algorithms

A great deal of prototyping time is involved in the slicing of the CAD file. The slicing process has a direct effect on the length of time to build the prototype, as well as on the quality of the finished product. Any modifications to the slicing process will help with the speed of the program.

The current algorithms that create slices by intersecting planes with objects can be enhanced. Each facet that is intersected with a slicing plane, forms a line segment of the contour. Each of these unorganized line segments is then ordered from head to tail to create the closed contour. One method (Chalasani et al., 1991) offers an alternative to plane intersection slicing using the scan-line type approach to find the contours of an object. By utilizing scan-line type slicing, the contour segments are found in a closed loop. The search starts at an origin point and searches until it "intersects" the object at either an edge or a vertex. Then, the tracing procedures are executed to generate the contour. This search continues, finding all of the contours that are present on that slice of the object. Although scan-line type slicing is more complex, it yields considerable savings. Two reasons explain this: there is no need to store the contour segments and then sort them at the end, and it reduces the need for difficult geometric calculations to find the intersections with the facets. After the contours have been traced, a simple hatching routine is formed to drive the laser in the build process.

At the time this chapter was written, the scan-line type approach had not yet been implemented. This process requires an exhaustive search of the entire object to find all of the contours on any given layer. This searching is a computationally expensive procedure.

One enhancement to this method utilizes topological information to increase slicing speed (Rock and Wozny, 1991). The first information generated by this algorithm is connectivity information about the facets. Topology building compensates for some of the deficiencies in the STL file by linking the three edges of each facet with the three neighboring facets. By searching through the STL file, the connections between facets and their edges can be saved for future use. A sort is used to find the neighboring facet for each edge. The connectivity information is used during contour tracing to move from one facet to the next. The facets are then spatially partitioned into areas called bins. These bins store the numbers of the facets that pass through them. A facet may belong to one bin, several, or all of them, depending on the facet's orientation and size. Figure 1.5 shows several facets of an object in relation to the different bins. Facet 1

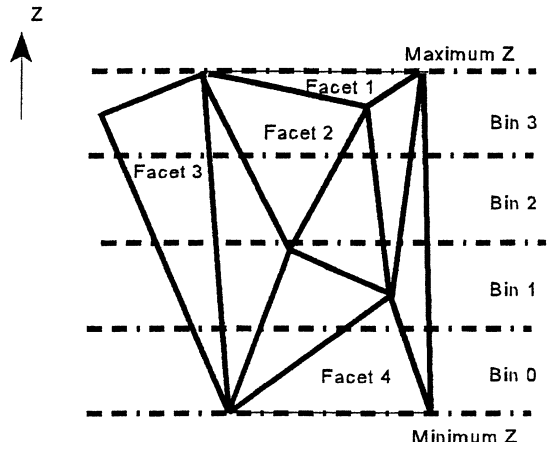


FIGURE 1.5 The four bins of the object, created in the *z*-direction.

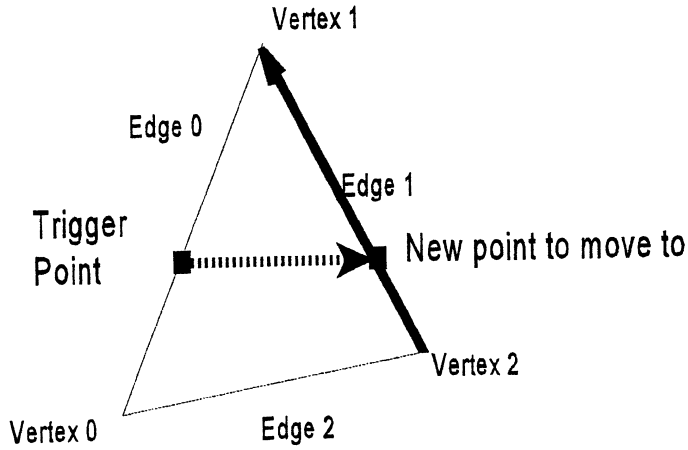


FIGURE 1.6 Generating a line segment across a facet.

in this figure is completely contained within bin 3, whereas facets 2 and 4 are contained within several bins. Facet 3 lies in all four bins and would be recorded as such in the bin listing.

These two steps of the algorithm are important for reducing slicing time. The next step is to find an intersection of one of the facets with the slicing plane. Once this is found, the line segments of the contour are generated by intersecting the slicing plane with the next facets. This intersection determination is simplified by realizing that the normal to the slicing plane does not change, and the new point can be found through interpolation. The new point is given by:

$$New\ point\ =\ Vertex_2 + t\,(Vertex_1 - Vertex_2)$$

The new point is shown in Fig. 1.6. As each edge is used in the tracing procedure, a flag is set so that the edge is not used again later. However, this method does not help to determine the starting point of the contour, as it still needs to be determined by intersecting two planes.

Other research (Dolenc and Mäkelä, 1994) determines proper slicing procedures for handling flat areas. During slicing, it is possible for the slice height to miss the beginning of a flat feature (flat meaning parallel to the slicing plane). Also, the abrupt change of the slope of the object must be noted to minimize

the cusp height. The algorithm presented varied the slice height based on these problems of flat surfaces, changing slopes, and cusp height.

Cusp Height and Accuracy

One methodology (Kulkarni and Dutta, 1995) varies the slicing thickness based on the geometric curvature of the surface at that slice. In areas where there is a large curvature, the slicing thickness is reduced; while in non-curved areas, the slice can be thicker to allow faster processing time. The amount of curvature of a surface is measured based on the changes in the normal to a surface in the area of concern. By computing the local curvature and comparing to the required cusp height for the model, the authors adjust the slicing height as necessary.

To demonstrate their work, they use an analytical ellipsoid as a case study. First, the equation of the ellipsoid is parameterized and an equation for the normal curvature is defined. By maximizing the normal curvature with a constraint that the slice thickness must remain the same, the maximum curvature and layer thickness can be found.

Typical solutions previously had been to check the cusp height after slicing and to reslice at an intermediate level if the cusp height was too great. By utilizing the new method, greater flexibility and control is allowed over the cusp height. The case study presented shows the limitations of this method as it is a precise analytical model of an ellipsoid which has a known equation for its representation. The equation can then be used to test the curvature and set the cusp height accordingly. However, the idea needs to be extended to other objects and ultimately to any shape at all.

Another possible modification to help part accuracy is to look at the file format in which the object is stored. When an object is tesselated, the accuracy is only as good as the approximation made by the STL file. The density of the tessellation can be adjusted to give better part accuracy, but at the expense of file size and slice time. Another method (Vuyyuru et al., 1994) gives an alternative file format by slicing the CAD object using non-uniform rational B-spline (NURBS) curves.

Utilizing previous research (Rajagopalan, 1992) in which a SDRC I-DEAS solid model is sliced into NURBS for each layer, the curve information is stored in a universal file for later processing. Because the stereolithography apparatus requires two-dimensional vectors to move the laser, the NURBS curves must be changed to short line segments. Using two different methods, the authors create these line segments and create the SLI file from it, which drives the laser.

The authors have managed to decrease the file size and increase the finish quality of the part using this method. It is obvious that the rapid prototyping world needs to move to more accurate parts in less time. Currently, there is no standard that allows NURBS slices to be created on all platforms with all CAD packages. The STL file still prevails due to its ease of implementation and acceptance by CAD vendors. In the World Wide Web project presented in this work, the STL file is used because of its commonality between platforms.

Other accuracy problems exist in the tolerances that are held within the faceted model and the laser beam width (Kirschman, 1992). Ideally, the width would be zero, as each contour would then be traced out exactly. However, this is not the case. Due to the large creation table area relative to the laser, error is induced as the beam spreads when it is at an angle. Although suggestions are proposed to correct some of these problems, the other errors associated with layered manufacturing need to be fixed first.

Decreasing Processing and Build Time

As the word *rapid* implies, other methods must be looked at to make the entire process faster. On small parts, the process is governed by the "dip and dunk" time of the stereolithography apparatus. The ability to create several parts on the same platform would aid in creating more parts in less time.

One process (Wodziak et al., 1994) uses a genetic algorithm to optimize the automatic placement of multiple objects on a platform at one time. The process looks at not only the location on the table, but also whether it should be rotated first. Other work (Frank and Fadel) has been done on part orientation

to aid in cusp height minimization and surface finish. An expert system is used and works with two of the features of the object to determine its orientation.

Another way to decrease processing time is to slice the file using a multi-processor system. Programs (Kirschman and Jara-Altamonte, 1992) that use the parallel processors are able to decrease the processing time of the STL file significantly. Other file formats and computer processes will evolve to meet the needs of rapid prototyping as it grows. A neutral file format needs to be extended to all CAD vendors to be able to slice any part with the required accuracy and finish.

Other researchers are looking into the Internet/intranets to provide additional functionality for the rapid prototyping process. Tele-manufacturing (Bailey, 1995) is one term that has been created to represent the use of rapid prototyping over the Internet. The main area that is presented is the repair of an incorrect STL file. Due to the lack of topology or order in the facets, the file often needs to be repaired to handle cracks, overlapping triangles, and zero-thickness surfaces. Although the STL file is passed across the Internet to the site to be processed, the slicing has not been implemented directly on the Web.

Observations from the Reviewed Research

The main contribution from the research to be implemented in this work is the use of the scan-line type approach for generating contours (Chalasani et al., 1991) along with topological information (Rock and Wozny, 1991). This will be used, in an enhanced form without continuously searching through the entire object, to quickly trace out each contour. In addition to this, the tele-manufacturing (Bailey, 1995) process will be extended to a complete Internet slicing program. Although there are suggestions for better file formats for the unsliced object, the STL file is used to do its neutral file format supported by most CAD packages.

1.4 Computational Aspects and Procedures

This section presents an alternative method for slicing a stereolithography (STL) file. It uses a modified method of the trigger point, contour tracing method with topological information. The basic idea behind this process is to search throughout the object, trying to find an edge or vertex of any facet on the slicing level, and then trace out the entire contour from there. The algorithm uses vectors between vertices of the facets to create each line segment of the contour. Generating slices by marching around the object eliminates finding unsorted line segments of the contour, and reduces processing time by not calculating complex geometric intersections of planes. An overview of the implementation is shown in Fig. 1.7.

Preparing the STL File for Slicing

All slicing procedures should be preceded by a check for completeness and accuracy of the STL file. Due to the topology-deficient format that the object is stored in, it is important to check the file to make sure it can be sliced. The STL file may not have been tessellated properly, resulting in gaps, improper triangulation, or intersecting facet edges.

Several criteria have been evaluated for the completeness and accuracy check but most importantly, each face must have three edges, each edge must have two vertices, each vertex must be part of exactly two of the face's edges, and each edge must be part of an even number of faces. If problems exist in the STL file, they must be corrected before slicing. Algorithms have been developed to fill in the missing data from these files (Ngoi et al., 1993).

Many layered manufacturing processes require support structures for the base of the object or any overhangs. These structures should be added to the STL file by a preprocessor before slicing in a method similar to the one created by Kirschman et al. (1991). Preprocessing adds facets to the STL file, and results in additional material being added to the object. These structures support other parts of the object so

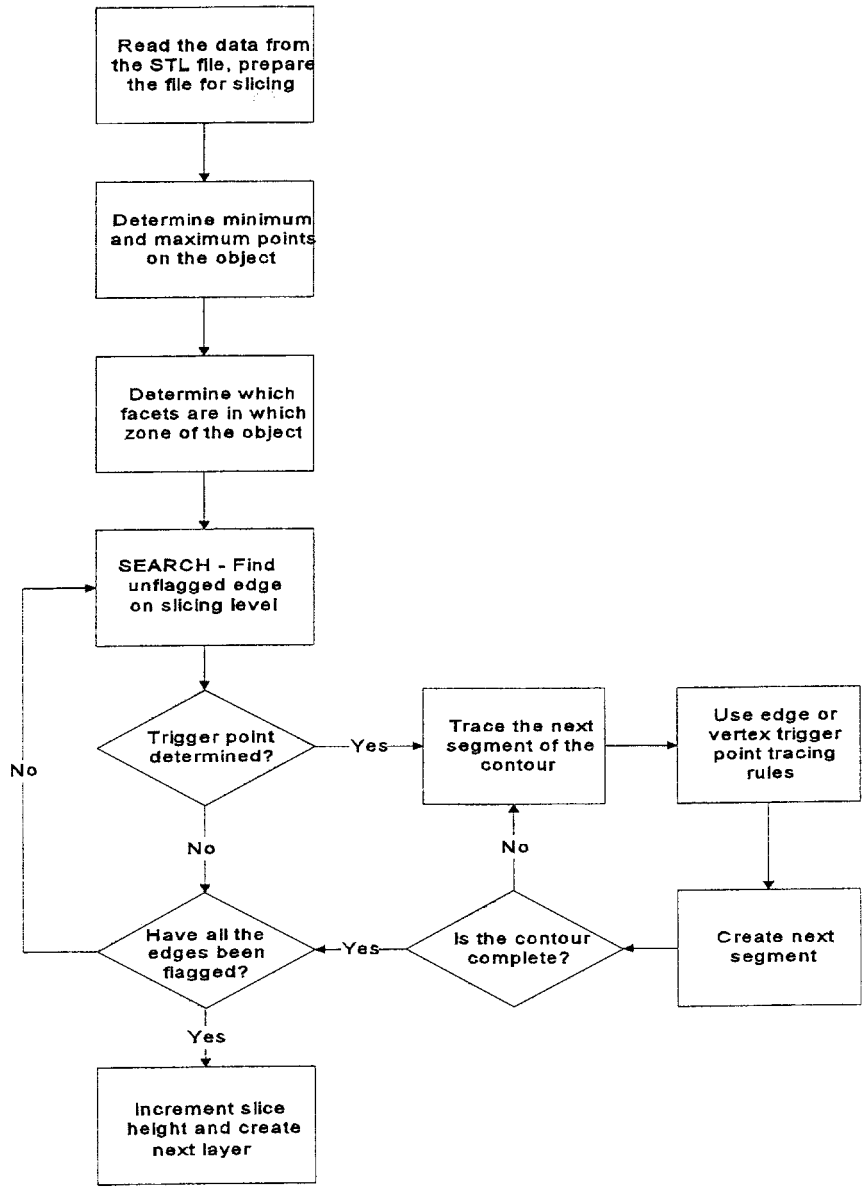


FIGURE 1.7 Flow of slicing system architecture.

that overhangs or other unsupported features do not break off during the build process. Once manufacture is complete, these areas are removed, leaving the desired prototype.

Searching for the Start of a Contour (Trigger Point)

To minimize the search time, four zones (spatially partitioned bins) are created in the *z*-direction, which split the object based on the maximum, minimum, and quarterly points. These zones store the numbers of the facets that pass through them. A facet may belong to one zone, several, or all of them, depending on the facet's orientation and size.

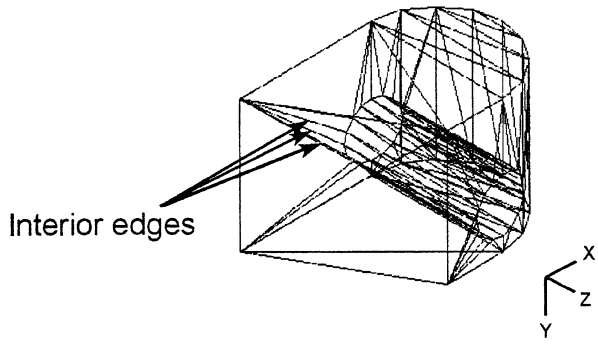


FIGURE 1.8 Interior edges of the object that will not be used for contour tracing.

Without contour or feature information on the object, a search of edges on the slicing level is required to find all possible contours. During the searching and tracing procedures, it is important that the line segments are never created inside the contour. All contours for prototyping are on the "outside" of an object of feature. Interior edges that will not be used for contour tracing are shown in Fig. 1.8.

These edges are never used in the contour tracing because they are inside the contour for the first slice. If these edges were used, they would result in an undesirable slice that does not show the true shape of the object. To find interior edges, each edge is checked to see if the two facets that share this edge are parallel to the slicing plane. If so, the edge is interior to the contours of the object and will not be used for tracing.

The search method starts by selecting an unflagged edge in the slicing zone. The trigger point is found by comparing each of the unflagged edges in the zone to the following rules:

1. If the edge is completely above or below the slicing level, do not use this edge.
2. If either end point of the edge is on the slicing level, use this vertex as the trigger point.
3. If neither rule 1 nor rule 2, then this edge passes through the slicing level. Find the point on the edge at the slice height to use as the trigger point.

The contour is traced from this point moving from one facet to the next around the object until a contour is traced out. As each segment is created, the edges that were visited are flagged so they are not used again. Next, the search continues within the same level to find any remaining unflagged edges that form other contours on this level. It is important that searching continues through all of the facets on each slicing level to insure that no contours or features are missed.

Determining the Trigger Point Edge

At this step in the algorithm, a possible vertex or edge intersection has been found. To proceed, the edge that the point lies on must be determined. In addition, a validation process is completed with the edge information to ensure that the trigger point is actually on an edge of the facet. The edge determination and validation procedure is comprised of the following steps:

- Three vectors are created between each of the vertices of the facet as shown in Fig. 1.9.
- Three vectors are created between the trigger point and the vertices of the facet as shown in Fig. 1.10.
- The three trigger point vectors and the three vertex vectors are then crossed with each other, resulting in nine cross products.

$$Cross\ product\ =\ Point\ \overrightarrow{Vector}[\,] \times Facet\ \overrightarrow{Vector}[\,]$$

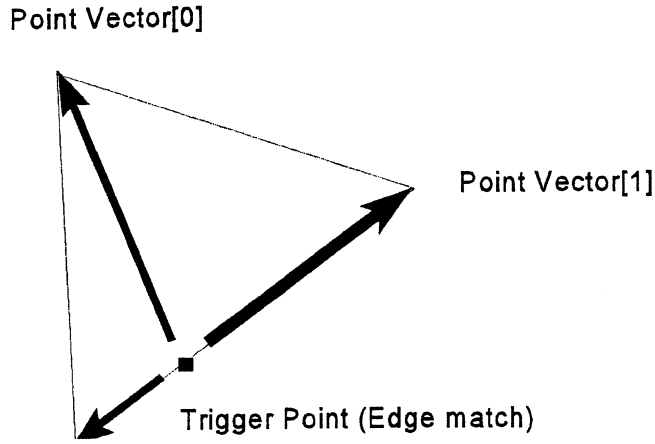


FIGURE 1.9 Vectors created between the trigger point and the vertices.

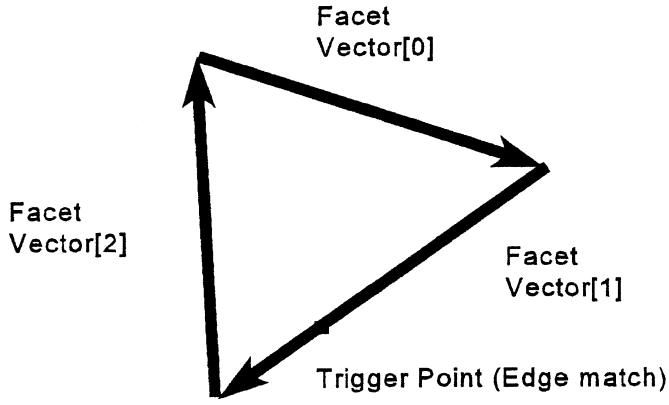


FIGURE 1.10 Vectors created between the vertices of the facet.

Whenever one of these cross products is zero, the trigger point possibly lies on that edge. In the case of a vertex trigger point, one of the point vectors will be zero. The other two point vectors will coincide with two facet vectors. This results in five of the nine cross products being zero.

As shown in Fig. 1.9, point vector[2] and point vector[1] are aligned with facet vector[1] from Fig. 1.10. When these vectors are crossed, there will be two zero cross products, as is the case for all edge points.

In Fig. 1.11, two examples of trigger points are shown. The first search through the unflagged edges results in a vertex trigger point at the corner of the object. Later in the search, a trigger point is found at the hole.

Tracing Out the Contour

Once a trigger point is found, a contour can start to be traced from this point. At each point, information is only known about the current facet and its neighbors. Because there is no other topological information to drive the creation of the contour, some procedures must be defined. In general, the contour will be constructed by creating line segments across each facet around the entire object.

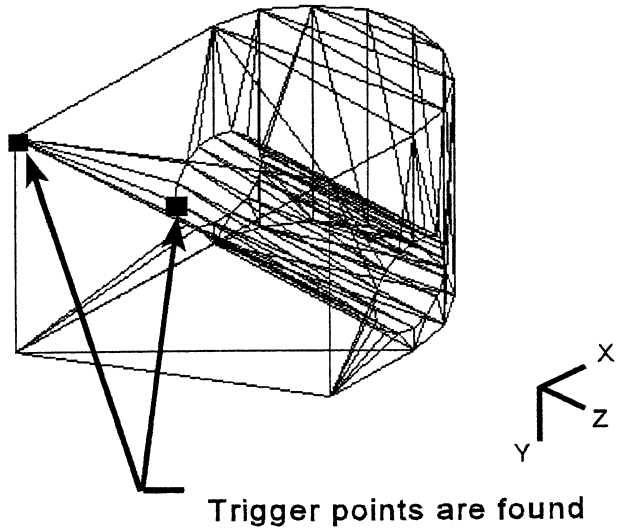


FIGURE 1.11 Establishment of trigger points on the object during the search pattern.

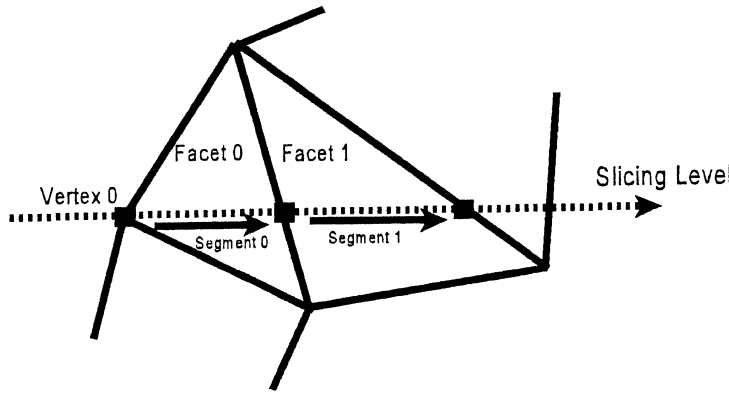


FIGURE 1.12 Contour segments created for vertex and edge trigger points.

Sample contour segments are shown in Fig. 1.12. The first trigger point from the search is on vertex 0 of facet 0. The contour will continue by tracing segment 0 to land on the other edge of the facet. Then facet 1 will be used with the "new" edge trigger point. Segment 1 will then be constructed from edge to edge of facet 1. The process continues around the object until the complete contour is traced.

When Tracing Begins from a Vertex

For vertex trigger points, there are three possibilities for tracing the next segment of the contour:

1. If the trigger point is "above" both of the other vertices, or completely "below" the other vertices, then the trigger point does not change and the next facet is used.
2. If only one of the other vertices is at the same z level as the trigger point, move all the way to that other vertex.
3. If the z level of the trigger point is between the z levels of both of the other points, the segment is constructed across the facet to an edge point between the other two vertices.

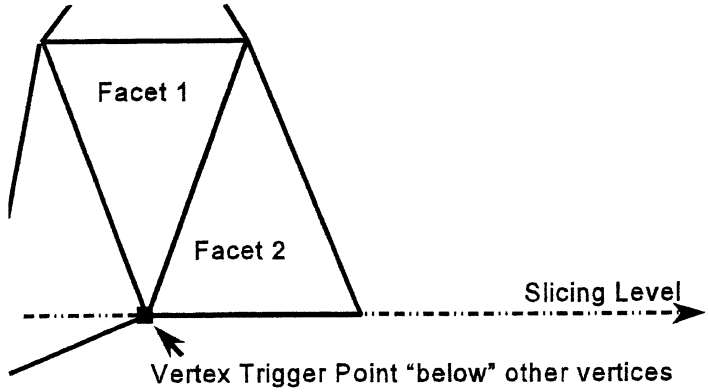


FIGURE 1.13 Vertex trigger point completely "below" the other vertices of the facet.

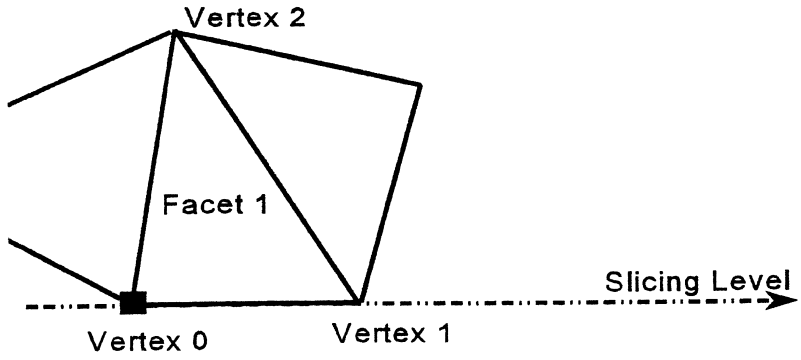


FIGURE 1.14 Another vertex of the facet lies on the same slicing level as the trigger point.

In the first case, the trigger point is on a vertex that is the "lowest" or "highest" point on the facet relative to the slicing level. As shown in Fig. 1.13, every other point on facet 1 is above the current trigger point; therefore, the contour will not move because moving to any other point on the facet would not maintain the current slicing level. The tracing will continue directly to facet 2 using this same point as the "new" point.

In the second case, another vertex of the facet (not the trigger point vertex) is on the slicing level. In Fig. 1.14, the trigger point is on vertex 0, while vertex 1 is also on the slicing level.

The new segment of the contour is created directly all the way to the other vertex that is on the same level. Because these two points are co-planar and on the slicing level, there is no need to do any computations to determine this segment.

In the third case, the facet is as shown in Fig. 1.15. The trigger point lies on vertex 0. The z value of vertex 0 is between the z values of the other two vertices. The contour must be traced "across" the facet, from the vertex to edge 1 on the facet.

When Tracing Begins from an Edge

For the case of an edge trigger point, vectors are used to move to the other edge of the facet. As shown in Fig. 1.16, the trigger point lies on edge 0. To move to edge 1, and stay on the slicing level, a similar

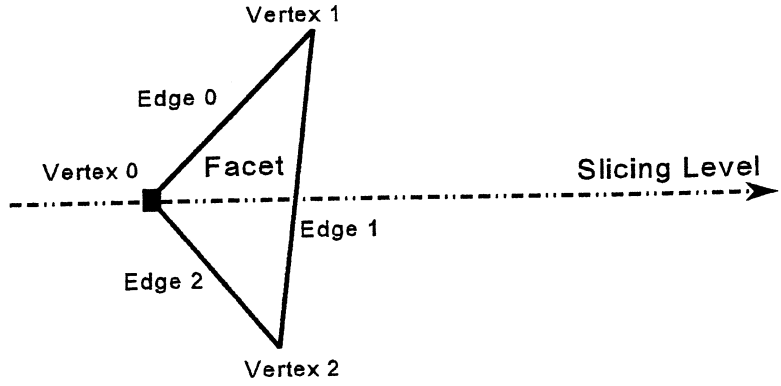


FIGURE 1.15 The vertex trigger point height lies between the levels of the other two vertices of the facet.

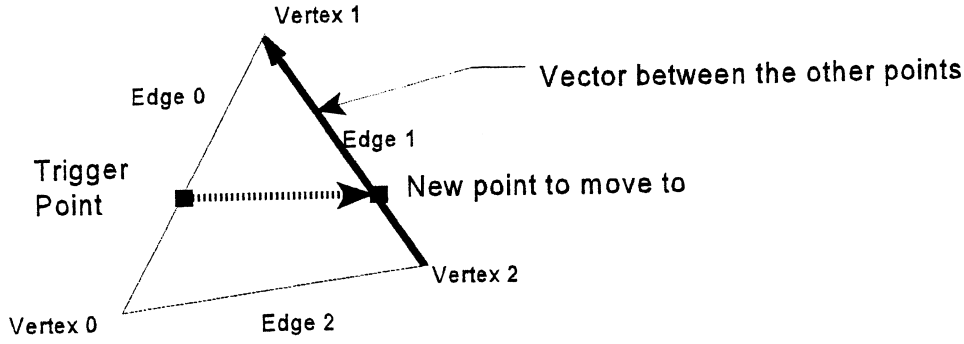


FIGURE 1.16 Moving from an edge trigger point to another edge.

method as implemented by Rock and Wozny (1991) has been followed:

- A vector is constructed from vertex 2 to vertex 1.
- A percentage of the vector is used to move from vertex 2 to the new point on the slicing level on edge 1.
- The new point is used as the trigger point for the next segment of the contour on the next facet.

1.5 Prototype Implementation and Case Studies

The Internet and corporate intranets provide a wonderful opportunity for collaborative design. Communication and modeling of a new product can be done over the Web with users participating from everywhere. This rapid prototyping module is one small part of the collaborative design process. Once the design has been finalized, this system can be called to create the prototype.

The methodology of Section 1.4 has been implemented in Java to run on the World Wide Web. Each section of the slicing algorithm has several parts associated with it. The sections of the program, including data collection, searching, and slicing, are explained here as they have been implemented.

System Architecture of the Prototype

The required modules for the slicing program are briefly outlined in Fig. 1.17. Each of these modules is executed in sequence to start the slicing procedures. Once a contour has been traced, the search module is called again to continue checking for other contours. The cycle repeats until the object has been completely searched.

Program Initialization

HyperText Markup Language (HTML), the language of the World Wide Web, is used to set up the Web page and launch the Java program. Two windows are created within the page, one running a program that displays the STL file and another that does the slicing. Once the programs are called, Java begins by initializing the window and calling the start method. The start and init methods handle the operations of the program, including setting display colors, adding buttons, and handling events that may be triggered during execution. Some of these events are button pushing, mouse movements, and mouse clicks. After the initialization, the start function calls the rest of the modules needed for the program's execution.

Data Input and Manipulation

The entire slicing process starts by gathering the necessary data, proceeding as shown in Fig. 1.18.

The first procedure is to retrieve the STL information. A class called triangle, shown in Fig. 1.19, is created to store the facet information and contains variables to hold the points of the three vertices as well as the normal of the object. A method creates triangle objects to hold all of the facets and stores each triangle in an array. The class also contains a variable to store the facet number of the three adjacent facets to each edge of the triangle.

One of the advantages of Java is the use of the vector class. This class allows the program to create an array to the required size only. Because each STL file will have a different number of facets, it is important to have enough triangle objects while minimizing computer memory usage. With the vector class, an object is created with 20 slots. Each triangle object is added to the vector as the data is read in. If there are more facets, the vector object grows by 10 facets at a time. Finally, the object is capable of trimming to size so that no extra memory

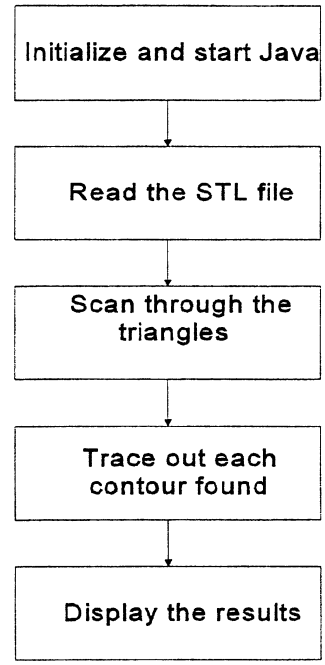


FIGURE 1.17 Flow of information in the prototype system.

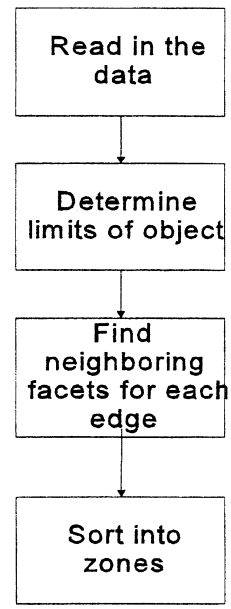


FIGURE 1.18 Preprocessing of the STL file before slicing.

```
class triangle {
   double[][] point = new double[3][3];
   double[] normal = new double[3];
   int[] neighborFacets = new int[3];
   int[] edges = new int[3];
}
```

FIGURE 1.19 Portion of Java code for the triangle class.

```
Vector facetVec = new Vector(20,10);

try {
  URL url = new URL(getDocumentBase(), "FILENAME.STL");
  InputStream in = url.openStream();
  DataInputStream is = new DataInputStream(in);
    readTheData();
  facetVec.addElement(newFacet);
}
catch(Exceptions);

facetVec.trimToSize();
```

FIGURE 1.20 Java code that opens the STL file and stores the data in a Vector array.

is used up by an array that is larger than the number of facets. The initialization and trim methods are shown in Fig. 1.20. This figure also shows the code to open a file on the server and read in the data.

A method is used to determine the minimum and maximum points on the object. All of the vertices of the faceted object are compared to an initial minimum value. If the new point is less than that value, it is selected as the new minimum point. This continues until all of the vertices have been checked. Finding the maximum point is done in a similar manner.

To compensate for some of the deficiencies in the STL file, some connectivity information must be determined. Each facet has three edges that it shares with three other facets. By searching through the STL file, the connections between facets, and their edges, can be saved for future use. A sort is used to find the neighboring facet for each edge. These adjacent facets are stored in the neighbor facets array which has three slots, one for each edge. The neighbor information is used during contour tracing to move from one facet to the next. Without sorting and storing the adjacent facets, every facet would need to be searched after finding each segment of every contour—an exhaustive computational procedure.

Search Method to Find Trigger Points

The search method scans for an intersection point with the object and follows the procedure in Fig. 1.21. Once the facets are stored, the slicing begins on the lowest level of the object. For each slice height, the appropriate zone is consulted to search for unflagged edges. Each edge that is unflagged ($a - 1$ value) is checked against the search rules to determine a trigger point.

For edges that pass through the slicing layer, the trigger point is found by first constructing a vector from end point to end point of the edge. Then the trigger point is found by moving a percentage of this vector until reaching the slicing level. The entire search method is described by the following pseudo-code:

```
method ScanTriangles()
  set DX, DY, DZ
  for each slice height do
      determine zone
      while there are unflagged edges in zone
          find trigger point on edge
          traceContour()
  nextslice++
endMethod
```

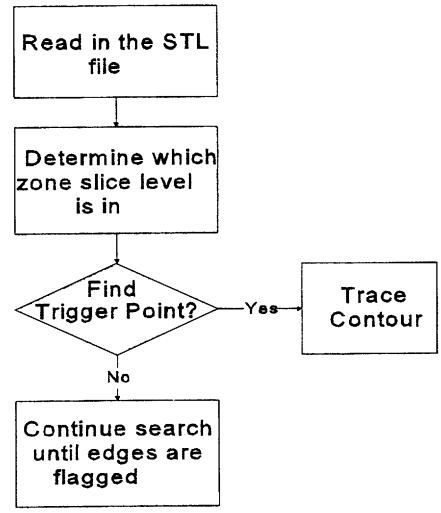


FIGURE 1.21 Initialization and scanning of STL file.

Once a trigger point is determined, the edge or edges that it lies on are found. The cross products are found for the facet edge vectors and the trigger point vectors. Cross product information is also used to verify that this point is a vertex or an edge point, as will be shown in the next section.

Contour Tracing Implementation

The trigger point and current facet are sent to the contour tracing method. The first step for the trigger point and facet is a validation process to ensure that the point does lie on a vertex or an edge of the facet. By keeping track of the cross products that are zero, the following validity check is then done:

- For a vertex match, two of the edges will have two zero cross products each, and one edge will have one. The sum of all the zero cross products will be five.
- In the case of an edge match, one of the edges must have two zero cross products, while the other two edges do not. The sum of all the zero cross products will be two.

If either of the conditions above are not met, the contour tracing is stopped and program control is returned to the search method. For either type of matching, the tracing starts on one point of the object and continues from one facet to another until it returns to the starting point. Once the contour is traced, it is stored for subsequent processing.

```
method trace_contour
    while contour_start != segment start do
            retrieve facet info (edges, normals, and points)
            find next intersection
            store line segment
            next intersection = segment start
    end
```

To prevent the same contour from being traced again, the edges of the facets are flagged when they have been used in a contour. There is one flag for each edge in a triangle object. When these edges are used, by a trigger point or a line segment, the flag is set so the edge is not used again. These flags show which edges of the facets have already been visited. There are four methods used for flagging: isFlagOn, flagEdge, eraseContour, and eraseAllEdges. An edge is unflagged when its value is −1.

The tracing may result in several contours on one layer, each of which is numbered. The flagEdge method sets the variable for that edge to the contour number. The isFlagOn method checks to see if the flag is on for that edge. If so, it returns control back to the search program. In the event that a contour is not completely traced out, the flags must be cleared. The eraseContour method does this by comparing all of the flagged edges with the current contour number and resetting them to -1, the unflagged state. Once all contours have been traced, and the program is ready to move to the next slicing level, all of the edges are returned to the unflagged state with the eraseAllEdges method. All trigger points and contour segments are checked throughout the tracing process to see if they fall on a flagged edge.

Vertex Trigger Point Contour Tracing (Cases 1 and 2)

Each of the first two cases of the vertex trigger point are handled in a straightforward manner. In Case 1 above, where the trigger point lies completely above or below the other vertices of the facet, the trigger point remains the same and the next facet is used. The neighbor facets array is consulted to find the next facet to use. The point becomes the trigger point on the new facet.

In Case 2 above, the program moves the contour from the vertex to the new vertex that is on the same slicing level. This new vertex is used as the trigger point for the new facet. In both cases, the edges that the trace has been on are flagged so that they are not used in future contour searches.

Edge Trigger Point and Vertex Case 3 Trigger Point Tracing

Case 3 above is the same method as edge trigger points. To create the new segment, the z terms of the other vertices are compared with the trigger point and the edge that will maintain the slicing level is chosen.

Because the tracing method will now move to the other edge, it must determine which point to move to on that edge. A vector is constructed between vertex 0 and vertex 1 of Fig. 1.22 and the new point is found by moving a percentage of that vector to maintain the required slicing level. A variable is set to determine if the trigger point is above or below (z level) the end of the vector. The new point is now found by starting at the other vertex and moving by the vector multiplied by the percentage. The new edge trigger point is used on the next facet.

End of Each Segment

After the new point has been determined and the next facet found, the information for our new coordinates is written to an array in the addToContour() method. The new point is checked against the starting point of the contour. If the two points are the same, the contour is complete and the tracing is stopped. Otherwise, the new point is used as the trigger point on the next facet. The process continues until the program traverses completely around the contour to the starting point.

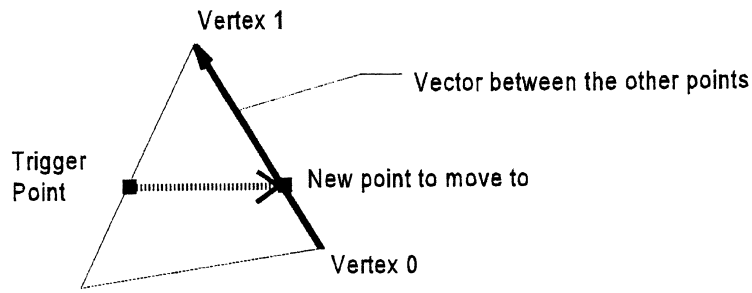


FIGURE 1.22 Moving to a new edge.

World Wide Web Access

The applets for viewing the STL file and for slicing it are stored on a Web server. These two programs can then be started by loading a Web page which calls the two programs. The HTML code for the Web page first splits the page into two windows. The Web browser will load the view class applet in the left window first and read in the STL file and display it on the screen. Then the sliceclass applet starts in the right window by reading in the file and beginning the slicing procedure.

Results of Enhanced Algorithm

The output from several samples are shown in this section. All of the examples were run on a Pentium 90-MHz computer with 16 MB of RAM using Netscape 2.0. A discussion of the meaning of these results follows the case studies.

Case Study 1

The first test object was the simple polyhedra with a hole that has been shown throughout this work. This is a simple object that has 116 facets. As can be seen in Fig. 1.23, most of the facets are used for the hole and the rounded rear surface. In the right window, a few slices of the object are shown.

This type of object uses vertex matching for the lower and upper slices, while in between it uses edge matching to find the contours for the hole.

Case Study 2

Figure 1.24 depicts a more complex object. In this case, there are no flat edges parallel to the slicing plane. All of the slices are created primarily through edge intersections.

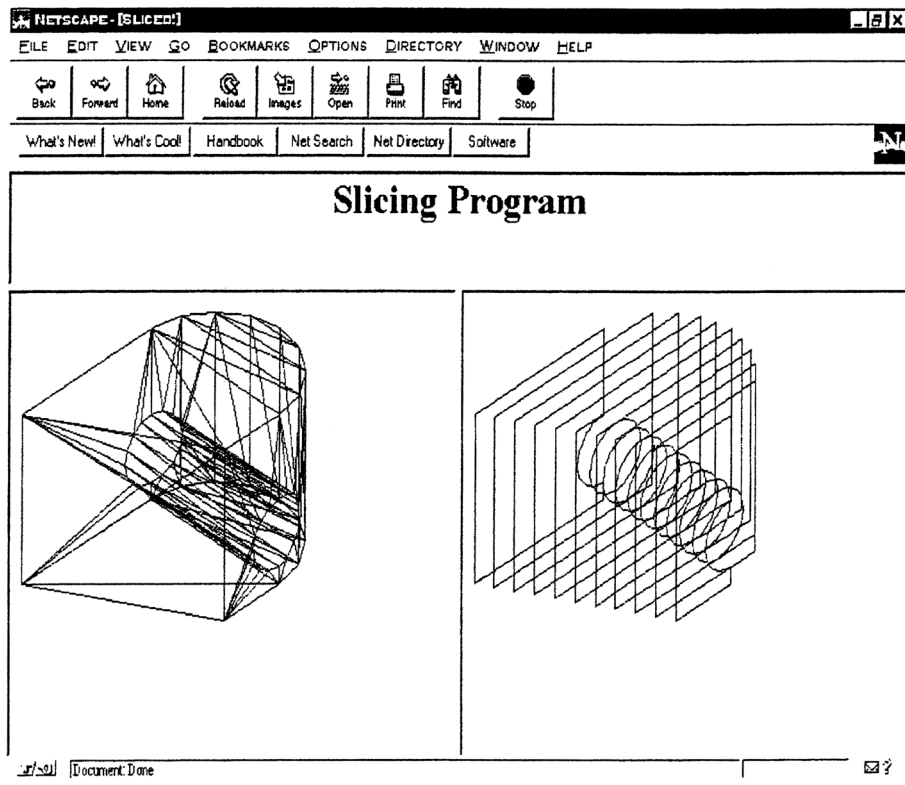


FIGURE 1.23 Simple polyhedra with a hole that has been sliced.

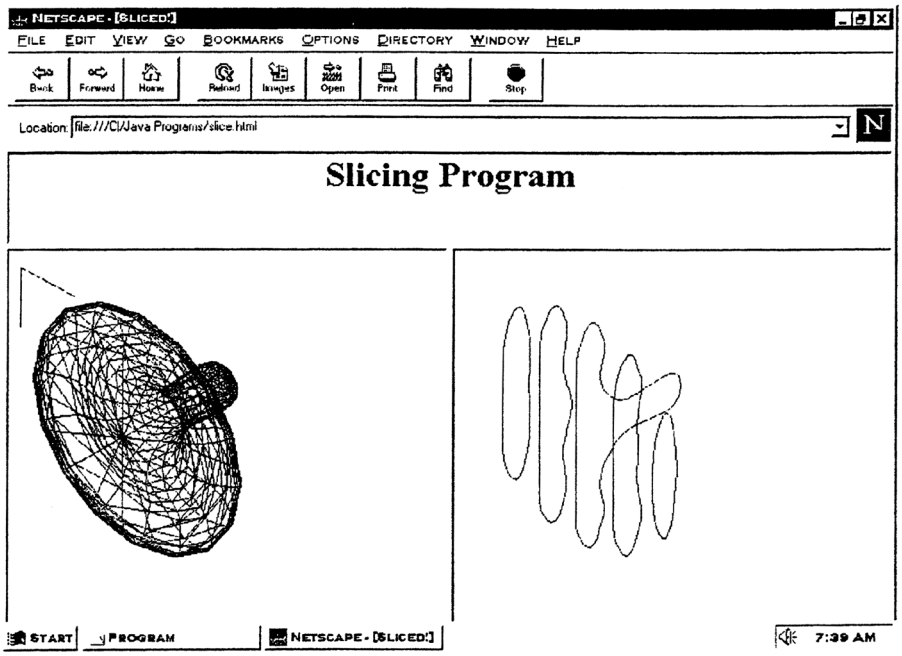


FIGURE 1.24 A more complex object that requires more edge point calculations.

The tessellation of this object is shown in the left window; the protrusion is going away from the viewer. As displayed in the right window, even difficult contoured surfaces can be traced quickly.

Case Study 3

This example shows a pinion part which has a number of features. The pinion itself is a long protrusion with a hole through the center and a number of teeth around the outside of the shaft at different angles. The output of the program is shown in Fig. 1.25.

Case Study 4

In the final example, a more complex and resulting larger STL file part is used. This is a gear that has several protrusions to comprise the shaft, as well as a pattern of gear teeth around the radius of the gear. The STL file has over 2600 facets associated with it. The sliced object is shown in Fig. 1.26. The scan-line type approach was implemented using both the exhaustive search method (Chalasani et al., 1991) and the modified unflagged edge starting method. The times presented in Fig. 1.27 were noted for the objects shown in the case studies.

The times noted reflect the improvement of utilizing unflagged edges to determine a trigger point to trace from. The exhaustive search method is affected by the number of facets, the size of the part, and the resulting facet vertices precision. The search will not find suitable matches unless the search increment is small enough to match the data in the STL file. The unflagged edge method overcomes this limitation by determining a point that lies on an edge that passes through the slicing plane. Then, only the edges that have not been used yet are consulted for further possible contours, a much faster process.

In comparison with the work reported by Rock and Wozny (1991), savings are found in two places. In one improvement, slicing is enhanced in areas of the object where the surfaces are parallel to the slicing plane. The bottom slice of Fig. 1.28 is generated using contour tracing case 1 and 2 only, which require very little computation time.

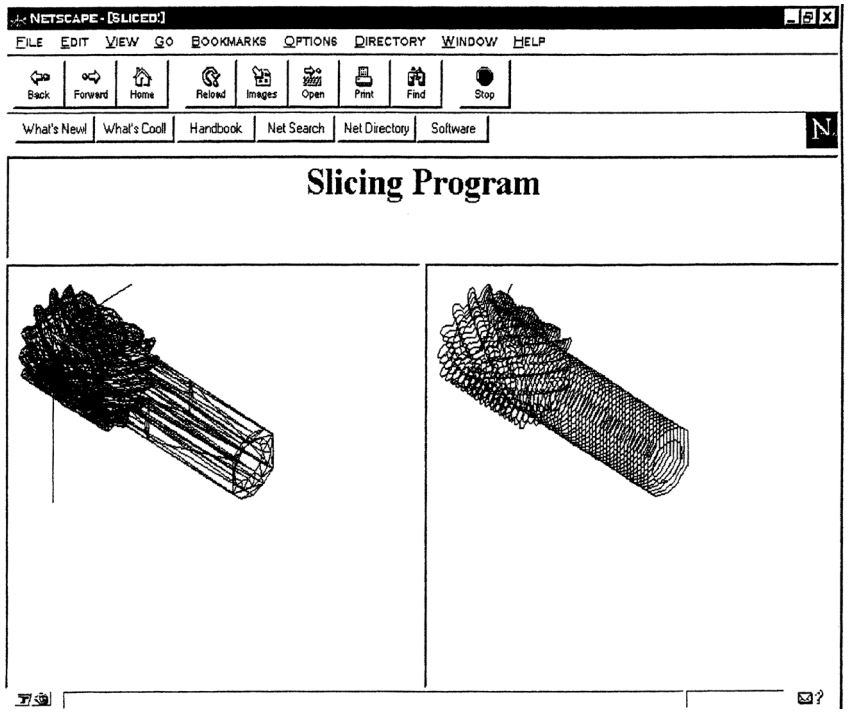


FIGURE 1.25 Pinion part with a number of features shown sliced in right view.

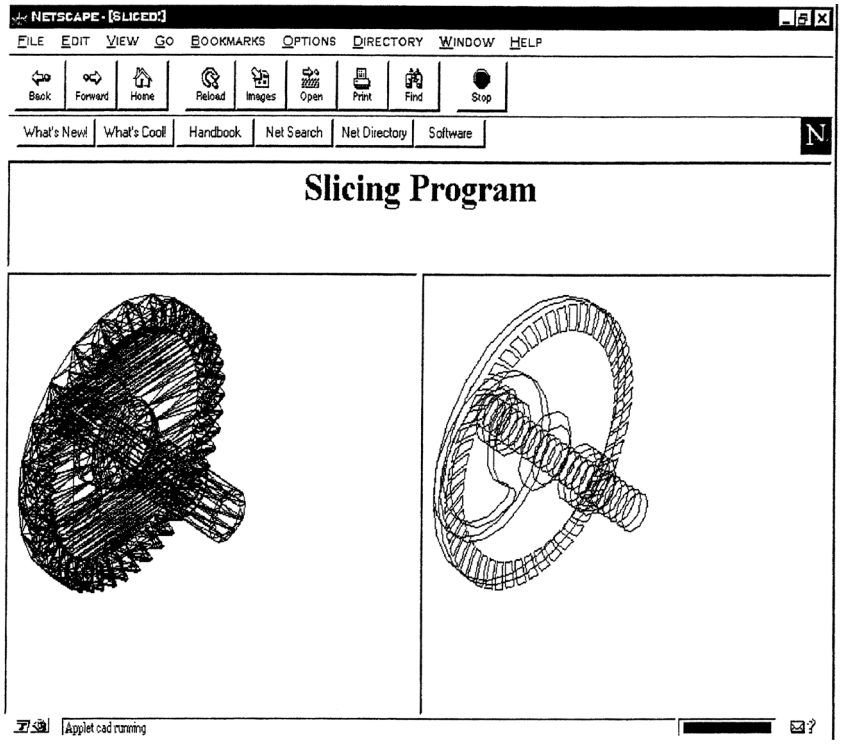


FIGURE 1.26 A more complex gear part.

Case Study Number and Description	Time to Create 10 Slices of the Object Using Search Method to Determine Trigger Point[a]	Time to Create 10 Slices of the Object Using Modified Unflagged Edge Method
1 - Polyhedra with hole and rounded rear surface, (see Fig. 1.23) - 116 facets	11.2 minutes	43 seconds
2 - Revolved surface (see Fig. 1.24) - 832 facets	1.1 hours	1.95 minutes
3 - Pinion gear (see Fig. 1.25) - 1468 facets	2.3 hours	9.2 minutes
4 - Main gear (see Fig. 1.26) - 2684 facets	3.4 hours	14.3 minutes

[a]Based on search increments to 1/100th of object size (x and y-direction).

FIGURE 1.27 Comparison of Slicing Times for Standard and Enhanced Scan-line Type Algorithms.

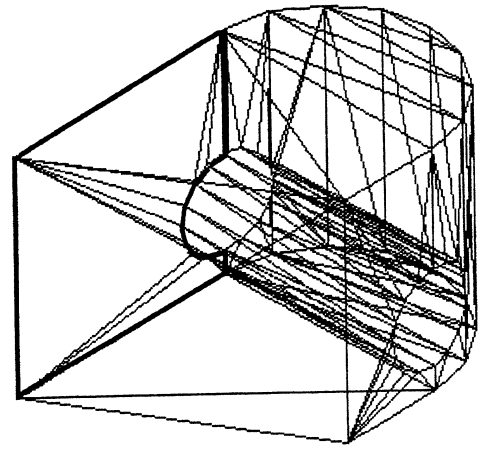


FIGURE 1.28 Facets parallel to the slicing plane can be traced out quickly.

The second improvement is the method of determining the starting point for tracing. In the plane-plane intersection method, the normals of the slicing plane and normal of each facet are crossed to determine a vector parallel to their intersection. Then, this vector and a point on the facet are used to find this line segment. Using the modified unflagged edge method, these computations are greatly reduced.

1.6 Conclusions

The proposed algorithm and its implementation have shown an alternative method to extend the rapid prototyping process. By utilizing the World Wide Web and Java, a slicing algorithm has been constructed that can be available to anyone. There are still many areas of research and work that need to be continued to make the process more efficient.

Each of the CAD models once translated to an STL file, can be read and sliced by this program and displayed through the VIEW.JAVA program. Once displayed, the slicing program begins the process and displays it on the Web page. The Web-accessible program allows designers to see the faceted and sliced object without the need for expensive software or dedicated hardware at their physical site. All of this can be accessed over the World Wide Web, allowing the possibility of a completely sliced file to be sent directly to a layered manufacturing machine connected to the Internet or intranet. The work presented here emphasizes four issues of rapid prototyping on the Internet or intranets:

1. Display of an STL file into a local browser showing a projected view of the object that can be rotated
2. Implementation of a modified scan line-contour tracing algorithm to slice the object
3. Platform-independent, sharable software that can be accessed over the Internet/intranet, as well as being able to run locally on a machine; the abilities of the Internet and collaborative design have been extended to the rapid prototyping process
4. Utilizes Java as a first step toward creating a true Internet-manufacturing process

Additionally, this software is one of the services within the collaborative product design environment. This module can be called after the CPD evaluation of an object to start the prototyping process.

The framework for this software can be further improved. The graphics of the program can be extended into a complete design environment to cover the entire rapid prototyping process. Viewing of the object could be enhanced with shaded views or views of the completed prototype showing the cusp height. The software will need to be extended to create the hatching patterns and the file that actually drive the layered manufacturing machine. A module of this type could be added directly to the program at any time using the contour information.

The continued expansion of the World Wide Web lends this program to possible connections with VRML or collaborative design environments. The Web is an open-ended medium that can extend the method for design and research. Modules could be added to the software to allow other information to be obtained as well.

There are many areas of research still left unsolved in the rapid prototyping world. Research areas exist in part orientation, slicing directly from the CAD file (no STL file), and multiple part optimization. Additional work will be done in the area of producing more accurate parts directly from the CAD file. This will require a new common part storage instead of the STL file; perhaps some of the emerging formats that extend STEP will take its place.

Ultimately, rapid prototyping will evolve into rapid manufacturing. Many of the parts on those systems are used "as is" for tooling or in a product. As software and hardware increase in speed and accuracy, additional parts can be made for instant use. Additionally, by harnessing the power of the World Wide Web, the entire prototyping process can reach new levels of power and application in today's design process.

References

Bailey, Michael, 'Tele-Manufacturing: Rapid Prototyping on the Internet with Automatic Consistency Checking', University of California at San Diego.

Chalasani, K.L. et. al., 'An Algorithm to Slice 3-D Shapes for Reconstruction In Rapid Prototyping Systems', *Computers in Engineering,* V 1, ASME 1991, 209–215.

Cornell, Gary, and Horstmann, Cay S., *Core Java,* SunSoft Press, Prentice Hall, 1996.

Dolenc, A., 'An Overview of Rapid Prototyping Technologies in Manufacturing', *http://www.cs.hut.fi/~ado/rp/rp.html,* Helsinki University of Technology, 1994.

Dolenc, A., Mäkelä, I., 'Slicing Procedures for Layered Manufacturing Techniques', *Computer Aided Design,* Vol 26, Number 2, February 1995, 119–126.

Frank, Dietmar, and Fadel, Georges, 'Expert System Based Selection of the Preferred Direction of Build for Rapid Prototyping Processes', IWB, Technical University of Munich, Munich, Germany.

Kirschman, C.F. et. al., 'Computer Aided Design of Support Structures for Stereolithographic Components', *Proceedings of the 1991 ASME Computers in Engineering Conference,* Santa Clara, CA, August 1991, 443–448.

Kirschman, C.F., 'Error Compensation of Stereolithographic Parts', Clemson University, Clemson, SC.

Kirschman, C.F., Jara-Altamonte, C.C., 'A Parallel Slicing Algorithm for Solid Free Form Fabrication', *Proceedings of the 1992 Solid Freeform Fabrication Symposium,* Austin, Texas, August 1992.

Kulkarni, Prashant and Dutta, Debasish, 'Adaptive Slicing of Parametrizable Algebraic Surfaces For Layered Manufacturing', DE-Vol 82, 1995 *Design Engineering Technical Conferences,* V 1, ASME 1995, 211–217.

Ngoi, B. et. al., 'Development of a Stereolithography Preprocessor for Model Verification', *Computing and Control Engineering Journal,* October 1993, 218–224.

Rajagopalan, M. 'An Approach to CAD Interface for Stereolithography Using Non-Uniform Rational B-Splines', Master of Science Thesis, Clemson University, Clemson, SC, 1992.

Rock, S, and Wozny, M., "Utilizing Topological Information to Increase Scan Vector Generation Efficiency", *Proceedings, Solid Freeform Fabrication Symposium,* University of Texas at Austin, Austin, Texas, August 1991.

Vuyyuru, P. et. al., 'A NURBS-Based Approach for Rapid Product Realization', *Proceedings of the Fifth International Conference on Rapid Prototyping,* Dayton, Ohio, June 1994.

Wodziak, J., Fadel, G., Kirschman, C., 'A Genetic Algorithm for Optimizing Multiple Part Placement to Reduce Build Time', *Proceedings of the Fifth International Conference on Rapid Prototyping,* Dayton, Ohio, June 1994.

2

Models and Algorithms for Machine Scheduling with Setup Times

2.1 Introduction

2.2 Development and Solution of Mathematical Scheduling Problems2
Models of Jobs and Operations • Models of Machines and Resources • Usual Assumptions in Scheduling Models • Objective Functions • Models of Setup Times • Scheduling and Lot-Sizing • Algorithms and Complexity • A Classification Scheme for Scheduling Problems • Reducibility Among Scheduling Problems with Setup Times

2.3 Makespan and Flowtime on Single Machines
Minimizing Makespan with Sequence-Dependent Setup Times • Minimizing Makespan in Class Scheduling Problems • Flowtime Problems with Sequence-Dependent Setups • Additive Changeovers on a Single Machine • Flowtime Problems with Sequence-Independent Setup Times • Flowtime Problem Extensions

2.4 Summary

Simon Dunstall
University of Melbourne

Andrew Wirth
University of Melbourne

2.1 Introduction

Effective scheduling is essential in a vast number of areas of industrial and commercial activity, where tasks must be accomplished using resources that are in limited supply. The degree to which the objectives of an organization can be achieved is often greatly influenced by the quality of the schedules followed. The quality of these schedules is in turn a direct result of the strength and appropriateness of the scheduling methodologies utilized.

In many situations, mathematical methods can provide indispensable scheduling tools. This chapter explores the use of mathematical methods in industrial scheduling environments, discussing the modeling techniques and solution procedures that are commonly reported in operations research literature. In doing so, this chapter takes a close look at a selected set of scheduling problems involving setup times.

The major objectives of this chapter are:

1. To present the essential elements of mathematical scheduling problems
2. To introduce various models of setup times and discuss the difficulty they bring to the solution of scheduling problems
3. To survey scheduling research applicable to certain scheduling problems involving setup times

4. To address a wide range of mathematically based scheduling methods, in the process of undertaking the above
5. To assess the applicability of various models and algorithms to different scheduling environments

This chapter was written with the overall aim of providing useful insight for both scheduling researchers and those less familiar with mathematical scheduling approaches. Given the vast amount of literature on scheduling with setup times that now exists, this chapter can only serve as a starting point for a study of this field.

2.2 Development and Solution of Mathematical Scheduling Problems

The formulation and solution of a *mathematical scheduling problem* stems from a desire to control and optimize the performance of a real-world scheduling situation. A mathematical *model* is constructed to represent this situation, sufficiently complex to include the key aspects of the problem, yet simple enough to be amenable to a good solution. An appropriate set of scheduling aims is identified and typically expressed as a mathematical *objective function* whose value represents the quality of the solution (and is to be optimized). The objective of finding a feasible solution is less commonly reported in scheduling literature, but is appropriate when dealing with highly complex and constrained systems. A *solution methodology* is then adopted in an attempt to optimize the value of this function, subject to the relationships and constraints arising from the model chosen.

This chapter concentrates on scheduling problems that involve *deterministic data*—data that is assumed to be perfectly accurate and complete. In practical terms, this means that scheduling data such as the processing times of jobs and the availability of machines are expressed as a "best estimate" and *deterministic scheduling models* make no allowance for uncertainty in these estimates. As such, one may need to be conservative in estimating data if, for example, processing overruns can lead to schedules becoming infeasible. Unplanned and unpredictable events, such as machine breakdowns or arrival of new jobs, are not considered in deterministic scheduling models.

It can be observed that the assumption of deterministic data is a common one. For example, *Material Requirements Planning* (*MRP*) uses deterministic data, as do *Critical Path Methods* (*CPM*). On the other hand, the *PERT* technique for project management uses *stochastic data*; each task duration is assigned an expected value and a variance (derived from pessimistic and optimistic estimates). A restricted range of *stochastic scheduling models* and algorithms have been reported in the scheduling literature.

A typical scheduling model of an industrial scheduling environment is comprised of a model of the processors (*machines*) used to carry out tasks, a *machine model*, and a model of the tasks (*jobs*) themselves, a *job model*. Essential elements of a machine model are the number of machines, their configuration, their capability, and their processing capacity. Machines may often be items of industrial equipment, yet the use of the term "machine" is traditional and does not exclude the modeling of workers or computer systems, for example. A job model will incorporate properties of individual jobs such as processing characteristics, product type, number of items, total processing time and job due dates. If necessary, the job model will also address relationships between jobs, such as precedence. The attributes of the job model will usually be determined or revised on each scheduling occasion (to allow for customer orders, for example). In contrast, the machine model will often remain unchanged.

Where it is required that the durations of setup times appear in a schedule, the job model and machine model are complemented by the *setup time model*. In a scheduling problem, a model of setup times essentially incorporates (1) a logical structure that is used to determine whether a setup is necessary, and (2) a means of calculating the duration of a setup when required. A range of setup time models has appeared in the scheduling literature, and the initial focus of this chapter will be a survey of a number of these setup time models.

Scheduling problems with setup times included in schedules can be compared to scheduling problems that may consider setups but do not involve the *scheduling of setups*. Problems of the latter kind are not of immediate interest in this chapter.

Inclusion of setup times in a scheduling problem greatly increases the practical applicability of scheduling problems. Commenting (in 1971) on the apparent gulf between theoretical scheduling research and industrial scheduling environments, Panwalkar, Dudek, and Smith [65] conclude from the results of an industrial survey that the applicability of scheduling algorithms can be greatly enhanced by the inclusion of setup times in scheduling problems. Since the time of this survey by Panwalkar et al., a sizeable body of research into problems with setup times has been generated by many researchers.

There exists a broad range of *solution methods* available for the solution of mathematical scheduling problems. An *algorithm* is a procedure followed in order to arrive at a solution, and is based on one (or more) of these solution methods. Scheduling research focuses on the development of effective algorithms. The process of algorithm development is based on an analysis of the *structure of the problem*, followed by the adaptation of a particular solution method. Some algorithms are assured of returning the optimal solution to the problem (*optimal algorithms*), while others do not come with such a guarantee (*approximation algorithms* or *heuristics*).

The primary measures of algorithm performance utilized in scheduling literature are (1) the time required to arrive at a solution (*running time*), and (2) the quality of the schedules produced, as measured by a mathematical *objective function* deemed appropriate for the situation. It is common for a compromise to be necessary between these two measures, due to the inherent difficulty of scheduling problems.

The difficulty in scheduling problems lies in their *combinatoric* nature. Finding the correct order in which to process the jobs involves making a selection from an extremely large population of solutions. For example, a scheduling problem that involves sequencing N jobs can have as many as $N!$ potential solutions. While it is possible to enumerate all possible solutions when the "size" of the problem is very small, the time required for this approach ordinarily becomes unreasonable for any scheduling problem of meaningful size. Effective algorithms, therefore, must be structured in a way that allows them to locate good (near-optimal) solutions with the minimum computational expense.

It is typical even for advanced mathematical scheduling problems to be simplistic in comparison to the "real-world" scheduling problems that they represent. This is due to the difficulties associated with computing solutions to the "large" mathematical problems in a reasonable amount of time. Other common criticisms of mathematically based scheduling approaches include difficulties in reacting to updates and changes in scheduling information, in adequately dealing with conflicting objectives, and general user-unfriendliness. In contrast, typical advantages of mathematically based scheduling approaches are the amenability of algorithms for computing, optimality or near-optimality of solutions, the ability to handle otherwise incomprehensibly large amounts of data, consistent and repeatable performance, and (in some cases) relative speed of schedule generation. Additionally, analysis of mathematical scheduling problems can provide important insights into the "structure" of a scheduling problem, leading to the development of improved scheduling rules and strategies.

Successful scheduling often relies on an integration of mathematical scheduling with other sources of knowledge, analysis, and judgment. To this end, Morton and Pentico [63] describe four major complementary approaches to scheduling:

1. Improved methods for training human experts
2. Expert systems to imitate human experts
3. Mathematical scheduling systems
4. Hybrid systems combining the strengths of other approaches

The same authors categorize scheduling problems into levels, differentiating between classes of problem according to the scope of the scheduling decisions and the types of decision carried out. The five levels they present are summarized in Table 2.1. Lawler, Lenstra, Rinnooy Kan, and Shmoys [53] suggest a three-level categorization encompassing approximately the same range, using the terms strategic-level, tactical-level and operational-level scheduling.

This chapter concentrates on mathematical scheduling models for operational-level scheduling (i.e. the final two levels shown in Table 2.1). Much research into scheduling problems with setup times has been directed at this level of scheduling.

TABLE 2.1 Levels of Scheduling

Scheduling Level	Examples of Problems
Long-range planning	Facility layout and design
Middle-range planning	Production smoothing, logistics
Short-range planning	Requirements planning, due date setting
Scheduling	Job shop routing, lot-sizing, machine scheduling
Reactive scheduling/control	Urgent jobs, down machines, late material

Source: Adapted from Morton and Pentico [63].

In scheduling problems, the choice of *scheduling horizon* (interval into the future to which scheduling decisions are to apply) is influenced by (1) the natural time-scale of the system being modeled, (2) the availability of data regarding the activities that must be completed, and (3) the difficulty of the scheduling problem (associated problem size restrictions can lead to small horizons being chosen). Essentially, choice of scheduling horizon represents a trade-off between the value of scheduling to a "long" horizon and the cost associated with this choice.

The intended application of mathematical scheduling approaches is usually the scheduling of an existing and currently operating production facility. There are, however, other applications of mathematical scheduling approaches, typically within simulation experiments, such as:

- Determination of the true and optimal capacity of an existing or proposed production facility, this being a function of the scheduling approach, the processing characteristics of the products and the "mixture" of products being produced
- Economic and technological analyses of the effect of modifications to a manufacturing system, such as reduction in setup times, increases in production rates, or revision of process plans

These applications provide important opportunities for mathematical scheduling approaches. The typical practical disadvantages of mathematical scheduling approaches may often be less significant for these applications, while the usual advantages of these systems may be well exploited.

Models of Jobs and Operations

In general, a *job* is comprised of a number of steps, or *operations*, each of which needs to be carried out in some part- or fully specified order.

In a *multi-operation model*, each operation has various processing details associated with it (e.g., work content and machine identification number) and represents one production task. A job is a collection of these operations and corresponds to some "thing to be accomplished." For example, if a job represents a customer order placed for a set of items of product X, each operation may represent one step in the manufacturing process for product X.

By contrast, jobs in a *single-operation model* consist of one operation only, and it is customary not to distinguish between a job and an operation but instead to use the term "job" universally.

In a mathematical scheduling problem, each job will usually be assigned a numerical index; for example, an N job problem may have jobs indexed from 1 through N. In some cases, a job is conveniently referred to using its index only, for example, "job 2" or "job j," while in others the jth job may be represented by a_j or a similar symbol. The set of jobs to be scheduled is often denoted by $\mathbf{J}$, with N being commonly used to denote the number of jobs in this set; for example, $\mathbf{J} = \{a_1, a_2, \ldots, a_N\}$ represents a set of jobs when the a_j referencing style is in use.

Where a job is comprised of a number of operations, the kth operation of the jth job can be denoted by $a_{j,k}$. Where the set of jobs is divided into B mutually exclusive and exhaustive subsets (i.e. $\mathbf{J} = \cup_{i=1}^{B} \mathbf{J}_i$), the jth job of the ith subset can be denoted by $a_{i[j]}$ and the total number of jobs in subset $\mathbf{J}_i$ denoted by $N_i (N = \sum_{i=1}^{B} N_i)$. These means of denoting jobs are summarized in Table 2.2.

Particular subsets of $\mathbf{J}$ often represent distinguishable *classes* of jobs. When scheduling with setup times, the formation of classes will typically represent the logical grouping of jobs according to similarities in the processing requirements of jobs. Hence, setup times can be reduced by scheduling jobs of the same class together.

TABLE 2.2 Job Referencing Summary

Problems without classes	jth job	a_j
Problems with classes	jth job of ith class	$a_{i[j]}$
Multi-operation models	kth operation of jth job	$a_{j,k}$

Associated with a job will be data (*job properties*) relevant to the scheduling problem. The job properties surveyed in this section will be the most common of these, and are discussed in terms of the single-operation model.

A common job property is the *processing time p* of the job. This is the work content of the job expressed in units of machine time. Processing time is alternatively referred to as a *processing requirement* (or similar), and the latter is a more appropriate term within problems where, for example, dissimilar machines in a bank of parallel machines have different production rates. The use of the symbol p_j for the processing time of a job is consistent with the above a_j notation for jobs.

Another typical job property is the *job weight w*. This weight corresponds to the importance of a job, as evaluated by some measure, and is used within an objective function to produce a schedule that accounts for the relative importance of jobs.

The *ready time* (or *release date*), *r*, is the instant at which a job becomes *available*. Processing of a job cannot begin on any machine prior to the job's ready time. If all jobs have identical ready times, the problem is said to involve *static job arrivals* and is called a *static problem*; otherwise, *dynamic job arrivals* are being modeled and the problem is a *dynamic problem*. Although they have restricted practical applicability, static problems are more common than dynamic problems in scheduling literature. The increased difficulty of scheduling problems involving ready times would appear to be the primary reason for this.

It is customary to view a *due date d* as a "target" instant by which processing of the job should be completed, while a schedule is considered infeasible if a job is completed after its *deadline* $\bar{d}$. Many frequently applied scheduling objectives, such as the minimization of maximum tardiness or number of late jobs, naturally involve due dates. In contrast, job deadlines appear in scheduling constraints rather than objectives.

Jobs may consist of a number of identical *items*, these items being examples of a particular *product* produced by the facility. It is commonly assumed that there is no need to consider the individual items in a job; rather, the job is treated as either indivisible or continuously divisible. If, during a particular operation, the processing of an item must be finished once it has begun, these common cases are equivalent to (respectively) one-item-per-job and infinite-items-per job.

Precedence constraints are often important features of scheduling problems. Precedence constraints can be applied between jobs (e.g., job a_i must complete before job a_j can be started) and also between individual operations of a job in a multi-operation model (e.g., operation $a_{j,x}$ must precede operation $a_{j,y}$). We concentrate here on precedence relations between jobs.

The expression $a_i \rightarrow a_j$ indicates that a_i must precede a_j, that is, the processing of job a_i must complete before the processing of job a_j begins. In this precedence relation, job a_i is the *predecessor* of a_j, and a_j is the *successor* of a_i. Precedence relationships are transitive; that is, $a_i \rightarrow a_j$ and $a_j \rightarrow a_k$ implies $a_i \rightarrow a_k$.

A *precedence graph* consists of *N* nodes, each representing a job, and a set of directed arcs representing direct precedence relationships. Key types of precedence graphs include *assembly trees*, *branching trees*, and *chains* (Fig. 2.1). It is not unusual for scheduling problems involving precedence relationships to be more difficult to solve optimally than "precedence-free" but otherwise identical scheduling problems.

Where jobs (or operations) are related by precedence to one another, they are known as *dependent jobs* (or *dependent operations*); otherwise, they are *independent jobs*. The majority of single-operation scheduling problems addressed in the literature assume jobs are independent.

To denote the processing times, ready times, due dates, and deadlines of jobs, the symbol *a* is replaced by the appropriate symbol for the job property. For example, job $a_{i,j}$ in a multi-operation model will have its processing time written as $p_{i,j}$, while the deadline of job $a_{i[j]}$ in a problem with class-based referencing will be denoted by $\bar{d}_{i[j]}$.

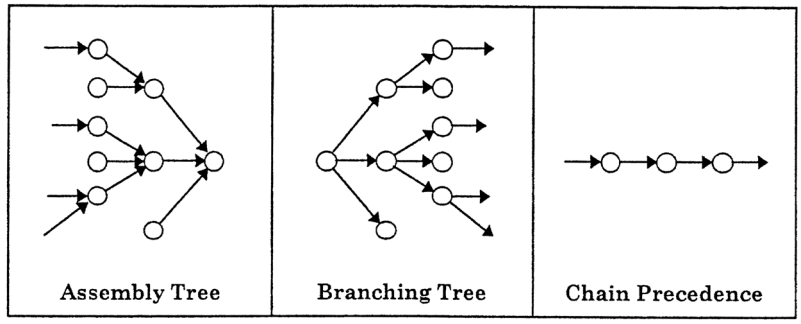


FIGURE 2.1 Key types of precedence graphs.

In some scheduling problems, the processing of jobs may be *preempted* (interrupted) and re-commenced at some later stage. In the *preempt-resume* model, no time penalty is paid for preempting a job, so that the total time devoted to any job is equal to its processing time. At the other extreme, all prior processing is lost in the *preempt-repeat* model. It is interesting to compare these models of preemption to the view that a job is comprised of a number of individual items of a particular product. The preempt-resume model can be viewed as an infinite-items-per-job case, while the preempt-repeat model can be compared to the one-item-per-job case.

Julien, Magazine, and Hall [46] introduce two further models of preemption. In the *preempt-setup* model, it is assumed that a setup of fixed duration must be carried out each time processing of a job is commenced or re-commenced. The preempt-setup model is a natural extension of the preempt-resume model, while the *preempt-startup* model extends the preempt-repeat model by assuming that a (prespecified) fraction $\alpha_j\,(0 \leq \alpha_j \leq 1)$ of the processing of job a_j (begun previously) must be repeated on re-commencement.

Models of Machines and Resources

Undertaking each operation within a job will require the allocation of a number of *resources* to the processing task. These resources can be "items" such as tools and machinery, as well as raw materials, energy, and labor. Some of these resources can be considered to be in limitless supply while others may be available in limited quantities and/or for limited periods. Additionally, a *cost* will be associated with the usage of each resource. Whereas some resource usage costs will either be able to be implied by (and optimized with) the schedule, or considered unimportant, others may have a considerable unit cost whose total value is dependent on the schedule followed. In the latter case, it may be desirable to optimize the use of the resource by directly incorporating the minimization of this cost into the scheduling objective.

It is common in mathematical scheduling problems to assume that the machines (processors) represent the only limited resources; more specifically, the capacity of each machine at each unit of time is the only limited resource. Usually, the machine will be able to undertake at most one operation at a time, exceptions being *batch processors* such as ovens, vehicles, or painting booths.

Where machines are not considered to be the only resources needing to be coordinated efficiently, the corresponding scheduling problems are usually quite difficult to solve. Some multi-resource problems consider the optimization of a "traditional" scheduling objective, such as the minimization of the mean completion time or makespan, subject to the constraint that one or more additional required resources is available in limited supply (e.g., Józefowska and Węglarz [45]). In other problems with additional resources, the objective is to minimize the consumption cost of an expensive resource. An example of such a problem can be found in Janiak and Kovalyov [44], where allocation of the expensive resource leads to increased processing rates and is required (over time in some quantities) in order for jobs to be completed prior to their deadlines. These problems are examples of what are commonly referred to as *resource-constrained problems*.

Single and Parallel Machines

For models of machines in scheduling problems, the *single-machine model* represents the simplest case. In a single machine problem, one machine (acting independently) is considered to be the only limited resource. It is assumed that a single machine has a fixed capacity which allows one task to be processed at any instant (i.e., a batch processor is not termed a single machine.)

Other simplifying assumptions are typically made in a single-machine problem, the most common being that the single machine processes at a constant rate and is available and fully functional at all times. This last assumption is not peculiar to single machine problems—it is an assumption made in the vast majority of scheduling problems.

The single machine problem is a *one-machine one-resource problem*, this being a special case of *one-machine problems*. One-machine problems are those that deal with any production facility consisting of one processor acting independently and having a one-operation-at-a-time processing capacity.

A *parallel machine problem* is a special case of a multi-machine problem. A group of machines are commonly described as parallel machines if they serve a single input queue of waiting jobs (Fig. 2.2). Typically, a parallel machine problem involves machines that individually are single machines.

There are three basic types of parallel machines modeled in scheduling problems: *identical* parallel machines, *uniform* or *proportional* parallel machines, and *unrelated* parallel machines. In a problem with identical parallel machines, all machines operate at the same speed (processing rate) and have the same processing capabilities. Uniform machines have the same processing capabilities but each has a different processing rate $\rho_m (\rho_m > 0, 1 \leq m \leq M)$, with the processing time of job j on machine m given by $p_{mj} = p_j/\rho_m$ for a given processing requirement p_j for job a_j.

Unrelated parallel machines represent the most complex of the three "standard" parallel machine types; such machines do not necessarily have identical processing capabilities, and the processing time of each job on machine m need not be related to either the processing times of other jobs on the same machine or to the processing time required on other machines. For each job j and machine m, a "job-dependent speed" ρ_{mj} is specified and used to provide processing time $p_{mj} = p_j/\rho_{mj}$. If a job cannot be processed on a certain machine, the use of a value of ρ_{mj} "near to zero" can prohibit the job from being scheduled on that machine, due to the extraordinarily large processing time assigned to it.

An interesting extension to the "standard" parallel machine models is the *parallel multi-purpose machine model*. Each job or operation of a job can be processed on a particular subset of the parallel

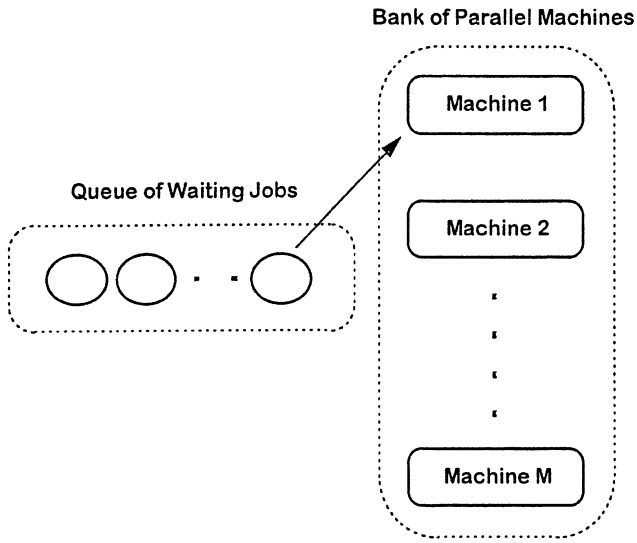


FIGURE 2.2 A representation of parallel machines and a single input queue.

machines, and the parallel machines are otherwise either identical or uniform. Only a small amount of scheduling research has been directed toward parallel multi-purpose machine models, although interested readers are referred to Brucker [11], for example.

Single and parallel machines can be seen as representing individual processing units, or workcenters, in a plant. Where the execution of entire workorders (i.e., all operations of a job) can be carried out on one workcenter, these machine models can incorporate almost all of the relevant machine characteristics.

Where jobs consist of multiple operations and require the attention of more than a single workcenter, it is on some occasions satisfactory to undertake detailed operational-level scheduling of each workcenter independently as a single or parallel machine. This might be accomplished, for example, by using release date, due date (or deadline), and job processing data based on a timetable generated by *material requirements planning* (*MRP*) or *production activity control* (*PAC*). With this input, the schedule at the machine can be optimized according to a relevant measure, subject to constraints such as deadlines that flow naturally from the timetable.

In other situations, a "bottleneck" machine (or bank of parallel machines) in a process can be identified and scheduled independently, with other processes around it "compensating to suit." However, the independent scheduling of one bottleneck workcenter may simply produce other bottlenecks in the system. There exist advanced and successful techniques for the application of single/parallel machine algorithms to multi-machine problems; although this is beyond the scope of this chapter, interested readers are referred to the *shifting bottleneck procedure* developed by Adams, Balas, and Zawack [1].

Other Multi-machine Models

The applicability of single or parallel machine models is limited. Other multi-machine, multi-operation models are required to appropriately model many facilities. There are three classical multi-machine models in addition to the parallel machine model that regularly appear in the scheduling literature, these being (see Fig. 2.3):

- *Flow shops*, where all work "flows" from one machine (workcenter) to the next; that is, jobs share a common operation (processing) order and hence a common routing through the shop. A flow shop model implies that chain precedence holds between the operations of each job.
- *Job shops*, where operations of a job must be carried out in a prespecified order (chain precedence) and on a prespecified machine (or parallel machines), so that individual job routings are fixed, but can vary between jobs.
- *Open shops*, where restrictions are not placed on the operation order (no precedence). Job routings are part of the decision process, but operation-machine assignments are predetermined.

Some multi-machine environments will be inadequately represented within the "classical" classification scheme of flow shops, open shops, and job shops. For the purposes of this chapter, however, there is no need to extend the classification.

Usual Assumptions in Scheduling Models

Although objective functions and setup time models have yet to be discussed, it is already clear that a scheduling model will potentially incorporate many features. When defining a particular problem, it would be a tedious process to address every possible feature. Thus, it is useful to provide a set of basic assumptions for a scheduling problem, and require exceptions to these assumptions to be stated when defining a problem.

In this chapter, the usual assumptions are as follows:

1. The starting point of the schedule is at time zero, and all assigned work is to be included in the schedule.
2. The only limited resource is machine capacity, which is one job per machine at any instant.
3. A single machine is to be scheduled, this machine being continuously available, having a capacity of one job per unit time, and operating at a constant unit speed.

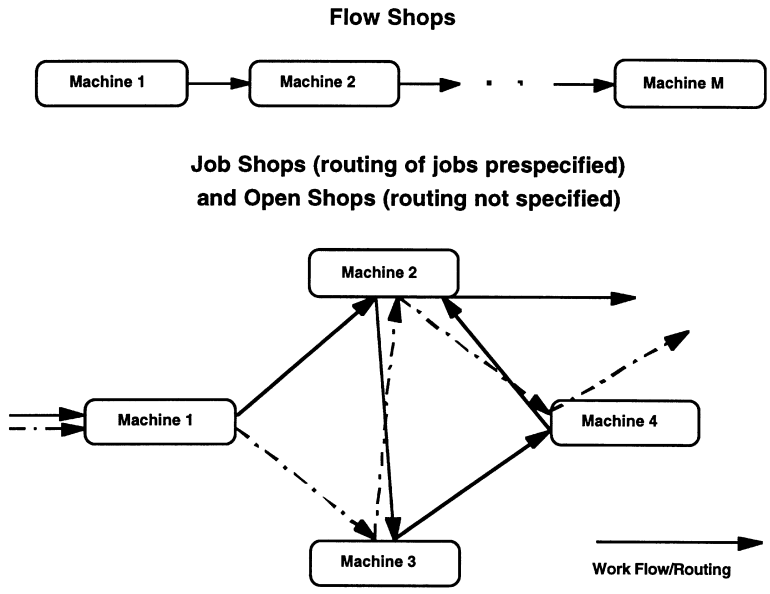


FIGURE 2.3 Flow shops, open shops, and job shops.

4. A set of N independent jobs, each comprised of a single operation and available at time zero, is to be scheduled to the machine.
5. All data is predetermined, and thus known deterministically, with numerical values expressed as arbitrary integers.
6. Preemption of jobs is not allowed.
7. Machine idle time is allowed.
8. No deadlines are imposed on job completion.
9. Setup times are negligible and need not be modeled.

In this chapter, most problems do incorporate setup times, so that Assumption 9 is very often replaced.

Objective Functions

Objective functions represent quantitative measures of schedule quality. In a scheduling problem, the value of the objective function is to be optimized (minimized or maximized, depending on the nature of the measure).

An "ideal" objective function will be able to represent the true economy of a particular schedule, as assessed according to all quantifiable costs and benefits. Constructing such a cost function presents a quite difficult proposition. Subsequent solution of the scheduling problem will typically be beyond the current capabilities of mathematical methods. For this reason, common (scheduling) objective functions represent a more restricted assessment of schedule quality. For example, the *makespan* objective is concerned with minimizing the total time required to complete a set of assigned jobs; this objective may minimize total cost under certain circumstances, yet in general can only be expected to address a restricted subset of the relevant economic considerations in the facility.

It is convenient to view each job as being assigned a cost based on its position in the schedule, f_j being the cost of job j, and $w_j f_j$ its weighted cost. The weight w may represent a size, importance, or economic coefficient. An objective function Z may be the sum of these costs (a *sum objective*, $Z = \Sigma_{j=1}^{N} w_j F_j$), the maximum of these costs (a *maxi objective*, $Z = \max_{j=1\ldots N}\{f_j\}$), or be some combination of maxi and sum terms. The cost f_j usually depends on the job completion time C_j. Where Z is non-decreasing with

each C_j, the objective function is known as a *regular objective function* and will exhibit characteristics that arise from the desire to complete jobs as soon as possible.

The makespan is usually given by the maximum completion time, $\max_{j=1\ldots N}\{C_j\}$, and is represented by the shorthand notation $C_{\max}$. Naturally, the *utilization* of a machine is inversely proportional to the makespan. Makespan problems can involve static or dynamic job arrivals.

The *flowtime* of a job is the total time spent by the job in the shop, from arrival to completion of processing. Scheduling job j to complete at time C_j yields a cost given by $f_j = C_j - r_j$ (symbol F_j often replacing f_j). When due dates are substituted for release dates to yield $f_j = C_j - d_j$, f_j represents the *lateness* of job j and is written as L_j.

The *weighted flowtime* F_W of a schedule is equal to the weighted sum of individual job flowtimes, $F_W = \Sigma_{j=1}^{N} w_j(C_j - r_j)$ for an N job problem. *Total flowtime* (or *unweighted flowtime*) F is a special case of F_W with all weights equal to one, while the terms *mean flowtime* $\bar{F}$ and *weighted mean flowtime* $\bar{F}_W$ are self-explanatory.

In a static problem, all ready times equal zero, and F_W is equivalent to the *weighted sum of completion times* $C_W = \Sigma w_j C_j$. For dynamic problems, $r_j \neq 0$ (in general) and $F_W \neq C_W$, yet the same sequence will optimize both measures. This is seen in Eq. (2.1). The $\sum_{j=1}^{N} w_j r_j$ term is constant for a given set of jobs, so that a sequence that is optimal for C_W is also optimal for F_W.

$$F_W = \sum_{j=1}^{N} w_j(C_j - r_j) = \sum_{j=1}^{N} w_j C_j - \sum_{j=1}^{N} w_j r_j \tag{2.1}$$

This *equivalence* leads to the terms "flowtime" and "sum of completion times" being synonymous, with flowtime proving to be a more convenient term.

This and further results provided by Conway, Maxwell, and Miller [20] can be adapted to show that a solution that is optimal for F_W (and C_W) is also an optimal solution for the *weighted lateness* L_W and the *weighted sum of waiting times* W_W objectives, where the waiting time of a job is defined as the time spent waiting for processing to commence, equal to $W_j = C_j - p_j - r_j$. The F_W, L_W, C_W, and W_W measures are thus termed *equivalent*.

F_W does not include due dates, and is not expected to be a good choice of objective in situations where the due-date performance of a schedule is critical. Through the equivalence of F_W and L_W, it is seen that L_W is also not a good objective in these circumstances. Lateness rewards early completion as much as it penalizes late completion, hence the net effect is one in which due dates are essentially irrelevant in the objective.

Nevertheless, where the schedule has most due dates set relatively early (the "earliness reward" is reduced) or most due dates set quite late (few jobs will be late), minimizing the lateness or flowtime becomes more appealing. The minimization of these measures is, however, more suited to cases where efficiency of work flow is the primary aim.

The *tardiness* of a job is given by $T_j = \max\{0, L_j\}$, and the weighted tardiness $T_W = \Sigma_{j=1}^{N} w_j T_j$ provides a far better measure of due-date performance than L_W. Minimizing maximum tardiness $T_{\max} = \max_{j=1\ldots N}\{T_j\}$ is another common objective, and a schedule that minimizes the maximum lateness $L_{\max}$ also minimizes the maximum tardiness. The reverse is not true; where all jobs can be completed by their due dates, $T_{\max} = 0$ (and many schedules may be optimal); whereas the minimum value of $L_{\max}$ may be negative and the maximum lateness objective acts in these cases to provide the greatest possible degree of earliness.

Similarly, when all jobs will be tardy (late), a schedule that minimizes L_W (and thus F_W) will also minimize T_W, because there now exists no "earliness reward." This result can be generalized: once a point is reached in a schedule where no remaining job can be completed on time, the remaining jobs can be scheduled according to minimization of T_W, L_W, or F_W. This is a useful result because minimization of the latter objectives is typically easier than minimization of T_W.

Tardiness measures the degree to which jobs are late. As suggested by its name, the *weighted number of late jobs objective* (N_W) aims to maximize the (weighted) number of jobs completed before their due date. A cost $f_j = U_j$ is assigned to each job according to:

$$U_j = \left\{ \begin{array}{l} 1 \text{ if } T_j > 0 \\ 0 \text{ otherwise} \end{array} \right\}$$

and the objective given as $N = \sum_{j=1}^{N} w_j U_j$.

All of the measures presented thus far are regular. *Earliness E* is an example of a non-regular objective function.

$$E_j = \max\{0, d_j - C_j\}$$
$$E_W = \sum_{j=1}^{N} w_j E_j$$

When scheduling with this objective, the impetus is to schedule jobs as late as possible. Thus, either individual or common job deadlines need to be specified, or inserted machine idle-time needs to be prohibited. Ordinarily, earliness would seem a curious measure, yet when it is combined with the tardiness measure to form the *earliness/tardiness* objective (abbreviated to *E/T*), a more useful scheduling aim is represented.

$$E/T = \sum_{j=1}^{N} w_{E_j} E_j + w_{T_j} T_j \tag{2.2}$$

The *E/T* objective is non-regular, and is relevant in situations where the thrust of this objective is to schedule a job as close to its due date (or *target date*) as possible. As such, a number of researchers reporting on *E/T* problems have seen an application of their analysis to "just-in-time" production systems. In Eq. (2.2), the *E/T* objective is shown to incorporate separate weights for the earliness and tardiness components. These will represent the relative importance or true relative cost of each job being early or late by one time unit. The assumption of a common due date for all jobs is a regularly modeled feature of many early-tardy problems.

Earliness/tardiness can be viewed as a single performance measure, or alternatively as an example of a *multiple criteria objective function* composed of two (single) criteria in a linear combination. This type of approach to multiple-criteria optimization is known as the *weighting of criteria approach* or *weighting method*. The use of "fixed" (predetermined) weights, as for the *E/T* objective, represents a special case of a more general method that dynamically modifies weights in order to generate a set of *non-dominated solutions*. The set of non-dominated solutions is a subset of feasible solutions; for any feasible solution that is dominated, there will exist a non-dominated solution for which all criteria are unchanged or improved and at least one criterion is strictly improved (for detailed discussion of this topic, the reader is referred to Goicoechea, Hansen, and Duckstein [36], for example).

Chen and Bulfin [17] observe that in addition to the weighting of criteria approach, there are two other approaches to multi-criteria problems relevant to scheduling problems. In the *efficient set generation method*, a set of non-dominated schedules is generated by an algorithm, and a decision-maker then chooses "the best schedule" according to an explicit trade-off of objectives. The *secondary criterion approach* or *goal approach* (with two criteria) designates one criterion as primary and the other as secondary. The best schedule according to the secondary criterion is selected from the set of alternative optimal schedules for the primary objective; in other words, the secondary criterion is optimized according to the constraint that the primary criterion takes its optimal value.

Essential Results: the SWPT and EDD Rules

The *shortest weighted processing time* (*SWPT*) and *earliest due date* (*EDD*) rules represent two basic yet powerful and generally applicable scheduling rules. The SWPT rule was shown by Smith [75] to provide optimal solutions to the single machine weighted flowtime problem (scheduling N statically available jobs to one machine to minimize the flowtime, without deadlines), while Jackson [43] established that the EDD rule solves the equally fundamental single machine maximum lateness problem. Each of these rules also optimally solve certain special cases of more advanced problems, and they remain relevant strategies in many others, including those with setup times.

The SWPT states that an optimal sequence for the single machine weighted flowtime problem has jobs appearing in non-decreasing order of weighted processing time (p/w). Where all job weights are equal, the SWPT rule reduces to the *shortest processing time* (*SPT*) *rule*. This celebrated result was established using an *adjacent pairwise interchange* argument. A schedule that does not satisfy the SWPT rule can be improved by taking any pair of consecutively scheduled (adjacent) jobs a_i and a_j which have $p_i/w_i > p_j/w_j$ and performing a pairwise interchange (i.e., swap their positions). This interchange must strictly improve the flowtime.

The SWPT rule provides a *dominance relationship* between jobs (a_i *dominates* a_j if $p_i/w_i \leq p_j/w_j$) and SWPT order is an *optimality condition* for this simple problem, as an optimal solution must have jobs sequenced in SWPT order. A set of optimality conditions is *necessary* if an optimal solution to a problem must satisfy them. A set of optimality conditions is *sufficient* if a schedule satisfying them is guaranteed to be an optimal solution. For the single machine weighted flowtime problem, SWPT is a sufficient condition of optimality because there is essentially only one sequence for which the SWPT optimality condition is satisfied throughout. When speaking of the special structure of a scheduling problem, we are often making reference to the optimality conditions that hold for this problem.

Although the SPT rule is a special case of the SWPT rule, Smith provides a separate proof for the SPT rule. This proof displays the important fact that the (unweighted) flowtime objective $F = \sum_{j=1}^{N} C_j$ for a single machine problem can be written as:

$$F = \sum_{k=1}^{N} (N - k + 1) p_{(k)} \tag{2.3}$$

This expression is constructed by considering that the processing time $p_{(k)}$ of the kth-sequenced job contributes to the completion times of $(N - k + 1)$ jobs; that is, it *delays* these jobs. Smith [75] observes the sum of the product of the two sequences $N - k + 1$ and $p_{(k)}$ in Eq. (2.3) will be minimized when they are monotonic in opposite senses. The sequence $N - k + 1$ is non-increasing, so that the sequence of processing times should be formed non-decreasing (i.e., $p_{(1)} \leq p_{(2)} \leq \ldots \leq p_{(N)}$). Expressing flowtime in the form of Eq. (2.3) or similar has assisted many researchers in deriving important results and developing algorithms.

Pairwise interchange is also utilized to establish the EDD rule. Any schedule that has two adjacent jobs a_i and a_j with $d_i > d_j$ can be improved by interchanging the positions of these two jobs. Therefore, the EDD rule states that to minimize maximum lateness on a single machine, sequence jobs in non-decreasing order of due dates.

The same rule also minimizes maximum tardiness, as $T_j = \max\{0, L_j\}$. As already noted, minimizing weighted lateness ($\Sigma w_j L_j$) is equivalent to minimizing weighted flowtime (SWPT rule). Neither the EDD nor SWPT rules is sufficient for minimizing weighted tardiness ($\Sigma w_j T_j$) on a single machine, although each has a part to play in algorithms for this problem.

Choice of an Objective Function

The minimization of makespan for a single machine problem with setup times is exactly equivalent to the minimization of total setup time, and thus the maximization of machine utilization. For multi-machine problems, the relationship between makespan, total setup time, and utilization is also strong, although

with makespan defined as the time by which all machines have finished processing, there does not exist an exact equivalence. In the industrial survey carried out by Panwalkar, Dudek, and Smith [65], the objective of minimizing total setup time was rated as the second-highest priority behind due-date performance.

The flowtime objective function is a useful measure of schedule quality when minimization of work-in-process inventory is desired, or customer satisfaction is assumed to decrease linearly with job completion time. In the latter case, average (weighted) customer satisfaction can be measured directly using average (weighted) flowtime. A primary disadvantage of flowtime-based measures is that the due-date performance of a schedule is not evaluated—a scheduling problem involving minimization of flowtime will tend to have optimal solutions that emphasize SWPT, which is not associated with due dates.

Lack of consideration of due dates in the flowtime objective is a particularly acute shortcoming when job weights are not modeled. When job weights are included, there exists a means of promoting a close-to-due job to an early position in the schedule, by imparting a high SWPT priority to the job. Although this type of approach does yield some control over due-date performance, the practicality of this approach is limited. The imposition of job deadlines can considerably improve the situation; however, if these (externally set) deadlines cannot be met, an alternative objective to flowtime needs to be considered. Use of an objective function combining flowtime with a due-date measure such as tardiness or weighted number of tardy jobs is one such alternative.

Once all remaining jobs (past a certain point in a schedule) are tardy, T_W and F_W will be minimized by the same sequence. Hence, flowtime becomes an increasingly relevant measure when scheduling a *busy period*—that is, when a facility is heavily loaded and many due dates are not achievable. However, minimization of flowtime is most suitable when the aim of reducing work-in-process inventory is assigned a high priority.

Conway et al. [20] provide proof that a strong and useful relationship exists between flowtime and inventory. For static problems, they provide the result that the ratio of the mean number of jobs in the shop $\bar{N}$ to the total number of jobs is equal to the ratio of the average flowtime to the maximum flowtime; that is,

$$\frac{\bar{N}}{N} = \frac{\bar{F}}{F_{\max}}$$

In this result, $\bar{N}$ is the average number of jobs taken over the interval $(0, F_{\max})$.

Hence, when $F_{\max}$ is constant for a given problem, minimization of unweighted flowtime leads directly to minimization of the number of jobs-in-process. There are many problems for which $F_{\max}$ varies between schedules however, particularly where setup times are involved. Dynamic arrivals complicate the issue further (Conway et al. [20] consider a dynamic case); nevertheless, flowtime remains a strong indicator of the work-in-process inventory, and the result generated by Conway et al. (for the static problem) can be generalized to the weighted case, giving:

$$\bar{N}_W(0, F_{\max}) = \frac{F_W}{F_{\max}}$$

where $\bar{N}_W$ is the *weighted average number of jobs* in the shop.

Tardiness is a relevant objective function when customers or downstream processes are sensitive to timely delivery, with job weights being able to reflect relative importance. When concerned with due dates, T_W is clearly a more appropriate objective function than L_W (as L_W is equivalent to F_W), yet the maxi criteria $T_{\max}$ and $L_{\max}$ are of approximately equal utility. Tardiness problems are typically more difficult to solve than corresponding flowtime problems.

While combined/multiple criteria objectives such as $T_W + F_W$ will evidently be superior performance measures in most situations, the solution of problems involving multiple objectives commonly presents far greater difficulty than the solution of problems with only one of the objectives.

Models of Setup Times

In an industrial machine scheduling environment, machine *setups* (or *changeovers*) are required whenever there is a need to change the process settings or configuration of a particular machine. This corresponds to a change in the *state of the machine*, and changes in machine state can be classified thus:

1. *Machine preparation*: setting a machine to a state in which it can undertake processing of jobs, from some initial shutdown state
2. *Machine switching*: changing the state of a machine, from that required to process one job (or class of job) to that required for processing another
3. *Machine stoppage*: the halting of a machine, removal of incomplete work, maintenance and/or repair of a machine, followed by a resetting the machine in order to continue processing after a planned or unplanned stoppage

While machine preparation and machine switching setups are looked at extensively in this chapter, setups due to machine stoppage are predominantly outside the scope of the analysis, as these are not ordinarily aspects of deterministic scheduling problems.

In addition to preparation and switching setups, *machine shutdowns* can also be modeled. These activities are performed in order to "shutdown" the machine, and these are considered in terms of the *additive changeovers model* of setup times.

Typically, the state of a machine is defined by various attributes such as the presence and positions of tooling mounted on it (e.g., dies, cutting tools, punches) the type of stock material available to it (e.g., width of strip metal, color of plastic or paint), and other factors such as machine temperature in the case of ovens. This represents a complex set of properties.

For the machine scheduling problems discussed in this chapter, it will be assumed that the state of a machine is sufficiently specified by (the processing characteristics and properties of) the job currently being processed on the machine. If the machine is considered to be "shutdown" initially, the initial state of the machine cannot be specified by a job; if necessary, this initial state can be represented by a "dummy job" a_0 assigned appropriate properties.

The assumption that the machine state is determined by (or well described by) the job currently being processed appears reasonable for many facilities. However, a limitation of this type of setup times model is that few appropriate mechanisms exist for controlling the state of a machine in order to best cater to future processing demands. This makes it unsuitable for certain problems.

For example, consider a manufacturing process in which each job requires the use of one tool, and a machine may hold two tools at any one time in a tool magazine, these currently being tools A and B. Job a_j is to be processed next, using an alternative tool C, and job a_{j+1} will follow it, using tool A. Evidently, tool B should be removed from the machine and replaced by tool C. Using the setup model proposed in this section, the two-tool magazine is not able to be well modeled (e.g., machine switching setups will be assumed to always remove the current tools and replace them), and optimization choices concerning the tools to be exchanged are not available. While scheduling problems such as that above are motivated by a number of significant industrial scheduling situations, their study is not within the scope of this chapter.

Sequence-Dependent Setup Times

The sequence-dependent setup times model is the most general model available when scheduling according to the assumption that the machine state is specified by the job currently being processed. For a one-machine application of this model, the setup time s_{ij} between (any) two jobs a_i and a_j is assigned a non-negative value, and no relationship is assumed between this setup duration and any others. There are $N(N + 1)$ setup times for an N job problem with initial setup times. These setup times can be recorded in a matrix.

When M machines are being scheduled, M setup-time matrices will be required and the duration of a setup between two arbitrarily chosen jobs a_i and a_j on machine m is written as s_{ijm}. Initial setup times, from

a bare machine to the first-scheduled job, can be written as s_{0jm}, where j is the index of the first-scheduled job and m is the machine index. Where $s_{0jm} = 0$ for all j and m, we have what is known as the "no initial setups" case.

As the time s_{ij} can generally be assigned any non-negative value, the sequence-dependent setup times model evidently does not explicitly incorporate knowledge regarding the true technological reasons behind setup durations. This keeps the model mathematically simple, yet the resulting lack of *structure* in problems incorporating a "general" sequence-dependent setup times model can make the analysis and solution of the problem very difficult—the existence of a known setup structure in a scheduling problem can be taken advantage of when formulating scheduling algorithms.

It is often the case, however, that certain assumptions are made regarding the setup times in problems with sequence-dependent setup times, a typical assumption being that the *triangle inequality* holds. A setup time s_{ij} is said to obey the triangle inequality if, given any two jobs a_i and a_j and any alternative job a_k, the inequality $s_{ij} \leq s_{ik} + s_{kj}$ is satisfied. In other words, setup times will obey the triangle inequality if it takes less time to directly prepare the machine for processing any job a_j than to prepare the machine for another job a_k followed by preparation to class a_j.

Problems incorporating setup times satisfying the triangle inequality are generally easier to solve than those with setup times that do not. It should be noted that when setup times satisfy the triangle inequality, this does not imply that setup times are *symmetric* (i.e., $s_{ij} = s_{ji}$ for all $i, j \in \{1, \ldots, N\}$), and vice versa. Generally, in fact, setup times are asymmetric.

The assumption that setup times obey the triangle inequality is reasonable in modeling many industrial scheduling environments. However, there are some scheduling problems based on practical situations which have setups that do not obey the triangle inequality. Sequence-dependent setup times not satisfying the triangle inequality typically arise in cases where jobs cause the state of the machine after their processing to be different from that which existed before their processing.

Our example is adapted from one provided by Pinedo [66]. This problem involves minimizing the time required to complete a given set of jobs that need to be "cooked" in an oven. Each job a_j is associated with two parameters: A_j being the initial oven temperature necessary for processing job a_j and B_j the oven temperature at the end of the processing cycle for job a_j. The setup time between two jobs a_i and a_j $(i \neq j)$ is given by the absolute difference between B_i and A_j.

Consider an instance of this problem where jobs belong to one of four products, the starting temperatures and finishing temperatures of products being given in Table 2.3. If the oven is assumed to have equal and uniform rates of heating and cooling, the setup times shown in the right-hand section of Table 2.3 can be calculated. A number of these setup times do not obey the triangle inequality; for example, $s_{41} = 250 > s_{42} + s_{21} = 150$. It can be noted, however, that although $s_{42} + s_{21} < s_{41}$, it is not possible to begin processing job a_1 less than $s_{41} = 250$ time units after finishing job a_4.

Pinedo provides an analysis of this problem, showing that an optimal solution for it may be gained in a time bounded by $O(N^2)$. This is a moderately surprising result which illustrates that although problems involving arbitrary sequence-dependent setup times are very difficult, practical situations may often exhibit *special structures* that can be exploited to produce efficient solution procedures.

The setup times prevalent in many other practical problems are likely to exhibit structure. Nevertheless, scheduling researchers might utilize a "general" sequence-dependent setup times model in preference to

TABLE 2.3 Temperatures and Setup Times for the Oven Example

			Changeover Times to Product			
Product	Starting Temperature	Finishing Temperature	From Product 1	From Product 2	From Product 3	From Product 4
1	100°	200°	100	50	300	250
2	250°	150°	50	100	150	100
3	250°	400°	50	100	150	100
4	300°	350°	100	150	100	50

more "structured" and "detailed" models of setup times. In part, this is because the assumption of a particular structure may be seen to sacrifice generality. Researchers who have developed and utilized detailed models of setup times (e.g., Charles-Owaba and Lambert [16], White and Wilson [77]) have demonstrated the potential of such models to provide insight into scheduling problems which can lead to effective solution procedures. In addition, Foo and Wager [29] utilize the sequence-dependent setup times model but recognize that setup times drawn from real industrial environments display patterns arising from the technological considerations in setups, and so assess their algorithms using representative data.

Inclusion of a "general" sequence-dependent setup times model in a scheduling problem usually results in a problem that is very difficult to solve optimally (or well), even for a single machine. For example, when there are N jobs, asymmetric sequence-dependent setup times, and the objective is to minimize the makespan, the single machine problem is equivalent to the *asymmetric traveling salesperson problem* (*ATSP*) with N cities (as observed by Smith [75]).

Both the ATSP and the (symmetric) traveling salesman problem (TSP) are well-known problems in operations research. The ATSP and TSP can be summarized thus: given a map of $N + 1$ cities $(0, 1, 2, \ldots, N)$, every city must be visited exactly once in a tour beginning and ending at city 0 (the "home" city), with the objective being to minimize the total cost of the tour.

If the cost of traveling from the home city to the first city in the tour is written as $d_{(0,1)}$, the cost of traveling between the $(k-1)$th and kth city in a tour is written as $d_{(k-1,k)}$, and the cost of traveling from the last city in the tour to the home city is written as $d_{(N,0)}$, the objective in a TSP or ATSP is given by Eq. (2.4):

$$\min\left\{\sum_{k=0}^{N-1} d_{(k,\,k+1)} + d_{(N,\,0)}\right\} \tag{2.4}$$

In the TSP, the cost $d_{(i,j)}$ (from city i to city j) is equal to the cost $d_{(j,i)}$ of traveling in the opposite direction (traveling costs in the symmetric TSP are typically euclidean distances). The ATSP has the $d_{ij} = d_{ij}$ restriction relaxed.

The objective function for the TSP/ATSP can be compared to the objective for a single machine makespan problem with sequence-dependent setup times

$$\min\left\{\sum_{j=0}^{N-1} s_{(j)(j+1)}\right\} + \sum_{j=1}^{N} p_{(j)} \tag{2.5}$$

where $s_{(j-1,j)}$ is the setup time from the job in the jth sequence position to the job in the $(j+1)$th sequence position, $s_{(0,1)}$ is the initial setup time, and $p_{(j)}$ is the processing time of job in the jth sequence position. The $\sum_{j=1}^{N} p_{(j)}$ term in Eq. (2.5) is a constant for a given instance of the makespan problem, so that when the cost of returning to the home city from any other city is set to zero, minimization of Eq. (2.4) is equivalent to minimization of Eq. (2.5) if traveling costs are replaced by setup times.

It can be noted that, in general, it is the ATSP that is equivalent to the single machine makespan problem with sequence-dependent setup times, due both to the costs of traveling to the home city being set to zero and the fact that $s_{ij} \neq s_{ji}$ in general for a machine scheduling problem.

However, when setup times are symmetric, the only difference between the TSP and the makespan problem is the setting of $d_{(N,0)} = 0$. When dealing with a *cyclic* scheduling problem, the makespan includes a final setup time corresponding to that required for preparing the machine for the next cycle. In terms of the TSP, this corresponds to a return to the initial city, so that a TSP can be used in this case.

Much effort has been expended over many years in the search for efficient solutions to the TSP and ATSP (for a guide, see Lawler, Lenstra, Rinnooy Kan, and Shmoys [52] or Reinelt [70]). These solution methods can be adopted directly when solving the single machine makespan problem with sequence-dependent

setup times. Currently, no exact (i.e., optimal) algorithm for which the running time is a polynomial function of the number of cities N has been devised. It is generally believed, but it has not been proven, that no such exact algorithm can exist (that is, the problem is *NP*-hard; see Algorithms and Complexity). Thus, essentially, for every exact algorithm for TSP, the running time grows exponentially with N, the problem size.

Because the single machine makespan problem with sequence-dependent setups and the TSP are mathematically equivalent, the single machine makespan problem with sequence-dependent setups is just as difficult as the TSP. Thus, in terms of the single machine makespan problem with sequence-dependent setup times, increasing the number of jobs in the problem ordinarily leads to an exponential increase in the time required to solve the problem optimally.

The Charles-Owaba and Lambert Model of Setup Times

Charles-Owaba and Lambert [16] propose an alternative to the sequence-dependent setup times model. In their model, it is assumed that a setup is comprised of one or more *machine setup tasks* (this assumption is adopted throughout this chapter). Each machine setup task is associated with a change in a certain process setting or tooling configuration of the machine. Likewise, each process setting or tooling configuration change is able to be associated with particular changes in the characteristics of the jobs to be processed.

In common with other models of setup times investigated in this chapter, the Charles-Owaba and Lambert model assumes that the machine state is specified by the job currently being processed. From this assumption it follows that the subset of machine setup tasks required in a particular setup is determined by considering the characteristics of the job currently being processed and the job being switched to.

The duration of a machine setup task can be fixed or variable. In the model utilized by Charles-Owaba and Lambert, durations are assumed fixed, these being the *standard times* of the machine setup tasks. A limitation of this assumption is that the degree of change in a job characteristic associated with a machine setup task does not influence the setup time. Due to this assumption, some facilities cannot be modeled sufficiently by the model proposed by Charles-Owaba and Lambert; for example, ovens or autoclaves (with variable temperature settings), or printing presses (with color as a process setting).

A machine setup task that could be required when setting up a machine for processing a particular job is termed a *performable task* for that job. The set of performable tasks for a job will be a subset of the available machine setup tasks, the complement of this subset being the set of *non-performable tasks* (for that job). Depending on the current and required configurations of the machine, some or all of the performable tasks for a job may be *eliminated*, that is, not necessary and consequently not performed. In the analysis undertaken by Charles-Owaba and Lambert, similarity in design characteristics is assumed to indicate similarity in processing characteristics.

Determination of setup times in the Charles-Owaba and Lambert model is based on inclusion or omission of the performable tasks for each job a_j (in a setup to that job) according to a measure of *similarity* in the characteristics of a_j and the job that precedes a_j in the schedule. When two adjacent jobs in a schedule (a_i and a_j) have highly similar processing characteristics, a large proportion of the performable tasks for job a_j might be able to be eliminated. Seeking to eliminate performable tasks by maximizing the similarity of adjacent jobs in a schedule clearly can lead to efficient schedules being produced.

We term *non-eliminable* a performable task for some job a_j that cannot be eliminated by similarity of this job to any other. When scheduling, the duration of a non-eliminable task for job a_j can be incorporated into the processing time p_j for the job and the non-eliminable task removed from the set of performable tasks for a_j. Commonly occurring examples of non-eliminable tasks include securing a workpiece in a holder or unloading a part from a machine. Some machine setup tasks may be non-eliminable for all jobs for which they are performable tasks.

As illustrated by Charles-Owaba and Lambert through a machining process example, the range of the values for each relevant design characteristic can be divided to create a series of categories (e.g., small, medium, and large) and each job classified for the purpose of determining similarity (jobs belonging to

the same category are considered similar). If a suitable design classification and coding scheme is already in use in the plant, this may be of assistance during this classification process.

Once jobs have been classified, similar jobs (according to any one characteristic) can be identified. At the time of scheduling, performable tasks are either eliminated or included in a setup based on the established similarity, and the total duration of a setup is calculated by summing the standard times for each machine setup task to be undertaken.

An implementation of the approach proposed by Charles-Owaba and Lambert requires:

1. Determination of the set of machine setup tasks, and their standard times
2. Identification of a set of relevant design characteristics (those that will influence setup times), and development of a suitable classification scheme for these characteristics
3. Establishment of the relationship between machine setup tasks and design characteristics (i.e., whether similarity in a design characteristic can lead to the elimination of a particular machine setup task)
4. Classification of the design features of each job

It would only be necessary to undertake (1), (2), and (3) in a planning stage. This is particularly important when considering facilities dealing with a variety of products, as alternative approaches requiring determination of each setup time on a job-by-job basis may become unworkable due to the time and effort required.

The Charles-Owaba and Lambert model is far more detailed, although less general, than the "common" sequence-dependent setup times model. Many of the technological considerations that contribute to the duration of setup times are captured by the model. This leads to scheduling problems incorporating this model displaying an identifiable (although complex) structure.

For the Charles-Owaba and Lambert model, the *set of machine setup tasks* can be denoted by $X = \{x_1, x_2, \ldots, x_U\}$ and the *set of design characteristics* denoted by $Y = \{y_1, y_2, \ldots, y_V\}$. The setup time s_{ij} from job a_i to job a_j is given by

$$s_{ij} = TE_{ij} = (t_1, t_2, \ldots, t_U) \cdot \begin{bmatrix} e_{ij1} \\ e_{ij2} \\ \vdots \\ e_{ijU} \end{bmatrix} = \sum_{u=1}^{U} t_u e_{iju}$$

where $e_{iju} = 1$ if machine setup task x_u is performable and not-eliminated, and $e_{iju} = 0$ if it is a non-performable task for job a_j or is eliminated by similarity in jobs a_i and a_j. T is a row vector of machine setup task durations, and E_{ij} is the *task elimination vector* whose uth element is e_{iju}.

The dissimilarity of two jobs is expressed by a vector D_{ij}. The vth element d_{ijv} of D_{ij} is zero if jobs a_i and a_j are similar in characteristic y_v ($1 \leq v \leq V$), or equal to one if the jobs are dissimilar in this characteristic. The construction of each D_{ij} can be greatly assisted if part classification and coding data are available, and it can be noted that $D_{ij} = D_{ji}$ for all i and j.

Information regarding the potential elimination of machine setup tasks is held in matrix Q. The element q_{uv} in row u of column v is assigned a value of one if dissimilarity in characteristic y_v necessitates machine setup task x_u being undertaken (or alternatively, if similarity in characteristic y_v can lead to elimination of task x_u). Element q_{uv} is set to zero if the necessity for x_u is not affected by similarity or dissimilarity in characteristic y_v.

It is therefore evident that the composition of Q is dependent only on the features of the facility being considered, and thus is independent of the requirements of the jobs being scheduled. In contrast, the value of elements of D_{ij} are dependent only on the similarity (dissimilarity) of the jobs.

The matrix Q, which applies to the problem as a whole, is then used to generate a set of matrices Q_j, one matrix for each job (or class of jobs). Rows in Q_j relating to performable tasks for a_j are as they

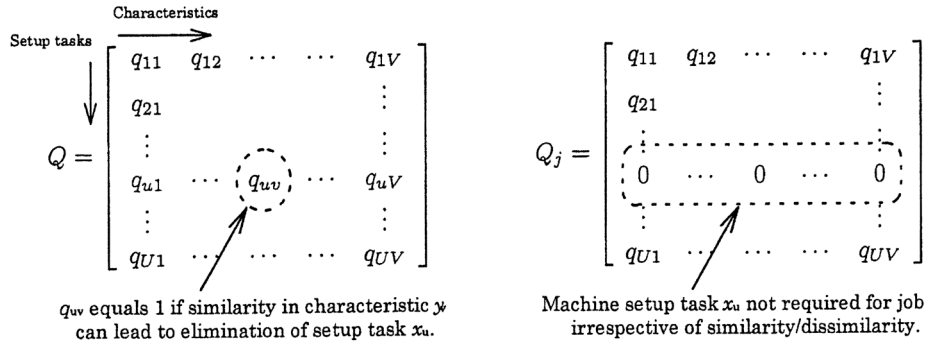


FIGURE 2.4 The Q and Q_j matrices.

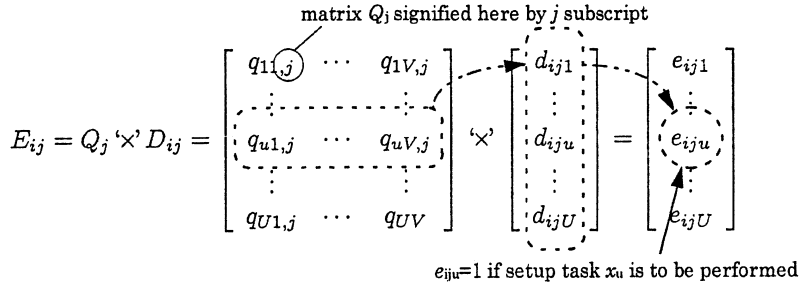


FIGURE 2.5 Generation of the E_{ij} vector by Boolean multiplication of the Q_j and D_{ij} matrices.

appear in Q. Rows in Q that relate to non-performable tasks for a_j have all elements set to zero when forming Q_j (Fig. 2.4).

Consider the setup from some job a_i to another job a_j. The matrix Q_j is generated from Q as in Fig. 2.4, and one can distinguish the elements of Q_j by writing $q_{uv,j}$ in place of q_{uv}. If $q_{uv,j} = 1$, dissimilarity in characteristic y_v leads directly to task x_u being performed when setting up for job a_j, regardless of similarity in other characteristics. Thus, a performable task is eliminated only if a_i and a_j are similar in every characteristic v for which $q_{uv,j} = 1$. If $q_{uv,j} = 0$, either the value of characteristic y_v is irrelevant to the requirement of setup task x_u, or setup task x_u is non-performable (unnecessary) for job a_j.

The task elimination vector E_{ij} for a setup from a_i to a_j is then generated by *Boolean multiplication* of Q_j and D_{ij} (represented by Q_j'×'D_{ij} in Fig. 2.5). The result of each row-by-column multiplication is either 0 if all element-by-element multiplications equal 0, or 1 otherwise (i.e., a boolean *OR* operation). Thus, a machine setup task x_u is carried out ($e_{iju} = 1$) unless all element-by-element multiplications equal zero ($q_{uv,j} \cdot d_{ijv} = 0$ for all characteristics $y_1, \ldots, y_V$).

Although $D_{ij} = D_{ji}$ for all i and j, it is not necessarily true that $s_{ij} = s_{ji}$ as the two jobs a_i and a_j may not share the same set of performable tasks (i.e., the Q_i and Q_j matrices may not be identical). Also, when a job is to be installed on an idle machine, it can be assumed that all performable tasks are carried out.

This description of the Charles-Owaba and Lambert setup times model has concentrated on a single machine. For a multi-machine problem, both T and Q may need to be specified for each machine.

Charles-Owaba and Lambert discuss procedures that can be followed to develop the matrix Q, which they term the *binary interaction matrix*. They propose that when considering each q_{uv}, a group of experienced engineers and shop-floor personnel may be asked whether similarity in characteristic y_v can lead to elimination of machine setup task x_u. In determining the set of machine setup tasks and relevant set of job characteristics a similar "team approach" could also provide an efficient and effective means for obtaining the necessary information.

It is also noted by Charles-Owaba and Lambert that the standard times t_u need not be obtained through observation of an existing process; they may be estimated using work measurement techniques or machine performance data. This allows determination of setup times for a system that is not yet in operation, allowing an appraisal of the performance of a planned or modified facility.

Class-Based Setup Time Models

A large proportion of the research into scheduling with setup times has been directed toward problems for which the set of jobs is partitioned into a number of mutually exclusive and exhaustive *classes*, each class containing one or more jobs. The corresponding models of setup times assume that setup times need to be (explicitly) considered when "switching" from the processing of jobs of one class to those of another. When two jobs of the same class are sequenced consecutively, it is assumed that all necessary setup tasks have been incorporated into job processing times; thus, setup times are ignored (zero) between two jobs of the same class.

Furthermore, for a single machine, it is assumed that the duration of a setup depends only on class membership; to determine the setup time, it is sufficient to specify only the classes of the jobs involved. For setup times on multiple-machine models, the machine must additionally be specified (we assume throughout that although setup times may be different, classes are identical at each machine being scheduled). Thus, it is evident that the formation of classes involves the grouping of jobs according to similarities in processing characteristics.

The assumptions of a class-based model of setup times are incorporated into Definition 1.

Definition 1

In a problem incorporating a class-based model of setup times, the setup time $s(a_x, a_y, m)$ between some jobs a_x (of class i) and another job a_y (of class k) on machine m is given by:

$$s(a_x, a_y, m) = \begin{cases} s_{ii,m} = 0 & \text{if } i = k \quad (a_x \text{ and } a_y \text{ belong to the same class}) \\ s_{ik,m} & \text{if } i \neq k \quad (a_x \text{ and } a_y \text{ belong to different classes}) \end{cases}$$

The value of each $s_{ik,m}$ $(1 \leq i, k \leq B,\ 1 \leq m \leq M)$ is an input parameter to the problem and is a non-negative integer.

It is also commonly assumed that the initial setup time from a "bare machine" to a job of class i is of duration $s_{0i,m}$, this time being dependent on class index i and machine index m only. We will also make this "uniform initial setup" assumption, although a class-based model of setup times can be developed without it (thus it does not appear in Definition 1).

As a general rule, if the ability to form classes of jobs exists, then such a grouping should be undertaken and a class-based setup times model used. This is because scheduling problems incorporating these setup time models are typically easier to solve.

Considering any class with two or more jobs, setup times between any pair of jobs a_x and a_y belonging to that class must satisfy the following constraints that arise from Definition 1. For simplicity, we refer here to a one-machine case.

$$s(a_x, a_y) = s(a_y, a_x) = 0 \tag{2.6}$$

$$s(a_x, a_z) = s(a_y, a_z) \quad \text{for all } a_z \in \mathbf{J}\backslash\{a_x, a_y\} \tag{2.7}$$

$$s(a_z, a_x) = s(a_z, a_y) \quad \text{for all } a_z \in \mathbf{J}\backslash\{a_x, a_y\} \tag{2.8}$$

Constraint (2.6) ensures that setup times within classes are zero. Constraints (2.7) and (2.8) ensure that the setup times to a class, and the setup time from a class, depend only on class membership and not

on properties of individual jobs. For setup times satisfying the "uniform initial setup" assumption, a fourth constraint is required

$$s(0, a_x) = s(0, a_y) \tag{2.9}$$

where $s(0, a_j)$ is the initial setup time to job a_j.

Formation of Classes

The process of class formation is not a common topic of interest in scheduling research. Nevertheless, it is a process that must be addressed during the application of a class-based setup times model to a production facility, as well as when assessing the practical applicability of a setup times model.

It is clear that classes can be formed when a number of jobs represent different orders for the same *product*, as jobs corresponding to the same product should satisfy constraints (2.6) to (2.8)—other than in "unusual" circumstances, any machine setup tasks required between jobs of the same product will be unavoidable (non-eliminable) and their duration may be aggregated with job processing times. Thus, we will assume that classes of jobs may be formed according to correspondence to identifiable products. It may also be possible to group different (though similar) products into "product types," leading to the formation of "larger" classes. This process is of interest here.

Formation of classes can be carried out using a one-stage process, where jobs are grouped directly into classes at each scheduling occasion. Alternatively, a two-stage process can be used. In a two-stage process, products are grouped into *product types* at a planning stage, and jobs are assigned to product types at each scheduling occasion.

If a two-stage process is used, the total number of classes can be no more than the total number of "product types." This is because (1) some product types may not be represented by a job, and (2) some rationalization of classes may be possible when one or more classes contain one job only. For one-job classes, many of the machine setup tasks will be unavoidable and, as a result, the job concerned may be able to be placed within an alternative class containing relatively dissimilar products. In comparison, some or all of these machine setup tasks may be eliminated completely if two jobs of this product are sequenced consecutively; in this case, the jobs are less likely to be grouped with dissimilar products. It can be noted that although we assume that setups can be successfully decomposed into machine setup tasks, we do not generally assume that machine setup tasks are assigned standard times.

From a purely mathematical viewpoint, the partitioning of a set of jobs into classes using a one-stage process should be relatively straightforward, given all setup times (determined by an analysis of the facility). An algorithm for forming classes according to constraints (2.6) to (2.8), based on known setup times, can be constructed without difficulty. Relevant setup times will typically need to be obtained from known product-to-product setup time data. Additionally, the durations of unavoidable machine setup tasks will need to be incorporated into job processing times before the jobs are partitioned; thus, the issue of one-job- or many-jobs-per product becomes relevant.

In practice, implementation of this process may be neither straightforward nor successful. Even for a single machine, N^2 setup times will need to be determined, a difficult and time-consuming process if direct observation or "intuitive" estimation techniques are used. Use of such methods to determine setup times can also result in setup times that display significant inaccuracies and random variations. Due to such errors, constraints (2.6) to (2.8) will have a much-decreased likelihood of being satisfied by a pair of jobs that correspond to similar products that "ideally" are able to be grouped. Allowing some tolerance in the satisfaction of the constraints may assist, yet on the whole use of this method alone appears generally insufficient.

Improvement in the setup time determination method may allow this constraint-based approach to class formation to be more successful. In cases where it is reasonable to assign standard times to machine setup task durations, the Charles-Owaba and Lambert model of setup times can provide such a method, for either one-machine or multi-machine problems. The modeling of setups as a set of discrete machine setup tasks, each with a standard time, will remove many of the causes of random variation in setup

times and provide a uniform and systematic structure for setup time estimation. After generation of task elimination vectors, and modification of these to remove non-eliminable tasks, classes may be formed directly using constraints (2.6) to (2.8).

The inability of the Charles-Owaba and Lambert to incorporate machine setup task durations that depend on the "degree" of dissimilarity of parts is a disadvantage of this model, which in turn is a shortcoming in the associated method of class formation. Considering such machine setup tasks separately and adding their durations to the modeled setup times represents a potential means of "stretching" the applicability of the approach. Extensions to the Charles-Owaba and Lambert model can also be envisaged.

The class formation approach based on the Charles-Owaba and Lambert setup times model may be unnecessarily complex when compared to alternatives. The attempt to directly satisfy constraints (2.6) to (2.8) throughout the class formation stage may also represent a disadvantage, as in some cases the number of classes might be significantly reduced by application of careful judgment in accepting small errors in constraint satisfaction. The approach also does not indicate with clarity the potential for modifying setup procedures in order to reduce setup times between particular products/classes.

Alternative approaches to class formation, based on a direct study of the processing characteristics of products, potentially addresses these issues. Unlike the constraint-based approaches outlined above, which can directly partition jobs into a set of classes, these alternative "product grouping" approaches form classes using a two-stage process. Facilities with a relatively stable range of products are more suited to product grouping methods than those dealing with one-of-a-kind-type production.

An *intuitive approach* to the product grouping strategy of class formation represents one potential option. Typically, the shortcomings of an intuitive approach will become apparent when applied to facilities either handling a large number of products or incorporating a number of machines. For example, inconsistencies and errors in the grouping of products can be expected to increase somewhat out of proportion to increases in the number of products or machines being handled. An intuitive method will not provide setup times, and additionally there is no guarantee that the classes will conform to the definition of a class-based model of setup times. These are disadvantages potentially shared by all product grouping approaches.

As discussed by Burbidge [14], the grouping of products into product types is a feature of the *production flow analysis* (*PFA*) approach to planning and implementation of *group technology* (*GT*) ideas for manufacturing. The relevant sub-technique of PFA is termed *tooling analysis* (*TA*) and the groups formed are known as *tooling families*.

Burbidge specifies that one aim of TA is to identify sets of products (tooling families) that can be produced "one after the other" using the same machine setup; thus, a tooling family corresponds to a product type as discussed above. This aim is clearly in agreement with Definition 1. A complementary aim of TA is to eliminate unnecessary variety in the tools used, in order to reduce the investment in tools. While not a scheduling objective, this is an important aspect of TA.

The tooling analysis technique is summarized by Burbidge as being composed of nine steps, the first seven of which are relevant here.

1. List parts with operations at each workcenter
2. Divide into sub-sets by type of material
3. Divide into sub-sets by form of material
4. Divide into sub-sets by size of material
5. Divide into sub-sets by material holding tooling
6. For each category found at (5), produce a tooling matrix
7. Find tooling families

In tooling analysis, tooling families are produced at each machine or set of largely identical machines, as described in Step 1. Difficulties may subsequently develop when multi-stage processes are dealt with, as the classes formed at each stage may not be identical. Whether this presents a problem in scheduling depends on the extent of class composition differences and the need for identical classes at each machine.

A pressing consideration is the issue of setup times between classes satisfying Definition 1. TA is not geared toward formation of tooling families whose setup times satisfy this definition, and we can only propose that, in favorable situations, any lack of satisfaction will not be significant.

We are not aware of publications that discuss the formation of scheduling classes from either practical or simulated data. White and Wilson [77] and Charles-Owaba and Lambert [16] bypass the class formation process, instead proposing a general sequence-dependent setup times approach to scheduling. White and Wilson use statistical treatment of observed setup times to establish a predictive equation for setup times, which can then be used in scheduling, as discussed below. In a similar way, Burns, Rajgopal, and Bidanda [15] use the existing design classification and coding scheme of a facility to generate (sequence-dependent) setup times and use TSP to solve the resulting single machine makespan problem. Karvonen and Holmstrom [47] describe a practical implementation of tooling analysis, although they do not discuss class-based setup time models nor scheduling algorithms. Given that class-based setup time models are not uncommon in scheduling literature, it appears that the issue of class formation (for class-based models of setup times) represents an important direction for future research.

Common Class-Based Models of Setup Times

There are three models of setup times that can be considered to be common in scheduling research. These are the sequence-dependent setup times model (discussed earlier), the *sequence-independent setup times model*, and the *major–minor setup times model*. Problems incorporating variations of these models, such as the *additive changeovers model* or the *unit setup times model* have also been studied. All the models listed above, except the sequence-dependent setup times model, are almost exclusively class-based models of setup times.

As noted, in a class-based scheduling problem, the scheduling of a machine setup is required only when *switching* between classes. A scheduling problem involving a class-based model of sequence-independent or sequence-dependent setup times can be termed a *class scheduling problem*. In the major–minor setup time model, a further division breaks each class into *sub-classes*; a major setup is required when switching between classes, and an additional minor setup is required when switching between sub-classes. We consider a problem incorporating a major–minor setup times model to be a class scheduling problem.

The model used by Charles-Owaba and Lambert belongs to less commonly occurring group that we will term *structured sequence-dependent models of setup times*. The approach taken by White and Wilson can also be said to yield structured sequence-dependent setup times. A structured sequence-dependent setup time model can provide an efficient and logical means of determining setup times, thus these "complicated" models are well suited for the purpose of providing setup times, even when they are replaced by a more "common" model in scheduling algorithms.

Sequence-Dependent Setup Times with Classes

The general sequence-dependent setup times model does not assume jobs to be partitioned into classes. For a problem with N jobs, if at least one class with two or more jobs exists, the class-based model may be distinguished from the general case. Furthermore, the problem is typically easier to solve when a division of the job set into classes is undertaken.

The class-based variant of the sequence-dependent setup times model can be mathematically described thus: the set of N jobs is partitioned into B mutually exclusive and exhaustive classes, each class contains one or more jobs, and every setup time satisfies Definition 1. The number of jobs assigned to class i ($1 \leq i \leq B$) will be denoted by N_i ($N_i \geq 1$); clearly $\sum_{i=1}^{B} N_i = N$. These definitions of B and N_i will be used throughout, for all class-based models of setup times.

Sequence-Independent Setups and Additive Changeovers

The *sequence-independent setup times model* is a special case of the sequence-dependent model with classes. The setup for a class k has duration s_k that does not depend on the class being switched from; that is, it is a *sequence-independent setup time*. Definition 2 states both the sequence-independent setup

times model and the *unit setup times model* for a single machine problem. In a multi-machine problem, a sequence-independent setup time to a given class can be specified for each machine.

Definition 2
In a single machine problem with sequence-independent setup times, the setup time from some job a_x (of class i) to another job a_y (of class k) is given by:

$$s(a_x, a_y) = \begin{cases} s_k & \text{if jobs } a_x \text{ and } a_y \text{ belong to different classes } (i \neq k) \\ 0 & \text{if jobs } a_x \text{ and } a_y \text{ belong to same class } (i = k) \end{cases}$$

where s_k is a non-negative integer and is termed the sequence-independent setup time to class k ($1 \leq i, k \leq B$). The unit setup times model is a special case of the sequence-independent setup times model with $s_k = 1$ for all classes k.

Compared to the sequence-dependent setup times model, which has quite general applicability as a model of setups in industrial scheduling environments, the sequence-independent setup times model is restrictive. In terms of machine setup tasks, accurate modeling of a facility using a sequence-independent setup times model requires that every performable and eliminable task associated with a class will be (1) carried out at each setup to the class, and (2) eliminated when two same-class jobs are sequenced consecutively. Thus, the degree of dissimilarity between classes must be great (so that no elimination of performable tasks can be undertaken when switching) and the degree of dissimilarity within classes must be negligible (only non-eliminable tasks are carried out when not switching).

The *additive changeovers model* is an extension of the sequence-independent setup times model, and was initially proposed by Sule [76]. The *changeover time* (switching time) is given by summing the sequence-independent *teardown time* for the outgoing class and the sequence-independent *setup time* for the class being switched to. The symbol t_i will be used to denote the teardown time of class i. It can be observed that setup times are not symmetric in either the additive changeovers model or the sequence-independent setup times model.

Definition 3
In a single machine problem with additive changeovers, the changeover time from some job a_x (of class i) to another job a_y (of class k) is given by:

$$s(a_x, a_y) = \begin{cases} t_i + s_k & \text{if jobs } a_x \text{ and } a_y \text{ belong to different classes } (i \neq k) \\ 0 & \text{if jobs } a_x \text{ and } a_y \text{ belong to the same class } (i = k) \end{cases}$$

where t_i is the (sequence-independent) teardown time from class i and s_k is the (sequence-independent) setup time to class k.

The additive changeovers model clearly has improved applicability compared to the sequence-independent setup times model, particularly in cases where changeovers comprise the removal of fixtures and tooling associated with an outgoing class (teardown) and the installation of same for an incoming class (setup). The sequence-independent setup times model cannot adequately model this situation unless all teardowns take a constant time for all classes, whereas the restriction is relaxed in the additive setups model to allow teardown times to depend on the class being removed.

A useful notation for representing jobs in a problem incorporating a class-based model of setup times is $a_{i[j]}$, where i is the index of the class the job belongs to ($1 \leq i \leq B$), and j is the index of this job within the class ($1 \leq j \leq N_i$). This method of referring to jobs, and the associated notation for job attributes such as processing time and weight, was introduced in Section 2.2.

A Comparison of Additive Changeovers and Sequence-Dependent Setup Times

The additive changeovers model can be seen to provide some degree of sequence dependency in changeover times. Nevertheless, most sequence-dependent class-based models cannot be represented by additive changeovers, even for problems with only two classes (consider the case $s_{01} = s_{12} = 0$, $s_{02} \geq 0$,

TABLE 2.4 Changeover Durations Resulting from Use of the Additive Changeovers Model

			Changeover Time to Class			
Class	Setup	Teardown	Initial Setup	From Class 1	From Class 2	From Class 3
1	3	4	3	•	4	8
2	7	1	7	11	•	12
3	5	5	5	9	6	•

$s_{21} > 0$ noted by Mason and Anderson [59]). On the positive side, changeovers may "appear" sequence dependent yet can be successfully modeled as additive changeovers. Table 2.4 gives the setup and teardown times for an instance of a problem with additive changeovers, and presents the resulting changeover times, which "look sequence dependent" despite being generated using an additive changeovers model.

A problem with additive changeovers will be easier to solve than a problem with sequence-dependent setup times; an additive changeovers model should be adopted if possible. Thus, there is significant benefit in investigating whether "apparently" sequence-dependent setup times actually follow an additive changeovers structure.

Such an investigation will be a component of a class-formation process, and it is therefore relevant at this point to adapt constraints (2.6) through (2.9) to the additive changeovers model (incorporating the "uniform initial setups" assumption). The resulting constraints are (2.10) to (2.13) which any two jobs $a_{i[x]}$ and $a_{i[y]}$ belonging to a class i ($1 \leq x,y \leq N_i$, $1 \leq i \leq B$) must satisfy with respect to all other jobs $a_{k[z]}$ belonging to alternative classes k ($1 \leq k \leq B$, $k \neq i$, $1 \leq z \leq N_k$).

$$s(a_{i[x]}, a_{i[y]}) = s(a_{i[y]}, a_{i[x]}) = 0 \tag{2.10}$$

$$s(a_{i[x]}, a_{k[z]}) = s(a_{i[y]}, a_{k[z]}) = t_i + s_k \tag{2.11}$$

$$s(a_{k[z]}, a_{i[x]}) = s(a_{k[z]}, a_{i[y]}) = t_k + s_i \tag{2.12}$$

$$s(0, a_{i[x]}) = s(0, a_{i[y]}) = s_i \tag{2.13}$$

An approach that specifically considers processing characteristics may be more successful in determining the applicability of an additive changeovers model, compared to a "testing" of established setup times against constraints (2.10) to (2.13); refer to the discussion regarding the formation of classes.

While the additive changeovers model is more complex than the sequence-independent setup times model, it is fortunate that this added complexity can be accommodated without an increase in difficulty for a number of scheduling problems. This is because these particular problems can be solved by a process that first transforms the problem into one with sequence-independent setup times alone and then solves this "easier" problem. This is discussed further in Section 2.3.

The Major–Minor Setup Times Model

In the sequence-independent, additive, and sequence-dependent setup times models, one level of classification is present. The major–minor setup times model adds a second level, the *sub-class*, which is a further division of the class (Fig. 2.6).

While it is less common than other models of setup times, the major–minor setup times model has received attention from a number of researchers. Typically, a sequence-independent major setup time s_i is required when switching to class i, to which a sequence independent minor setup time $s_{i(k)}$ is added whenever processing switches to sub-class $i(k)$ of class i. Classes are indexed from 1 through B and sub-classes of class i are indexed 1 through b_i. It is assumed that relationships between sub-classes of different classes have no influence on setup durations, so that a minor setup will also be carried out whenever a

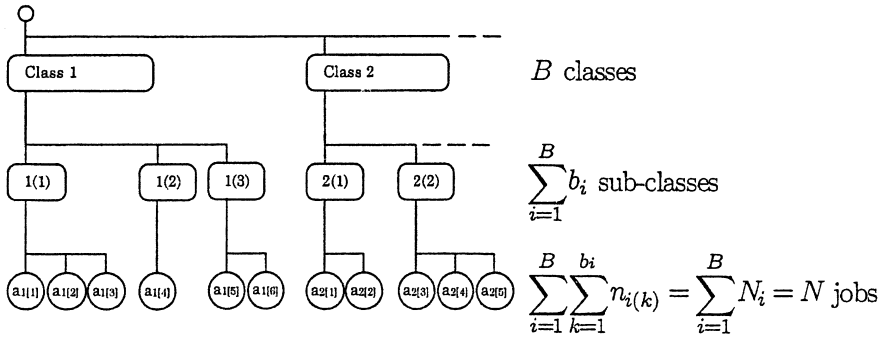


FIGURE 2.6 Classes and sub-classes in the major–minor setup times model.

major setup is necessary. Each sub-class $i(k)$ contains $n_{i(k)}$ jobs, so that $\sum_{i=1}^{b_i} n_{i(k)} = N_i$ and, evidently, $\sum_{i=1}^{B} b_i \leq N$ at each $n_{i(k)} \geq 1$.

Let the $b_{i[j]}$ be the index of the sub-class that job $a_{i[j]}$ belongs to ($1 \leq b_{i[j]} \leq b_i$). The total changeover time S scheduled between two jobs $a_{i'[j']}$ and $a_{i[j]}$ is given by:

$$S = s_{MJR} + s_{MNR}$$

where

$$s_{MJR} = \begin{cases} 0 & \text{if } i' = i \\ s_i & \text{otherwise} \end{cases} \qquad s_{MNR} = \begin{cases} 0 & \text{if } i' = i \text{ and } b_{i'[j']} = b_{i[j]} \\ s_{i(b_{i[j]})} & \text{otherwise} \end{cases}$$

The major–minor setup times model can be seen as a member of a broader class of setup time models that may be referred to as *hierarchial setup time models*. We can refer to a setup time model as being hierarchial if the set of jobs is successively partitioned into a series of levels. The lowest level (level K) consists of individual jobs only, the set of all jobs forms the highest level (level 0), while intermediate levels are comprised of sets of jobs that are mutually exclusive and exhaustive subsets of the sets in the level above. The major–minor setup times model is therefore seen to extend to a depth of $K = 2$.

A Classification of Class Scheduling Problems

Various terms have been utilized by researchers to describe and classify problems with class-based models of setup times. In this chapter we will adopt the following terms and definitions (see Fig. 2.7).

Class scheduling problems: problems within which jobs are partitioned into classes, with setup times being required when processing switches between processing jobs of one class and those of another. Problems with "hierarchical" setup time structures (e.g., major–minor setup time models) are included in this category. Setup times may follow any class-based setup times model.

Family scheduling problems: class scheduling problems with sequence-independent setup times or additive changeovers between classes (families) of jobs.

Group scheduling problems: family scheduling problems with the additional restriction that all jobs of a class must be sequenced together—the *group scheduling assumption*.

These definitions reflect the terminology commonly used in scheduling literature.

The terms "family scheduling problem" and "group scheduling problem" have their origins in the study of group technology (GT) environments. It is sometimes assumed that when scheduling machines that form part or all of a GT "group" or "cell," the ℓ products assigned to that group can be partitioned so that sequence-independent setup times between classes are the only setup times of considerable duration.

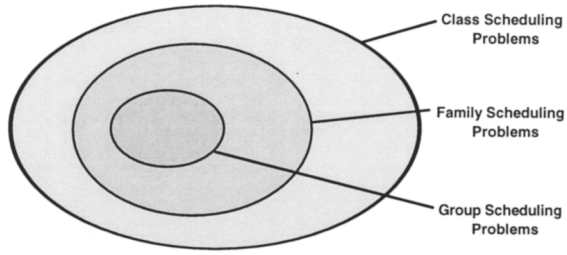


FIGURE 2.7 The division of the set of class scheduling problems into family and group scheduling problems.

These classes are termed *families* by a number of scheduling researchers, and appear to correspond to the *tooling families* described in Burbidge's production flow analysis.

As noted earlier, for setup times to be sequence independent, all performable tasks for any family (tooling family) must be carried out when switching to that family, while all machine setup tasks must be eliminated when not switching. Thus, while it is possible that setup times between families are sequence independent in a GT system, this requirement is restrictive and it is unlikely that sequence-independent setup times apply within GT systems in general.

Further, some researchers suggest that it is customary and generally beneficial to sequence all jobs of the same family (tooling family) together when scheduling within a GT environment. The resulting "group scheduling assumption" gives rise to the "group scheduling problems" term.

The common assumptions that will be made when discussing class scheduling problems are conveniently referred to as the *class scheduling assumptions*, and are:

1. All setup times satisfy the triangle inequality, but are generally not symmetric.
2. The first job in the schedule must be preceded by a setup for the class to which it belongs (i.e., an *initial setup* is required).
3. The machine may be left idle without the requirement of a teardown to remove the current class or a setup to reinstate it.
4. Idle time may be inserted.

In class scheduling problems, savings in non-productive time (i.e., changeover time) are possible by taking advantage of the group nature of jobs—sequencing jobs of the same class together. While the saving of setup time is important, the impetus to schedule jobs of the same class consecutively is tempered by other considerations, such as due-date performance for example. As observed by Potts and Van Wassenhove [68], class scheduling problems comprise decisions both about "batching" (whether or not to schedule similar jobs contiguously) and the scheduling of these "batches."

Scheduling and Lot-Sizing

Although jobs in scheduling problems can be considered to represent an "order" of one or more items of a particular product, they are typically treated in models and algorithms as being either wholly indivisible (non-preemptable) or continuously divisible (preemptable). Furthermore, in a multi-machine problem, a commonly adopted assumption is that a job can be processed by at most one machine at a time, although a job may include multiple items of the same product.

The potentially restrictive nature of this assumption is illustrated in Fig. 2.8, where a job comprising four items of a product is being processed in turn by three machines in a flow shop. Allowing the decomposition of jobs into component items has the potential to dramatically improve schedule performance. This will be true whether credit for completion is attributed only at the completion of the entire order (the usual case in scheduling problems) or reward is given for completion of each item (not common). The term *lot-streaming* has been used to describe the decomposition of jobs into smaller sub-jobs so that sub-jobs of the same job can be processed on different machines simultaneously.

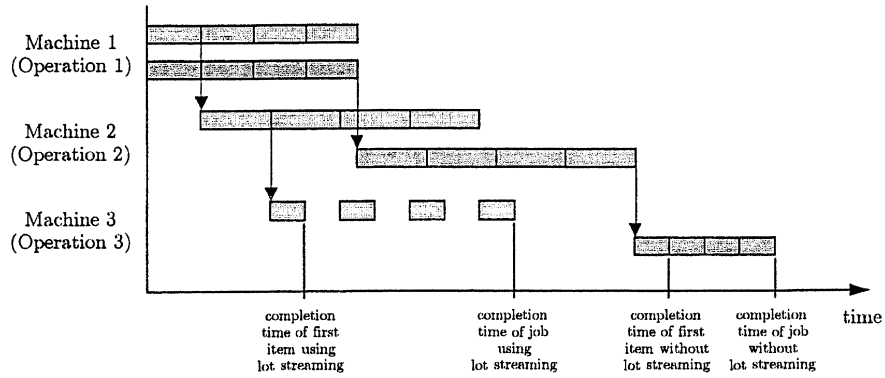


FIGURE 2.8 Effect of lot streaming on the completion time of each item within a job processed by a flow shop.

It is obvious that we can simply reduce the number of items per job to overcome restrictive scheduling assumptions. This corresponds to the division of an "order" into smaller units. The disadvantage of this approach is that the number of jobs requiring scheduling will increase in proportion. If established scheduling algorithms are used to solve the resulting instances of such problems, the execution times of the algorithms may well become unacceptable. This is because for all difficult scheduling problems (and even many simple ones), algorithm running times *must* grow exponentially with the number of jobs considered.

A more suitable strategy may be to explicitly model each job as being comprised of either a number of individual (and indivisible) items or a number of small groups of items. The symbol q_j ($q_j \geq 1$) can be used to represent the number of items or small groups in some job a_j; q_j becomes a property of a job. Scheduling algorithms can then be customized to deal with the problems as efficiently as possible.

Typically, problems concerned with the scheduling of many items of particular products are *lot-sizing problems*. Most lot-sizing problems are significantly different from most scheduling problems. In a lot-sizing problem, it is the product demands that appear as the primary input to the problem. Typically, a cost is associated with the formation of each lot of items, this tending to reduce the number of lots scheduled. The demand for products, in terms of due dates and quantities, provides an opposing force for the creation of multiple lots. Lots (the outputs of a lot-sizing problem) are very similar in definition to jobs in a scheduling problem (the inputs to a scheduling problem).

As noted by Potts and Van Wassenhove [68], scheduling literature nearly always assumes that lot-sizing decisions have already been taken. Lot-sizing literature does not usually consider sequencing issues; yet in industrial scheduling environments, these two types of decision are strongly interrelated. This interrelationship is clearly evident and an example of the implications of the interrelationship for a scheduling problem is provided by Fig. 2.8. Where setup times are considerable, the interrelationship is reinforced; these times are modeled directly in scheduling problems, while *setup costs* are commonly modeled in lot-sizing problems (avoidance of these costs contributing significantly to the tendency to form larger lots).

Literature on problems involving relevant elements from both scheduling and lot-sizing is surprisingly uncommon, particularly where setup times are explicitly considered in the schedule. This is unfortunate given the clear link between the features of such problems and practical scheduling needs and considerations. Potts and Van Wassenhove [68] have assisted in the study of such problems by presenting the "q_j-extension" (above) to the common job model in scheduling, providing a classification scheme for combined lot-sizing/scheduling, and surveying relevant literature on the topic.

Algorithms and Complexity

When speaking of a scheduling problem, we refer to a model (of a real-world scheduling environment) and an appropriate objective (relevant to that environment and the expectations made of it). What is missing is some (or all) of the data from the real-world problem. When all relevant data is incorporated

into both the model and the objective function, an *instance* of the scheduling problem is created. Although a new scheduling problem will be formulated relatively infrequently, new instances will be dealt with at each scheduling occasion.

Scheduling problems are solved by the specification of a method that can be used to solve some or all of the possible instances of the problem. In a mathematical approach to scheduling, this method will be an algorithm. Instances are solved by an implementation of that algorithm to produce a schedule. The development of algorithms involves an analysis of the structure of the problem and the adaptation of mathematical optimization methods to exploit this structure. It is algorithm development that commands the most interest in scheduling research.

A direct algorithmic solution to a given scheduling problem may not be possible, as the complexity of the problem may prohibit the successful application of available mathematical methods. Nevertheless, insights gained from the process of modeling, analysis, algorithm development, and study of solutions to simpler problems can often be valuable and lead to improved decision-making. It is unfortunate that most scheduling problems that accurately reflect real industrial environments are extremely difficult to solve completely (or even partially) using mathematical methods alone.

A primary requirement of a scheduling algorithm is that the schedules produced be *feasible* (i.e., all of the practical constraints are satisfied). For most scheduling problems analyzed in this chapter, it is not difficult to produce an algorithm that generates feasible solutions. The effectiveness of an algorithm is then measured by its performance, the basic measures being the quality of the schedules produced and the time taken to generate a schedule.

It is clear that there is considerable benefit in knowing in advance the likely performance of an algorithm, based on the properties of the problem being solved. If we know that the problem is difficult, then we might accept sub-optimal solutions if they are obtained in reasonable time, or accept protracted solution times to obtain optimal solutions. Similarly, if we fail to find an efficient algorithm, it would be helpful to know if our expectations were too great, based on some known problem difficulty. As discussed by Garey and Johnson in the introduction to their highly respected book [31], *the theory of computational complexity* can allow us to show that a problem is *inherently hard*; that is, no algorithm can be expected to produce optimal solutions in reasonable time (for all but the smallest of instances).

Algorithm Running Times

Consider an algorithm that operates on an ordered sequence of N jobs $\{a_1, a_2, \ldots, a_N\}$ searching for the adjacent pair of jobs with the greatest difference in processing times. The steps of this and any other algorithm can be decomposed into elementary operations, such as addition, subtraction, and comparison. The *running time* for the algorithm will be directly dependent on the total number of these elementary operations executed, and the total number of operations required will be a function of the input parameters of the problem (e.g., the number of jobs, the number of machines and the sum of processing times). Thus, an upper bound on the number of operations required becomes the *running time bound* or *time complexity function* of the algorithm, and is expressed in terms of the *problem size*.

That different types of operations will consume unequal amounts of time, and differing computer systems will execute operations at different speeds, is largely irrelevant—a running time bound expresses the relationship between the problem and the algorithm (it is a property of the algorithm). Additionally, a running time bound illustrates the problem parameters which contribute to the size of the problem. As shown in Fig. 2.9, the number of operations required by the example algorithm is $O(N)$; i.e., the number of operations is *in the order of* the number of jobs N and so is no greater than kN for some constant k. $O(N)$ becomes the running time bound for the algorithm, and the problem size is determined by the number of jobs N in this case.

The derivation of running time bounds is a routine process in the analysis of algorithms. Some algorithms have bounds that are polynomials in the problem parameters (e.g., $O(N + MN^4)$ as well as expressions such as $O(N \log N)$), these being *polynomial time algorithms*; while those algorithms without polynomial bounds are known as *exponential time algorithms* (exponential including time-complexity functions such as $(\log N)^N$ or N^N which grow more rapidly than, say, 2^N).

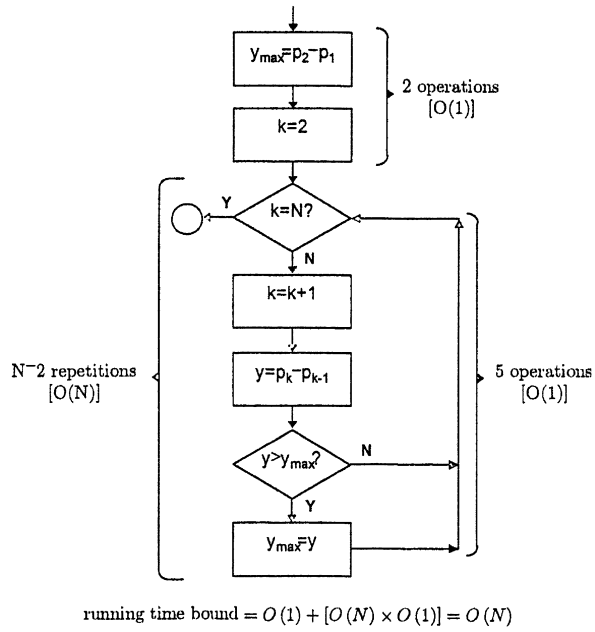


FIGURE 2.9 Flowchart and running time bound for the sequence searching algorithm.

TABLE 2.5 An Illustration of the Effect of Problem Size on the Running Times of Algorithms

Time Complexity Function	Problem Size n					
	10	20	30	40	50	60
n^3	0.001 seconds	0.008 seconds	0.027 seconds	0.064 seconds	0.125 seconds	2.6 seconds
n^5	0.1 seconds	3.2 seconds	24.3 seconds	1.7 minutes	5.2 minutes	13.0 minutes
2^n	0.001 seconds	1.0 second	17.9 minutes	12.7 days	35.7 years	366 centuries

Algorithms with polynomial running time bounds are of course able to be set to work on significantly larger problems, in comparison to exponential time algorithms. Table 2.5 is a condensed form of a table originally provided by Garey and Johnson [30], which emphatically illustrates the behavior of various time complexity functions. The running times are calculated from the given time complexity functions, each algorithm represented taking a millionth of a second to solve a problem of size $n = 1$.

The Theory of *NP*-Completeness

Running time bounds (or time complexity functions) are properties of algorithms, expressing the sensitivity of these algorithms to problem size. The theory of computational complexity tells us that there exist some problems for which (it is generally believed) there cannot be a polynomial time algorithm that is guaranteed to solve the problem optimally.

The set of *decision problems* for which polynomial time (optimal) algorithms exist is known as the set *P*. A decision problem involves deciding whether a statement is true or false; a decision problem in scheduling might be: "is there a feasible schedule that has a maximum lateness less than *K*," for example. Optimization problems (also known as *search problems*) are at least as hard as decision problems, as a decision problem can be solved by solution of an optimization problem (in the example, the optimal solution can be compared to the bound *K*).

A decision problem is said to be in set *NP* if a solution to the problem can be verified in polynomial time. For example, consider the decision problem: "can a given set of jobs to be sequenced to a single machine be completed in a makespan less than *K*." Given a sequence that is purported to have a makespan less than the value *K*, we can verify in polynomial time whether it indeed does, simply by constructing a schedule from the sequence; thus the decision problem is in *NP*. All problems in *P* are part of the set *NP*.

It is generally suspected that there are problems that are in *NP* but not in *P*; that is, that polynomial-time optimal algorithms cannot be found for some problems. That $P \neq NP$ is yet to be mathematically proven, however. To prove that $P = NP$, polynomial time algorithms would need to be found for the "hardest" decision problems in *NP*, these being the *NP-complete* problems. It turns out that to show that $P = NP$, we would require only one of the *NP*-complete problems to be shown to be in *P*, this result being central to *the theory of NP-completeness* and based on the concept of *reducing* one problem to another.

For two decision problems *A* and *B*, it is said that *B* reduces to $B(A \propto B)$ if the inputs *x* for *A* can be transformed, in polynomial time by some function *g*, into inputs $g(x)$ for *B* such that a "yes" answer is obtained for *A* if and only if is a "yes" answer for *B*. $A \propto B$ implies that *B* is at least as hard as *A*, *A* and *B* are equally hard if $A \propto B$ and $B \propto A$, and $A \propto B$ and $B \propto C$ implies $A \propto C$.

Cook [21] showed that every problem in *NP* is reducible to the *satisfiability problem* (a decision problem from Boolean logic). This implies that the satisfiability problem is the "hardest" problem in *NP*, and it is termed *NP-complete* in order to distinguish it from "easier" problems. If the satisfiability problem is solvable in polynomial time, so is every other problem in *NP* (hence, $P = NP$). On the other hand, if any problem in *NP* is truly unable to be solved in polynomial time, the satisfiability problem can only be solved in exponential time and $P \neq NP$.

Cook's theorem provides the result that $A \propto SAT$ for any problem $A \in NP$ (*SAT* being the satisfiability problem). Given a problem of interest $A \in NP$, if it is shown that $SAT \propto A$, problem *A* is therefore "equally hard" as the satisfiability problem and so is also *NP*-complete. Although the satisfiability problem was the first *NP*-complete problem, an *NP*-completeness proof for some problem *A* can utilize any *NP*-complete problem, as all *NP*-complete problems are equally hard (the transitivity of the '$\propto$' relation being important in allowing this). Essentially, a proof of *NP*-completeness provides the reduction from the known *NP*-complete problem to the problem of interest, and Cook's theorem provides the reduction in the opposite direction.

There now are many thousands of problems known to be *NP*-complete problems, with scheduling problems being well represented. If a decision problem is *NP*-complete, an optimization problem is said to be *NP-hard*, as it is at least as hard to solve as the decision problem.

While a problem may generally be *NP*-complete (or *NP*-hard), special cases may be polynomially solvable. For example, the traveling salesperson problem is *NP*-hard (see [52]); yet if all cities are arranged in straight line ("on the same road"), the problem is easily (and trivially) solved. Therefore, when reducing a known *NP*-complete problem *B* to a problem of interest *A*, it is imperative that every instance of *B* has a corresponding instance of *A*; otherwise, we may only be dealing with easily solved special cases of the *NP*-hard problem *B*.

A common in-practice method for establishing *NP*-completeness for a particular problem *A* involves the selection of a known *NP*-complete problem *B*, the provision of a transformation between inputs of *A* and the inputs of *B*, and proof that every instance of *B* has a (one-to-one) corresponding instance of *A*. It is not necessary that every instance of *A* has a corresponding instance in the known *NP*-complete problem *B*—the instances of *A* may be *restricted*, with this method of proof being known as *proof by restriction*.

Some restriction proofs are obtained simply, particularly for scheduling problems. For example, the problem of minimizing the makespan on a single machine with static job arrivals and sequence-dependent setup times is known to be *NP*-hard, by simple transformation from the TSP. It is obvious that the more general dynamic version of the problem is also *NP*-hard, as instances of this problem may be restricted to those with $r_j = 0$. Essentially, when an already *NP*-hard scheduling problem is extended to include

new properties (e.g., ready times), or more complex versions of existing elements (e.g., sequence-dependent setup times replacing sequence-independent setup times), the extended problem must also be *NP*-hard. For problems with setup times, this concept will be explored in detail shortly.

There are some *NP*-complete problems that have *psuedopolynomial-time algorithms*. The *partition* problem is one such example. Each item a_j in a set of items $A = \{a_1, a_2, \ldots, a_n\}$ has a "size" s_j, and the question in this decision problem is: "is there a subset A' of A such that $\Sigma_{j \in A'} s_j = \Sigma_{j \in A - A'} s_j$?." This problem can be solved in $O(n\Sigma_{j \in A} s_j)$ operations, so that if the sum of item sizes $\Sigma_{j \in A} s_j$ is restricted, the problem is solvable in polynomial time—hence, the algorithm is said to be psuedopolynomial. Many problems, for example, the TSP or the problem of minimizing the flowtime on a single machine with release dates, cannot have psuedopolynomial-time algorithms (unless $P = NP$) and thus are termed *strongly NP-complete* (decision problems) or *strongly NP-hard* (optimization problems).

The practical usefulness of the theory of *NP*-completeness is that, once a problem is shown to be *NP*-complete, approaches to solving the problem are restricted to (1) finding exponential time algorithms that are guaranteed to solve the problem optimally, or (2) accepting that finding optimal solutions for problems of practical size may not be possible in reasonable time, and developing algorithms which merely perform well.

Lower Bounds and Upper Bounds

The derivation of *lower bounds* and *upper bounds* on the optimal value of the objective function is often an important task when analyzing a scheduling problem. A lower bound is a value known to be less than or equal to the optimal value, and when studying problems we develop a procedure to generate a lower bound. This procedure can be termed a *lower bounding procedure*, although typically when speaking of lower bounds, it is the procedure that is being referred to. Upper bounds are values known to be higher than optimal, and are commonly simple to obtain using a known feasible solution. A bound is *tight* if it is "known" to be close to the optimal value.

As most scheduling problems are assigned objective functions that must be minimized (time, cost), a lower bound value will usually be an unattainable target, although a procedure that generates tight lower bounds may return the optimal value on occasion.

Bounds can be generated utilizing only the input data of the instance, or combining this data with partially constructed schedules. For example, a simple lower bounding procedure for a single machine makespan problem with setup times may add the processing times of all jobs to provide a lower bound. If an initial partial schedule has been specified, a lower bound on the makespan of a completed schedule beginning with this partial schedule can be obtained by adding the current makespan value (for scheduled jobs) to the sum of processing times of unscheduled jobs.

There are two primary uses of bounds in scheduling research. Lower bounds in particular can be used to gauge the performance of a non-optimal algorithm by establishing the approximate "position" of the optimal value, in the absence of a true optimal value. As a lower bound is always less than or equal to the optimal, and the algorithm will return a value equal to or greater than the optimal, the optimal is known to be "trapped" between the two values and small differences between lower bound and algorithm-returned value indicate good algorithm performance. An inability to obtain tight lower bounds limits the utility of this algorithm testing method.

Within algorithms, lower and upper bounds can also be used to guide the solution process, particularly in *branch-and-bound algorithms*, the primary features of which are discussed in the next section.

Optimal Algorithms

Some simple scheduling problems can be solved optimally using rules to construct a schedule, the SWPT rule for the single machine flowtime problem (mentioned earlier) being an example. However, in the great majority of cases, scheduling problems are *NP*-hard, and optimal algorithms for these problems are typically based on branch-and-bound techniques or *dynamic programs*. Algorithms of both types are presented in the sections that follow, while the principles and details of each have been well described in many texts.

At this point, we will outline the operation of branch-and-bound algorithms. In a branch-and-bound algorithm, an optimal solution is found using a *search tree*. A search tree consists of *nodes*, which contain partial or complete solutions. Directed arcs (*branches*) join nodes containing partial schedules to their successors. In a simple (yet typical) implementation, successor nodes (*child-nodes*) contain schedules produced by extending the partial schedules of their predecessor node (*parent-node*), and the *root node* at the top of the (inverted) tree contains an "empty" schedule. The algorithm begins at the root node and determines which child-nodes represent feasible partial solutions. This creates the initial node set; and at various points in the operation of the algorithm, child-nodes will be formed from existing parent-nodes in a *branching* process.

At each existing node, the current value of the objective function can be calculated, as can a lower bound on a complete solution beginning with the initial partial sequence at this node. Using prespecified criteria usually associated with lower bound values, nodes are chosen as parent-nodes; the chosen method of parent-node selection can be termed a *branching strategy*. Once one complete schedule is obtained, the objective function value at this node represents an upper bound on the optimal solution.

Any node containing a partial (or complete) solution of a greater objective function value than the upper bound can then be eliminated from further branching (considered *fathomed*), and the section of the tree that would otherwise lie beneath this node is *pruned*. By using lower bounds, pruning can also occur at any node with a lower bound value greater than the current upper bound; thus, use of a tight lower bound can dramatically reduce the number of nodes searched and hence the time required to reach the optimal solution.

In addition, pruning can be assisted by utilizing *optimality conditions*. Optimality conditions derived from an analysis of the scheduling problem allow partial or complete schedules to be deemed provably sub-optimal. In branch-and-bound algorithms, they represent a "test" that can be applied to partial schedules at nodes, eliminating those that do not have the potential to produce optimal solutions. It can be noted that most other solution methodologies also utilize optimality conditions in some manner.

In the worst case, a branch-and-bound algorithm will explore all possible solutions of the problem. However, when equipped with strong optimality conditions, tight lower bounds, and a suitable branching strategy, such algorithms have been shown to perform capably.

Heuristics and Approximation Algorithms

The *NP*-hardness of the vast majority of scheduling problems incorporating realistic models and assumptions often necessitates the use of algorithms that run in reasonable time but do not guarantee an optimal solution. The terms *heuristic* and *approximation algorithm* are commonly used to describe these algorithms, and can be used more-or-less synonymously.

Constructive heuristics, as the name suggests, build schedules one-job- or many-jobs-at-a-time, using heuristic rules arising from an analysis of the problem. At any point, only one partial schedule is considered. *Greedy heuristics* represent a common sub-type of a constructive heuristic. As the name suggests, the action of these heuristics is such that at any decision point, the option corresponding to the greatest immediate return is chosen, decision points being job completion times or ready times. *Dispatch heuristics* follow simple rules that choose the next job to be scheduled and thus are typically greedy, although queue-based scheduling disciplines such as first-in-first-out (FIFO) are also dispatch heuristics.

A pertinent example of a greedy heuristic is encountered when addressing the single machine makespan problem with sequence-dependent setup times. In this greedy heuristic, the next job scheduled is chosen so that the setup time from the currently last scheduled job to this next job is the minimum possible (so that if job a_i is the currently last scheduled job, the next scheduled job a_j is that which minimizes the resulting setup time s_{ij}).

Search heuristics are a varied class of heuristics that search for optimal solutions by iteratively modifying one or more existing solutions to the problem. For combinatorial scheduling problems, the modification process is such that, in time, improved schedules are found. It is possible that the schedule produced is globally optimal; however, no guarantee can be given, and all search heuristics exhibit an ability to become "trapped" by local, rather than global, optima.

We identify two broad types of search heuristics: *local search heuristics* and *evolutionary algorithms*. Local search heuristics generate a new schedule by undertaking a small modification to a current schedule. These modifications can be termed *moves*. Around a particular schedule there will exist a certain set of possible moves, that is, a *neighborhood* of moves. The size and scope of this neighborhood is determined by the range of moves made available within the heuristic specification. A simple and typical move may involve interchanging the positions of two individual jobs or groups of jobs in the schedule.

At each iteration of a local search heuristic, some or all of the moves in the neighborhood are assessed. One of these moves is accepted, via the use of a predefined acceptance criterion. This acceptance criterion may be either deterministic (e.g., in *descent heuristics*, *threshold accepting*, or *tabu search*) or probabilistic (e.g., in *simulated annealing*), and typically will assess moves according to the change in objective function value that will result. The acceptance criterion in a *descent heuristic* will only accept moves that do not worsen the objective function value. *Simulated annealing*, *tabu search*, and *threshold accepting* heuristics are equipped with acceptance criteria that can also allow moves which worsen the objective function value, and thus allow the heuristic to "escape" from local optima.

A local search heuristic maintains a single solution as current at any one time. The initial schedule (the current schedule at the beginning of the first iteration) is known as a *seed schedule*. *Multi-start* implementations of local search heuristics undertake the heuristic procedure more than once, using a different seed schedule on each occasion. By sequentially carrying out the search procedure on a number of different seed schedules, there is an increased probability that either the global optimum or a good local optimum will be found.

The larger the size of the neighborhood, the more time each iteration will be expected to take; yet while small neighborhoods may have a short time-per-iteration, many more iterations may be required to reach a reasonably good solution. When implementing local search heuristics, stopping criteria might need to be imposed, as otherwise a heuristic may fail to terminate. Typical stopping criteria involve limits on the total number of iterations, limits on the total execution time, or a minimum degree of improvement over a given number of iterations.

Evolutionary algorithms, or *genetic algorithms*, are search heuristics whose operation is modeled after natural evolutionary processes. A particular solution, or *individual*, is represented by a string, each element in a string determining some feature of the solution. Hence, the function of this string is analogous to that of genetic material in living organisms, and the string can be referred to as a *chromosome* comprised of a series of *genes*, each of which may take one of a range of allowed values (or *alleles*). We can also refer to this string as the *genotype* of the individual. A gene might correspond to a sequence position in a genetic algorithm for a machine scheduling problem, and the value of this gene will indicate the job to be sequenced at this position. Alternative *encodings* of schedules are commonplace, however. The schedule that results from interpretation of an individual's chromosome is termed the *phenotype* of this individual.

A *population* of individuals is maintained at any one time, the size of this population being predetermined. "Fit" individuals (good schedules) are allowed to *reproduce* to create new individuals, and "unfit" (poor) solutions are discarded. *Fitness* is determined through assessment of an individual's phenotype, usually by reference to its objective function value. In reproduction, approximately half the genotype of each parent is incorporated into the genotype of an offspring. Staleness in the genetic makeup of the population is able to be countered by allowing *mutation*, this being the alteration of one or more genes of an individual according to a randomized process.

A key assumption in evolutionary algorithms is that pieces of chromosomes that translate into good partial schedules will become more common within the population, due to the combined actions of fitness assessment and reproduction. This "natural selection" process steers the heuristic toward higher quality solutions. Although not initially developed for solution of combinatorial problems such as machine scheduling problems, application of genetic algorithms to these problems has been successful.

Beam search heuristics are search-tree-based procedures that, unlike branch-and-bound algorithms, are not designed to give a guarantee of optimality. Pruning in beam search heuristics is not confined to branches below nodes proved sub-optimal using bounding or optimality condition tests, but extends to

branches from nodes not considered "likely" to bear good/optimal solutions. Thus, efficiency of these heuristics is gained by restricting the number nodes explored.

Particular examples of the majority of these heuristic methods will be presented in the course of this chapter. It can be observed that, as heuristics are often quick-running, a set of heuristics may be implemented for a problem, with the best of the schedules generated by them accepted.

Worst-case error bounds can be associated with heuristics, these being analytically derived bounds on the worst-possible ratio of heuristic solution value to optimal solution value, considering all possible instances. Establishing worst-case bounds is typically an extremely difficult process for all but the simplest of heuristics. In addition, worst-case bounds may not convey useful information about the average or "usual" performance of a heuristic. For these reasons, the vast majority of heuristics reported in scheduling literature are not provided with worst-case error bounds, but rather with details of their typical performance as ascertained by computational experiment.

Assessing the Performance of an Algorithm

The in-practice performance of an algorithm can be experimentally measured by quantitative criteria such as objective function performance, running time, and data storage demands. Statistical methods are appropriate tools for carrying out a comparison between algorithms and/or lower bounds based on the above criteria. Often, the testing procedure will involve the generation of sets of representative problems using relevant probability distributions.

It is customary in scheduling research to choose many of the values in an instance (e.g., processing times) from uniform distributions. It is largely accepted that the instances result "test the bounds" of algorithm performance more rigorously, and that they represent more general cases. However, these instances may not be representative of instances arising from actual data, and practically-based instances can often be easier to solve (see, e.g., Amar and Gupta [4]). When attempting to establish a link between industrial scheduling environments and the scheduling problems that are motivated by them, the generation of sample instances can pose a dilemma that is compounded when relevant and previously reported testing has utilized a particular methodology.

The running time and objective function performance measures are evidently limited in their scope. The *utility* or *appropriateness* of a scheduling algorithm for tackling a practical problem will depend on factors such as:

- The ability to partially re-execute the algorithm to provide an updated schedule
- The degree to which necessary human intervention can be accommodated in the scheduling decisions
- The means by which secondary criteria (which are important yet not modeled) are able to be addressed and handled

These algorithm properties are more readily judged using qualitative measures and are difficult to assess without first selecting an application. Nevertheless, it is possible to make general observations; for example, Morton and Pentico state that combinatorial scheduling methods (tree-search based algorithms such as integer programming, branch-and-bound methods, and dynamic programming) have the advantage of providing optimal or near-optimal solutions, yet the disadvantage of not providing "any ordered intuition as to how to change the solution for emergency changes, slippages, and so on." By contrast, a simple dispatch heuristic will not be expected to perform as well as algorithms of the above type, but allows, for example, for a relatively simple recalculation of a job's priority (in the light of revised information) and subsequent modification of schedules.

A Classification Scheme for Scheduling Problems

To represent problems in a shorthand manner, we adopt a classification scheme based on that introduced by Graham, Lawler, Lenstra, and Rinnooy Kan [38]. Extensions of this scheme, to cater to more specialized problems such as those incorporating setup times, have been proposed by a number of authors. Some of these extensions are included in the scheme utilized in this chapter.

TABLE 2.6 Machine Environment Field Values

Field	Field Values	Meaning
α_1	o	One-machine problem
	P	Identical parallel machines
	Q	Uniform parallel machines
	R	Unrelated parallel machines
	O	Open shop
	F	Flow shop
	J	Job shop
α_2	o	If $\alpha_1 \neq$ o, arbitrary number of machines
	M	M machines involved in the problem

TABLE 2.7 Typical Specification of the Job Attributes Field

Field	Field Values	Meaning
β_1	*pmtn*	(Preempt-resume) preemption of jobs is allowed
	o	Preemption of jobs is not allowed
β_2	*prec*	General (arbitrary) precedence relationships hold
	tree	An assembly-tree or branching-tree precedence structure
	o	No precedence relationships specified
β_3	r_j	Jobs have prespecified release dates
	o	Jobs are available at all times (i.e., static availability)
β_4	$p_j = 1$	Each job has a unit processing time (single-operation models)
	$p_{ij} = 1$	Each operation has a unit processing time (multi-operation models)
	o	Each job/operation is assigned a integer and non-negative processing time

In the classification scheme, it is assumed that the problem includes N jobs that are to be scheduled on the machines involved, these jobs being comprised of one or more operations each. A scheduling problem is written in the form $\alpha|\beta|\gamma$, where α describes the *machine environment*, β the *job attributes*, and γ the *objective function*. Although useful, the scheme is most suited to representation of simpler scheduling problems, as the amount of detail able to be expressed by the $\alpha|\beta|\gamma$ format is limited. For complex problems, additional description may be necessary.

The machine environment represented by the field α is comprised of two sub-fields α_1 and α_2 ($\alpha = \alpha_1\alpha_2$); the values of these fields and their corresponding meanings provided are in Table 2.6.

Using this scheme, $1|\beta|\gamma$ is a single machine problem, $Q|\beta|\gamma$ is a problem with an unspecified number of uniform machines, and $F3|\beta|\gamma$ is a flow shop problem with three workstations in series. In the case of flow shop problems, for example, the lack of detail in the classification scheme can be inconvenient. Further specification of the machine environment may be necessary for complex problems.

The job characteristics field β has a number of sub-fields associated with it, each relating to a particular job property. The basic set of sub-fields (as given by Lawler, Lenstra, Shmoys, and Rinnooy Kan [53]) are shown in Table 2.7. It can be noted that the particular job properties indicating the empty values (i.e., o) default to those represented by the usual scheduling assumptions.

As examples, a single machine problem allowing preemption and with assembly-tree precedence structure will be represented by $1|pmtn, tree|\gamma$, while a job shop problem with equal (unit) operation times will be written as $J|p_{ij} = 1|\gamma$. Where necessary, other values of each sub-field will be used; for example, if all release dates are either equal to 0 or a common value r, this can be represented by assigning $r_j \in \{0, r\}$ to β_3 (e.g., $1|r_j \in \{0, r\}|\gamma$).

Researchers have used customized and additional sub-fields of β to allow representation of more advanced scheduling problems. Brucker [11], for example, uses β_5 to indicate job deadlines, β_6 for indicating a batching problem, and β_7 for information regarding setup times. Potts and Van Wassenhove [68] introduce a sub-field of β that allows various lot splitting and batching models to be efficiently represented. Table 2.8 provides extensions to the "standard" β field, which will be of use within this chapter.

TABLE 2.8 Extension of the Job Attributes Field

Field	Relates to	Examples (Entries and Meanings)	
β_5	Due dates d_j and deadlines $\bar{d}_j$	$d_j = d$	Common due date for all jobs
		$\bar{d}_j$	(Arbitrary) deadlines specified for each job
		o	Arbitrary due dates and no deadlines for due-date problems (e.g., minimising maximum lateness or weighted tardiness)
			No deadlines for 'flow-based problems' (e.g., problems with flowtime or makespan as the objective)
β_6	Setup times	s_{ij}	Sequence-dependent setup times with initial setups
		s_i	Sequence-independent setup times with $s_{oi} = s_i$ for all i
		s_+	Additive changeovers
		$s_i : s_i$	(Sequence-independent) major and minor setup times
		o	No setup times modeled

It can be observed that structured sequence-dependent setup times are not differentiated from general sequence-dependent setup times in the classification scheme. Importantly, the classification scheme does not contain information about the partitioning of the N jobs into classes and sub-classes.

A scheduling problem might not incorporate restrictions governing the number of classes (B) or the number of jobs in each class (N_i), other than those absolutely necessary ($1 \leq B \leq N$, $N_i \geq 1$ $\forall i$ and $\sum_{i=1}^{B} N_i = N$). If specific assumptions are made about the setup times model in a problem, then an additional sub-field of β (β_7) will be included in the classification to indicate this; for example, $1|s_{ij},B = 2|\gamma$ corresponds to a problem with sequence-dependent setup times and two classes. For problems incorporating the sequence-dependent setup times model, we will assume that jobs are not partitioned into classes if β_7 is empty (i.e., not used). A division of the job set into two or more scheduling classes will be assumed for all other models of setup times.

Specification of objective functions using the third field γ is comparatively simple. Each objective function is represented in "shorthand," and common values of γ for single-objective problems include $C_{\max}$ (makespan), $L_{\max}$ (maximum lateness), $\sum wC$ (weighted flowtime), $\sum wT$ (weighted tardiness), and $\sum wU$ (weighted number of late jobs).

In the classification scheme, we will differentiate between makespan problems where (1) the makespan is given by the maximum completion time $C_{\max} = \max_{j \in J}\{C_j\}$, and where (2) the makespan is defined as the total machine occupation time (including final teardown operations). The standard notation $\gamma = C_{\max}$ will be used for the former, while $\gamma = (C + t)_{\max}$ will be used for the latter.

Reducibility Among Scheduling Problems with Setup Times

Given N jobs partitioned into B classes, it is clear that a problem with sequence-independent setup times is easier to solve than a corresponding problem with additive changeovers, and that a problem with additive changeovers is in turn simpler than a problem with sequence-dependent setup times. Such relationships can be shown formally: any problem incorporating a simpler model of setup times can be written as a special case of a corresponding problem incorporating a more difficult model.

Fig. 2.10 shows a *reducibility* between problems with setup times. Directed arcs (arrows) in the top row ($x \rightarrow y$) indicate that model x is a special case of model y. As discussed by Lawler et al. [53], knowledge of such reducibility can be extremely useful in determining the complexity of scheduling problems.

Given $x \rightarrow y$ from Fig. 2.10, a problem X containing setup times model x reduces to a corresponding problem Y containing setup times model y (that is, $X \propto Y$). If problem X is *NP*-hard, so is problem Y; while if problem Y is polynomially solvable, so is problem X. The correctness of this particular argument requires that all other aspects of problems X and Y are identical. This is a restrictive condition, and far stronger results can be obtained by simultaneously considering reducibility graphs relating to other aspects of a problem, such as objective function and machine configuration (see, e.g., [38], [49], and [53]).

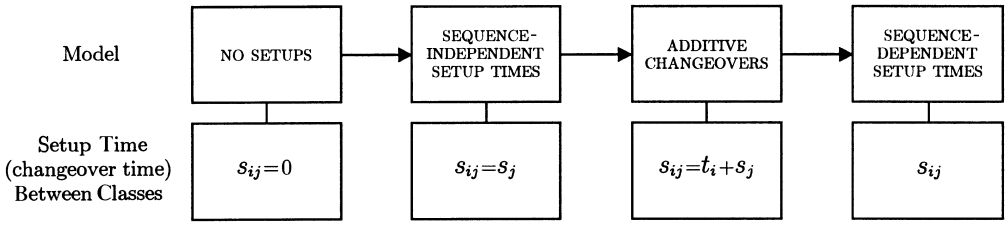


FIGURE 2.10 Reducibility between models of setup times.

Class	Setup	Jobs	
1	$s_1 = 2$	$p_{1[1]} = 2$	$r_{1[1]} = 0$
		$p_{1[2]} = 2$	$r_{1[2]} = 7$
2	$s_2 = 1$	$p_{2[1]} = 6$	$r_{2[1]} = 0$

FIGURE 2.11 Reducibility between models of setup times, including major–minor setup time models.

Problems with major setup times alone are special cases of those incorporating a major–minor model, where all minor setups have been set to zero. Thus, we can consider a general case where jobs are partitioned into both classes and sub-classes. Using the elementary result contained in Fig. 2.10, the reducibility of problems with setup times, including those with sub-classes, can be determined (Fig. 2.11). The reductions shown in Fig. 2.10 appear in the top row of Fig. 2.11, these being problems without minor setup times.

Problems with general sequence-dependent setup times between individual jobs correspond to the hardest problems of those where machine state is determined by the job being processed—hence the choice of the final node in Fig. 2.11.

The reductions shown in Fig. 2.11 are valid when dealing with either a general case of $1 < B < N$, or a special case where B takes a fixed value within this range. However, some caution is necessary; for example, if a problem is solvable in polynomial time only in special cases (such as when $B = 3$), this only implies that problems which reduce to it are in P when the same special case of B is applied. Naturally, if a special case of a problem is *NP*-hard, so is the general case of this problem.

It should be noted that one important relationship between problems with setup times is not displayed in Fig. 2.11. A problem Π with B classes, non-zero minor setup times, and no major setup times is equivalent to a problem Π' without minor setup times and with the number of major classes B' given by $\Sigma_{i=1}^{B} b_i$, (i.e., the number of minor classes in the original problem). Thus, $\alpha|o{:}s_i|\gamma$ is equivalent to $\alpha|s_i|\gamma$, for example. Although this may imply that setup time models without major setup times are redundant, use of such models in a problem may maintain the clearest connection between the scheduling model and the real-world problem; for example, if certain machines are prohibited from processing certain major classes.

The computational complexity of many problems with setup times can be established in a simple manner using Fig. 2.11 (and further results are possible when this reducibility graph is used in combination with similar graphs covering other problem classification fields). For example,

- The problem of minimizing the makespan on a single machine with additive major- and additive minor-setup times $(1|s_+:s_+|C_{\max})$ is known to be polynomially solvable. This implies that $1|s_i:s_+|C_{\max}, 1|s_+:s_i|C_{\max}$ and all *preceding* single machine makespan problems are also polynomially solvable.
- The problem of minimizing the total flowtime on two parallel machines with sequence-independent setup times $(P2|s_i|\Sigma C)$ is *NP*-hard (Cheng and Chen [18]). This implies that all parallel machine problems with setup times and the total flowtime (or weighted flowtime) objective are *NP*-hard.

Fig. 2.11 also illustrates the commonly utilized models of setup times in scheduling research. It is clear that research work into major–minor setup times problems with other than sequence-independent setup times may provide an important contribution. This will be particularly true if the problems are seen to represent common industrial cases.

2.3 Makespan and Flowtime on Single Machines

Section 2.2 discussed many of the fundamental elements of machine scheduling problems, presented a number of models of setup times, and outlined various solution methodologies that can be used to solve machine scheduling problems. This section investigates the development of algorithms for machine scheduling problems, concentrating on single machine problems where makespan or flowtime is the objective function.

Essentially, there exist two phases in the development of algorithms for machine scheduling problems. In the first phase, detailed mathematical analysis of the problem is undertaken to gain knowledge about the structure of the problem. In the second phase, one or more solution methodologies are adopted, and algorithms devised that take advantage of this structure.

Knowledge of the problem structure will influence decisions regarding the choice of algorithmic approach. For example, problems displaying little structure often cannot be successfully solved by either constructive heuristics or optimal algorithms, so that search heuristics might be most attractive. Even more importantly, exploitation of a problem's structure allows us to formulate more efficient algorithms.

Problems incorporating makespan or flowtime objectives reflect practical situations where the primary consideration is the efficiency of material flow. For a given set of jobs to be processed, each with a prespecified processing time, minimization of makespan is directly equivalent to minimization of the total time spent on activities other than job processing, that is, *non-productive time*. For the problems to be addressed, this non-productive time is due to setup times between jobs; and for dynamic problems, inserted idle time as well.

When minimizing the makespan, it is clear that a good scheduling tactic is to reduce the total number of setup operations carried out, and when possible, to minimize the durations of each of these setups. In cases where a class-based setup time model is an appropriate model for a problem, this strategy translates into the rule that all jobs belonging to a particular class should be scheduled consecutively, all jobs belonging to a particular sub-class should be scheduled consecutively, etc.

When minimizing flowtime in a class scheduling problem, different strategies to that utilized for minimization of makespan are often called for. In particular, significant savings in flowtime can be achieved by scheduling the jobs of a class in two or more *batches* (or *runs*). An initial appreciation of this result can be gained by considering that the flowtime objective favors the scheduling of the shortest jobs earliest in the schedule, so that where a class contains both "short" and "long" jobs, it may be beneficial to carry out the processing of shorter jobs early on, and return later to the processing of longer jobs. This class-splitting approach clearly represents a different strategy to that which may be suitable for a makespan problem.

It is not typical for deadlines to be imposed on the completion of jobs when makespan or flowtime objectives are minimized in a mathematical scheduling problem. Therefore, at least as far as individual jobs are concerned, the due-date performance of a schedule is largely ignored when a makespan or flowtime objective is utilized. This limits the practical application of scheduling models incorporating

TABLE 2.9 Computational Complexity of Single Machine Flowtime Problems

Polynomially Solvable	Open	*NP*-Hard
$1\|\|\Sigma wC$ ([75])		
$1\|\bar{d}_j\|\Sigma C$ ([75])		$1\|\bar{d}_j\|\Sigma wC$ ([55])
$1\|tree\|\Sigma wC$ ([42],[2],[74])		$1\|prec\|\Sigma C$ ([51],[54])
		$1\|r_j\|\Sigma C$ ([55],[71])
$1\|pmtn, r_j\|\Sigma C$ ([5])		$1\|pmtn, r_j\|\Sigma wC$ ([48])
	$1\|s_i\|\Sigma wC$	$1\|S_{ij}\|\Sigma C$ ([71])

TABLE 2.10 Computational Complexity of Single Machine Makespan Problems

Polynomially Solvable	*NP*-Hard
$1\|prec\|C_{max}$ ([50])	
$1\|r_j\|C_{max}$	
$1\|prec, \bar{d}_j\|C_{max}$ ([50])	
$1\|r_j, \bar{d}_j, pmtn\|C_{max}$ ([10])	$1\|r_j, \bar{d}_j\|C_{max}$ ([30])
$1\|r_j, \bar{d}_j, pmtn, prec\|C_{max}$ ([9])	
$1\|s_+\|C_{max}$	$1\|s_{ij}\|C_{max}$ (TSP)
	$1\|s_i, \bar{d}_j\|C_{max}$ ([12])

these objectives to situations where maximization of throughput is paramount, where most jobs are either already late or can be completed within time, or where job due dates are not of particular concern. It can be noted, however, that the assignment of priorities to jobs in a flowtime problem (by use of job weights) can assist in reducing the tardiness of jobs in a schedule.

The study of problems that do not explicitly consider setup times often provides a basis for an investigation of the structure of more general problems, which include setup times, and also provides useful insight into possible strategies for solving them. The (static) problem scheduling N independent jobs to a single machine in order to minimize the flowtime ($1\|\Sigma wC$) is a particular example of this. The SWPT rule, optimal for the $1\|\Sigma wC$ problem, is a very useful rule in any flowtime problem.

Although $1\|\Sigma wC$ is polynomially solvable, there are many single machine flowtime problems that are not. Table 2.9 summarizes the computational complexity of a number of "common" single machine flowtime problems.

From Table 2.9, it is seen that the imposition of setups, deadlines, general precedence relationships, or release times can make difficult the solution of an otherwise easy single machine weighted flowtime problem. Special and restricted cases of some of the above *NP*-hard problems are solvable in polynomial time; for example, the $1|s_{ij}|\Sigma wC$ problem can be solved in $O(B^2N^B)$ time, so that when the number of classes (B) is fixed, this complexity function is polynomial in the number of jobs (N).

Turning to single machine makespan problems (minimizing C_{max}), the computational-complexity picture is improved moderately. Table 2.10 outlines the computational complexity of some common makespan problems.

We will expand on some of the results summarized in Table 2.10. Lawler [50] provides an algorithm that enables $1|prec|f_{max}$ to be solved in $O(N^2)$ time, where f_{max} is any maxi form of a regular scheduling criterion (e.g., C_{max}, L_{max}, or F_{max}). Solving the $1|r_j|C_{max}$ problem merely requires the processing of any available job to be undertaken at any point in time — one such schedule sequences the jobs in non-decreasing release date order (i.e., earliest release date order). Similarly, the $1|\bar{d}_j|C_{max}$ problem is solved by ordering the jobs in non-decreasing order of deadlines (EDD order). A $1|prec, \bar{d}_j|C_{max}$ problem can be solved optimally using Lawler's algorithm, which will provide a feasible solution if one exists and will not involve the insertion of any idle time, and hence will give the optimal makespan of $\Sigma_{j=1}^{N} p_j$.

Where both deadlines and release dates are imposed ($1|r_j, \bar{d}_j|C_{max}$), minimizing makespan becomes an *NP*-hard problem as noted by Garey and Johnson [30]. Nevertheless, allowing preemption yields an efficiently solvable problem even with precedence constraints [9]. Optimal schedules to static single

machine makespan problems with setups can be found efficiently as long as the setup times are not sequence-dependent and deadlines are not imposed. Computational complexity results developed by Bruno and Downey [12] can be used to show that the $1|s_i, d_j|C_{\max}$ problem is *NP*-hard, while the relationship between $1|s_{ij}|C_{\max}$ and the traveling salesperson problem (TSP) has already been explored.

Preemption and Idle Time

The assumption of static job arrivals is restrictive. Thus, the introduction of *ready times* (dynamic job arrivals) is an important extension to static scheduling problems. However, dynamic job arrivals usually bring increased difficulty to scheduling.

For problems with static job availability and regular objective functions, it is clear that there is no advantage in considering schedules containing either preemption or idle time (see Conway, Maxwell and Miller [20]). In contrast, either machine idle time or job preemption, or both, will be present (if allowed) in good schedules for many problems with dynamic arrivals or non-regular objectives. It should be appreciated that a job ready time corresponds to the earliest time at which processing of a job may take place. The setup time for the job may be carried out prior to this time.

There are two fundamental reasons why idle time may be present in "good" schedules for dynamic problems. There is no choice but to schedule idle time when all available jobs have been processed and the machine must wait for further jobs to arrive. This type of idle time will appear in problems with and without preemption. Idle time can also be inserted to keep the machine unoccupied as it waits for an important job to arrive, if this is a preferred alternative to setting the machine to work on a less-important (yet available) job. Such *inserted idle time* can also be scheduled when jobs can be penalized for early completion (earliness penalties).

Schedules with inserted idle time are sub-optimal for problems with preempt-resume preemption, regular objectives, and no setup times. This observation holds whether or not ready times are known in advance [5]. Julien, Magazine and Hall [46] consider a case where setups are able to be interrupted and no information is known about jobs until they arrive at the system (i.e., there is no *look-ahead capability* when scheduling). A sequence-independent setup time of duration s_j is required prior to processing any part of job a_j, that is, before starting the job, and when resuming. This leads to a *preempt-setup* model of preemption. As there is no look-ahead capability, it is not reasonable to undertake a setup prior to the arrival of a job (at time r_j), as the arrival time and properties of the job are unknown. In this *dispatching environment*, where scheduling decisions are made at job completion times and at arrival times, schedules without inserted idle time are *on-line optimal* for regular objectives; a schedule being on-line optimal if it is optimal considering all "known" jobs. Julien et al. show that the *shortest weighted effective processing time (SWEPT)* rule is on-line optimal for the weighted flowtime problem. This rule states that the job with the least value of remaining processing time divided by weight should be scheduled next.

When information is known about the arrival times and processing requirements of dynamically arriving jobs, optimal schedules for problems with setup times might incorporate idle time even if preemption is allowed, as shown by the following example. The instance I being solved is of a single machine flowtime problem with sequence-independent setup times, and is presented in Table 2.11. Both preemptive and non-preemptive schedules will be considered.

It can be shown that there exists an optimal schedule for this instance within which job $a_{1[1]}$ precedes job $a_{1[2]}$; thus we need only consider schedules that have these jobs appearing in this order. Figure 2.12 illustrates a number of feasible schedules for this instance. The schedule with flowtime $F = 29$ is optimal for both the preemptive and non-preemptive problems, this schedule containing inserted idle time.

TABLE 2.11 Instance I, Which is to be Solved as a Single Machine Flowtime Problem

Class	Setup	Jobs	
1	$s_1=2$	$p_{1[1]}=2$	$r_{1[1]}=0$
		$p_{1[2]}=2$	$r_{1[2]}=7$
2	$s_2=1$	$p_{2[1]}=2$	$r_{2[1]}=0$

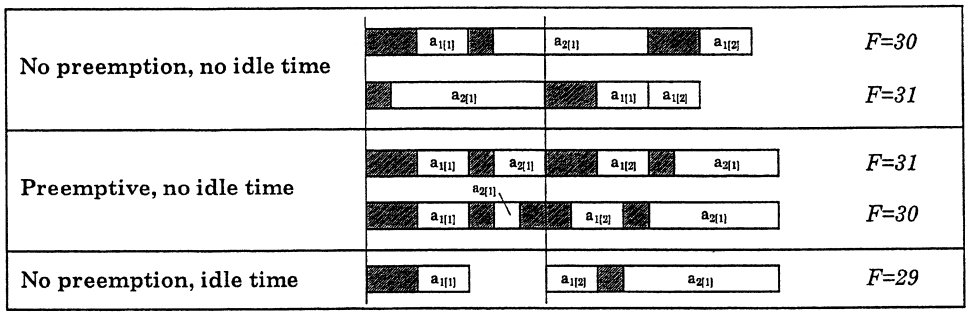


FIGURE 2.12 Schedules for instance *I*.

Minimizing Makespan with Sequence-Dependent Setup Times

This section considers the problem of scheduling N statically available independent jobs with individual sequence-dependent setup times on a single machine in order to minimize the makespan. For this $1|s_{ij}|C_{\max}$ problem, the minimization of the makespan is equivalent to minimizing the sum of setup times in the schedule. Optimization of this problem does not require consideration of individual job completion times, so that the duration of job processing times does not affect the optimal sequence. We will assume that the state of the machine after processing the last job is not of concern.

In Section 2.2 it was illustrated that the $1|s_{ij}|C_{\max}$ problem is equivalent to the strongly *NP*-hard (asymmetric) traveling salesperson problem (ATSP). The focus of this section will be upon relatively simple heuristic algorithms that illustrate certain basic scheduling strategies useful for solving the $1|s_{ij}|C_{\max}$ problem. Golden and Stewart (in [52]), Golden, Bodin, Doyle, and Stewart [37] and Reinelt [70] {Reinelt1994} provide useful surveys of the literature pertaining to the TSP and ATSP. These surveys can provide a basis for a more in-depth investigation.

When addressing the $1|s_{ij}|C_{\max}$ problem, Gavett [32] assumes that the machine is initially processing a particular job (let this be a_0), yet a machine which is shut down initially can be represented by considering a_0 to be a "dummy job." The setup time s_{0j} from a dummy job a_0 to any job a_j ($1 \leq j \leq N$) is assigned a value equal to the initial setup time for a_j. This dummy-job representation of the initial state of the machine is not an uncommon approach; for example, in solving the $1|s_{ij}|C_{\max}$ problem, Foo and Wager [29] utilize it when applying a dynamic program for the ATSP developed by Bellman [7] and independently by Held and Karp [41]. It can also be noted that the notation s_{0j} is used within this chapter as a standard means of denoting an initial setup for job a_j.

Gavett presents three simple heuristics for solving the $1|s_{ij}|C_{\max}$ problem, each a dispatch-type heuristic. These heuristics will be identified here using the prefix '*G*'− followed by Gavett's own identification.

- *Heuristic G − NB* ('*Next Best*' *rule*): At each decision point, select the unscheduled job with the least setup time from the job just completed.
- *Heuristic G − NB′* ('*Multi-start*' *version of G − NB*): For each unscheduled job (a_1, a_2,…, a_N) create a partial schedule in which this job immediately follows job a_0. Apply the *G − NB* heuristic to each of the N partial schedules produced, and select the best completed schedule as the solution. Where two or more unscheduled jobs have an equal-least setup time from the job just completed, create a partial schedule for each possibility.
- *Heuristic G − NB″* ('*Next Best with Column Deductions*' *rule*): Utilize heuristic *G − NB*, after the non-eliminable component of each job's setup time is incorporated into the job processing time.

One member of the set of solutions generated by the *G − NB′* rule will correspond to the solution obtained using the *G − NB* rule. Thus, it is not possible for the *G − NB* rule to outperform the *G − NB′*

rule on objective function value. The process of modifying setup times for use in $G - NB''$ is straightforward. Considering job a_j $(1 \leq j \leq N)$, the minimum setup time to this job, $s_{*j} = \min_{i \in \{0 \ldots N\} \setminus j} s_{ij}$, is found and subtracted from all setup times s_{ij}. Observing that the setup times in a problem with initial setups can be written as an $(N + 1) \times N$ matrix, with rows (0, 1,…, N) corresponding to preceding jobs and columns $(1, 2, \ldots, N)$ to succeeding jobs, this process is equivalent to subtracting the minimum value in each column from all values in the column (excepting each s_{jj} element); hence the term *column deductions*. A fourth rule, combining column deductions with multi-start, is an extension not considered by Gavett.

The $G - NB$ and $G - NB''$ heuristics both have $O(N^2)$ running times. At a stage where n jobs have already been scheduled $(n < N)$, the last of these being a_i, there are $N - n$ setup times s_{ij} to choose from. Assuming that the comparing, choosing, and other intervening steps take $O(1)$ time each, schedule construction requires $O(\sum_{n=1}^{N}(N - n)) = O(N^2)$ time, and as the "column deductions" procedure requires $O(N^2)$ time also, the $O(N^2)$ bound results. If implemented as described above, the $G - NB'$ procedure is in fact non-polynomial; if all setup times are equal, the $G - NB'$ rule will explore all $N!$ possible sequences (hence $O(N!)$ operations). This worst-case behavior could be countered using simple modifications to $G - NB'$ to give an $O(N^3)$ heuristic.

The thorough testing of these heuristics carried out by Gavett utilizes both normal and uniform distributions for setup times. For problems containing up to 20 jobs, the performance of each heuristic is assessed relative to optimum values obtained through application of the branch-and-bound algorithm developed by Little, Murty, Sweeney, and Karel [57]. The mean setup time (μ) in each test is taken to be one time unit. Sample sizes were between 10 and 100 problems at each level of number of jobs N and standard deviation σ. With mean setup times of $\mu = 1$, σ also represents the coefficient of variation in setup times (σ/μ).

Gavett's results illustrated that when setup times are normally distributed, the average deviation of the $G - NB$ heuristic from optimal was between 2% and 26%, while the average deviation of the $G - NB'$ heuristic from optimal was between 1% and 18% (these figures being the mean value of $(C_{\max}^{H} - C_{\max}^{OPT})/C_{\max}^{OPT}$ expressed as a percentage, where $C_{\max}^{H}$ is the makespan obtained using a heuristic and $C_{\max}^{OPT}$ the optimal makespan). The data presented by Gavett illustrates that a strong positive relationship existed between deviation from optimal and increases in the coefficient of variation of setup times (Fig. 2.13). The $G - NB''$ rule was not tested using normally-distributed setup time data.

For uniformly distributed setup times, the performance of the heuristics diminished (illustrated in Fig. 2.13). Of the three heuristic rules, the $G - NB''$ rule appeared to perform considerably better than the other two for most problem sizes; although for the greatest number of jobs tested (20), the $G - NB'$ heuristic displayed a slight advantage. As noted by Gavett, this may have been a chance occurrence (as may be the unusual result for $G - NB$ with $N = 9$). For uniform setup times in particular, it is seen that "next-best" type rules are not overwhelmingly successful in dealing with the $1|s_{ij}|C_{\max}$ problem.

The $G - NB'$ rule clearly outperformed the $G - NB$ rule for most problem sizes and for both setup time distributions. Nevertheless, Gavett makes the pertinent observation that the "principle" of the $G - NB$

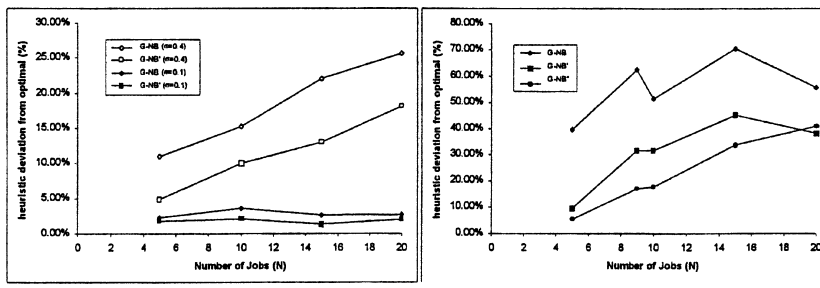


FIGURE 2.13 Experimental data provided by Gavett[32]: normally distributed setup times (left) and uniformly distributed setup times (right).

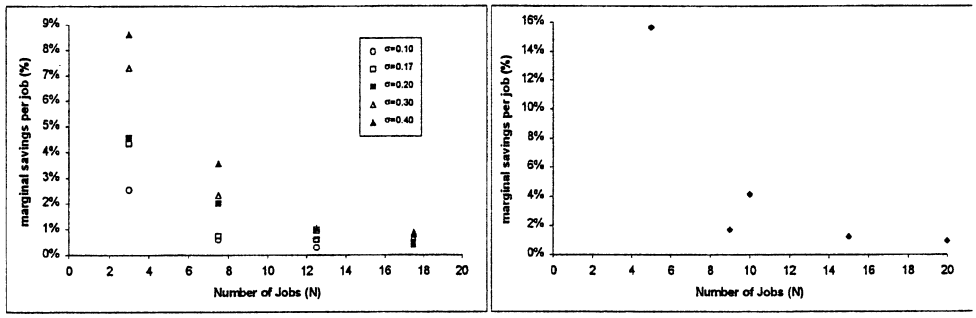


FIGURE 2.14 Effect of number of jobs N on percentage setup time savings, using data from Gavett [32]: normally distributed setup times (left) and uniformly-distributed setup times (right).

rule can be used as an on-line dispatch heuristic in cases where reliable setup time data is unavailable and/or sequencing is being carried out manually on the shop floor. The selection of the next part according to estimated least setup time becomes the rule.

Using the data from tests with normally distributed setup times, Gavett compared the expected makespan $\bar{C}_{\max}$ calculated from knowledge of the setup times distribution ($\bar{C}_{\max} = N\mu$) with the mean makespan ($\bar{C}_{\max}^{NB}$) obtained using the $G - NB$ rule. With $\bar{C}_{\max}$ being representative of the makespan obtained through random sequencing of jobs, the ratio $(\bar{C}_{\max} - \bar{C}_{\max}^{NB})/\bar{C}_{\max}$ provides a measure of the expected reduction of total setup time resulting from effective scheduling. In Fig. 2.14, $\bar{C}_{\max}^{OPT}$ has been utilized for a similar comparison using Gavett's published data, and the *marginal saving per (additional) job* is plotted. These marginal savings have been calculated according to the formula

$$MS\left(\frac{a+b}{2}\right) = \frac{PR(b) - PR(a)}{b-a}$$

where $MS(n)$ is the estimated additional saving of setup time obtained by increasing number of jobs n by one job (i.e., marginal saving); $PR(n)$ is the performance ratio $(\bar{C}_{\max} - \bar{C}_{\max}^{OPT})/\bar{C}_{\max}$ for n jobs; and a and b are numbers of jobs ($a < b$, $a, b \in \{1, 5, 10, 15, 20\}$). $PR(1)$ is taken to be equal to zero, as no reduction in setup time is possible with one job only.

As shown in Fig. 2.14, for normally distributed setup times with $\sigma = 0.40$, the marginal saving in setup time drops from 8.6% per job ($N = 3.0$) to 0.9% per job ($N = 17.5$). Similar trends are evident for each other coefficient of variation as well as for uniformly distributed setup times, although care must be taken when drawing conclusions from such a restricted data set.

With diminishing returns per job, there exists a point at which the advantages of considering a larger number of jobs N on a particular scheduling occasion may be outweighed by the effort of computing schedules for larger problems. Considering a subset of the jobs, corresponding to those at the beginning of the "queue," may represent an effective compromise. It must also be considered that the makespan objective is purely utilization oriented and does not take into consideration the requirements of individual jobs (and the customers whose orders they represent). In a practical environment, full implementation of schedules may not occur. "New" jobs will become known and available at various times, re-scheduling processes will revise schedules, and the processing of jobs that had already been waiting might be postponed.

There exist no mechanisms such as deadlines or job weights in the $1|s_{ij}|C_{\max}$ problem to ensure that a waiting job actually gets completed in a dynamic environment with re-scheduling. For example, if a "next best"-type rule is utilized for scheduling, a job with significant dissimilarity compared to other waiting jobs will be assigned a large setup time, and might invariably be sequenced near the end of schedules. To counter this behavior, we might impose a certain *batch size* N'. A schedule is created for the first N' waiting jobs, the schedule is carried out in full, and the process is repeated for a new batch.

The improved job-completion performance that results will be obtained at the cost of a minor reduction in utilization (i.e., makespan minimization). The batch size decision will be aided by results such as those contained in Fig. 2.14; choosing a batch size of 10 to 15 jobs, for example, may represent a reasonable compromise. The observed distribution of setup times will play a role in this determination of batch size.

In each of Gavett's heuristics, scheduling decisions look forward one job into the schedule. Baker [5] extends Gavett's ideas and proposes that the next job be selected on the basis of looking two jobs ahead. This selection rule may be incorporated into each of the Gavett heuristics, as well as the "column deductions plus multi-start" rule, although only modification of $G - NB$ will be considered.

Heuristic $B - NB$ ("Next-best" heuristic utilizing Baker's two-job look-ahead rule):
The heuristic is similar to $G - NB$ except that it deals with pairs of jobs. Let the currently-last scheduled job be a_i. For each pair (a_j, a_k) of unscheduled jobs, calculate the total setup time $s_{ij} + s_{jk}$. Select the pair $(a_{j'}, a_{k'})$ with the least setup time sum and schedule the pair next (i.e. after a_i).

The running time of this heuristic is $O(N^3)$, compared to $O(N^2)$ for the basic $B - NB$ rule, as there are $O(N^2)$ setup time pairs to survey at each decision point. Morton and Pentico [63] suggest a further variation, denoted here by $M\&P - NB$.

Heuristic $M\&P - NB$ ('Next-best' heuristic utilizing a variant of Baker's two-job look-ahead rule):
The heuristic is identical to $B - NB$ except that after selecting the pair $(a_{j'}, a_{k'})$ with the least setup time sum, only $a_{j'}$ is scheduled next.

The "next-best"-type rules can be compared to the "savings"-based rules utilized by Charles-Owaba and Lambert [16] and White and Wilson [77] for the same problem. Charles-Owaba and Lambert utilize their model of setup times (Section 2.2) to define the saving in setup time that results from scheduling job a_j immediately after job a_i as:

$$\bar{s}_{ij} = s_{0j} - s_{ij} \tag{2.14}$$

that is, equal to the sum of the durations of machine setup tasks eliminated by similarity in a_i and a_j. Their proposed heuristic selects the next job based on the maximization of the total setup time "saving."

Heuristic CO&L (Charles-Owaba and *Lambert heuristic)*:
Select the pair of jobs (a_i, a_j) with the greatest value of $\bar{s}_{ij}$, and schedule these as the first and second jobs, respectively, in the sequence. Then add jobs to the schedule by choosing the job α_j yielding the greatest saving in setup time from the currently last scheduled job a_i (i.e., a_j with $\max_{j \in J'}\{\bar{s}_{ij}\}$, where J' is the set of unscheduled jobs).

Equation (2.14) turns out to be a particular case of a general selection rule for TSP-type problems originally proposed within a heuristic by Clarke and Wright [19] for the *vehicle routing problem (VRP)*. The Clarke and Wright heuristic uses expression (2.15) to calculate savings, this expression reducing to (2.14) for the current problem where final teardown times are not modeled (i.e., $s_{j0} = 0\ \forall j$; in terms of the ATSP, there is no cost associated with the return to the "initial city").

$$\bar{s}_{ij} = s_{i0} - s_{ij} \tag{2.15}$$

This expression for $\bar{s}_{ij}$ can be interpreted as being the setup-time difference between (1) setting the machine to a bare state 0 once a_i completes, then setting the machine for a_j, and (2) after processing a_i, directly setting the machine for processing a_j. In cases where the makespan is defined to include a final teardown after completion of the last-scheduled job, expression (2.15) can be used in preference to (2.14).

The heuristic originally proposed by Clarke and Wright for the VRP can be used for the ATSP (hence, the $1|s_{ij}|C_{\max}$ problem); the VRP is a more complex problem than the ATSP so that simplification of the

Clarke and Wright heuristic is possible. Golden et al. [37] present such a variant of Clarke and Wright's heuristic. The *CO*&*L* heuristic given in this section is a further simplification of the original heuristic.

To compare the performance of next-best heuristics ($G - NB$ and variants) to savings heuristics (*CO*&*L* here), we refer to the experimental analysis presented by Golden et al. regarding symmetric TSPs. The Clarke and Wright heuristic variant presented in that paper is shown to be clearly superior to a next-best heuristic (equivalent to the $G - NB$ heuristic) when tested on five large problems ($N = 100$). The savings heuristic attained a maximum deviation from optimal of 6.37%, while the best result obtained by the next-best heuristic was 13.27% from optimal. While only limited conclusions can be drawn from these results, it is reasonable to suggest that the *CO*&*L* savings heuristic should perform better, on average, than the next best-type heuristics for the $1|s_{ij}|C_{\max}$ problem.

Charles-Owaba and Lambert do not report on experimental analysis of their heuristic. They do provide a worked example, which also serves as an illustration of their setup-time modeling technique. For the example containing eight jobs, their heuristic returns the optimal solution and in doing so outperforms the three heuristics of Gavett as well as the $B - NB$ heuristic. It is interesting to note that although the Charles-Owaba & Lambert setup times model does not necessarily provide symmetric setup times, setup-time savings calculated using this model are always symmetric (i.e., $\bar{s}_{ij} = \bar{s}_{ji}$).

White and Wilson utilize an approach based on their study of setup times on a particular model of a lathe (a "mono-matic" lathe). They begin by developing a model for the prediction of setup times, through statistical analysis of actual setup time data using a technique known as *multiple classification analysis*. The factors {*A*, *B*, *C*,…} thought to influence setup times are proposed, and the various levels of these factors are determined. The statistical technique is then used to develop a predictive equation for setup times; the time $S_{ijk}\ldots$ for a setup to a component associated with level *i* of factor *A*, *j* of factor *B*, etc. is expressed as:

$$S_{ijk\ldots} = \bar{S} + a_i + b_j + c_k + \ldots + \epsilon_{ijk\ldots} \tag{2.16}$$

where $\bar{S}$ is the grand mean of setup times, $\epsilon_{ijk\ldots}$ is the error in prediction, a_i is the effect of level *i* of factor *A*, b_j is the effect of level *j* of factor *B*, etc.

When this analysis is applied to the data collected, and after some simplification, White and Wilson are able to predict setup times on the "mono-matic" lathe using Eq. (2.17).

$$S = 31T_1 + 15A_2 + 11N_3 + 7N_2 + 4G_1 \tag{2.17}$$

which, as they show, is reasonably accurate although not without significant errors when compared to some setup time observations. In Eq. (2.17), $A_4 = 1$ if the workpiece holder (arbor) is changed, $G_1 = 1$ if spindle gears are changed, and $T_1 = 1$ if the tracer control element (template) is changed. Otherwise, these variables are zero. N_2 and N_3 are numerical variables associated with the number of tools inserted and required, respectively.

This predictive equation is then able to be used for scheduling production on the lathe. A formal heuristic definition is not provided by White and Wilson, although the idea is simple. As the "template change value" of 31 is the highest such value, all jobs sharing the same template will be sequenced together. Within the subset of jobs sharing the same template, jobs sharing the same arbor will be scheduled together. Hence, this heuristic is largely involved with setup saving.

As with Charles-Owaba and Lambert's article, no heuristic testing is undertaken other than an example showing that the proposed heuristic can provide an optimal solution. Another common aspect between the two papers is that both heuristics perform significantly better than the simple next-best proposed by Gavett when applied to the example problems. Both papers illustrate that analysis and subsequent exploitation of the special structure of sequence-dependent setup times in machine scheduling can lead to significantly improved schedule quality, compared to what otherwise might be achieved in similar time by general-purpose heuristics such as the next-best rules.

A commonality is seen in the White and Wilson and the Charles-Owaba and Lambert setup times models. Both are concerned with the similarity/dissimilarity of components and the subsequent relationship this has with setup activities and their durations. Each deals with binary ("yes/no") measures of similarity, with the White and Wilson approach also allowing numerical measures to be included. A particular advantage of the Charles-Owaba and Lambert model however is that setup times may be predicted using standard (or estimated) machine setup task durations, while the White and Wilson approach requires the collection of setup time data from an existing system over a period of time.

It is in the modeling of setup times, and the investigation of the technological basis of these times, that these two papers make—arguably—the greatest contribution. For the machining processes analyzed in both articles, it can be noted that the number of parts to be scheduled is relatively small while setup times are large (several minutes to a few hours). The magnitude of any of these times is likely to be greater than that required for optimal or near-optimal scheduling using advanced ATSP heuristics. Thus, such setup time models would provide greatest benefit for solution of problems more complex than $1|s_{ij}|C_{\max}$. And, it is unfortunate that (to our knowledge) published reports of such applications have not appeared.

Minimizing Makespan in Class Scheduling Problems

Where jobs are grouped into classes, and the setup times between jobs of the same class are assumed negligible, all jobs belonging to the same class should be sequenced consecutively if minimizing the makespan (provided jobs are statically available and do not have deadlines imposed, and setups satisfy the triangle inequality). The following simple argument proves the result.

Referring to Fig. 2.15, let jobs b and e, separated by jobs $c,\ldots, d$ in a sequence S_1, belong to the same class so that $s_{bj} = s_{ej}$ and $s_{je} = s_{jb}$ for all $j \neq b, e$. Jobs c, d, and f thus belong to classes other than that of b and e. Exchanging the position of $c,\ldots, d$ with that of e (yielding sequence S_2) leads to a net change Δ in the total setup time scheduled between the beginning of job b and the completion of job f.

$$\Delta = (s_{ec} + s_{df}) - (s_{bc} + s_{de} + s_{ef}) \tag{2.18}$$

As $s_{bc} = s_{ec}$, and $s_{df} \leq s_{de} + s_{ef}$ by the triangle inequality, Δ takes a non-positive value and thus S_2 is at least as good as S_1. Repeated application of such interchanges will lead to a sequence that has all jobs of each class sequenced consecutively. Expression (2.18) also indicates that when setup times between classes do not obey the triangle inequality, this "group scheduling rule" may not be valid. Regardless of whether the triangle inequality is satisfied, the order in which jobs of the same class appear in a schedule is immaterial when seeking to optimize makespan.

When minimizing the makespan in a (static, no deadlines) class scheduling problem where setup times satisfy the triangle inequality, the jobs $\{a_{i[1]}, a_{i[2]},\ldots a_{i[N_i]}\}$ belonging to each class $i(1 \leq i \leq B)$ may be treated as a single *composite job* with combined processing time $p_i = \sum_{j=1}^{N_i} p_{i[j]}$.

The formation of composite jobs will significantly reduce the number of (composite) jobs requiring consideration when scheduling. This can bring great benefits, particularly when the scheduling problem is *NP*-hard and the running time of optimal algorithms rises exponentially with N, that is, when class setup times are sequence dependent. Clearly, when classes are formed into composite jobs, the resulting $1|s_{ij}|C_{\max}$ problem

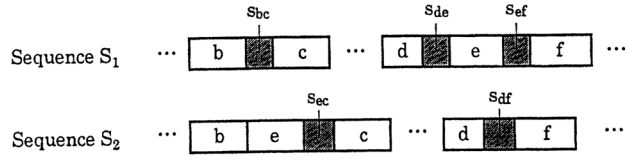


FIGURE 2.15 Reduction in makespan in a problem where setup times satisfy the triangle inequality.

"without classes" can be solved using scheduling algorithms such as those of the previous section. Without the triangle inequality, the composite-job approach is not valid and it may be necessary to treat each job separately.

Makespan with Additive Changeovers

As described in Section 2.2, the additive changeovers model of setup times incorporates sequence-independent setup times and sequence-independent teardown times. Changeovers from a class i to a class $k (i \neq k)$ have a duration given by the sum of the teardown t_i for class i and the setup s_k for class k.

Although each class may, in general, contain more than a single job, the results of the previous section indicate that to minimize the makespan, all jobs $a_{i[j]}$ belonging to the same class i can be combined into a composite job a_i. Thus, in this section when we refer to "job a_{ii}" we are in fact making reference to the composite job containing all class i jobs.

In (static, no deadlines) single machine makespan problems with additive setup times, savings of non-productive time are afforded at the beginning and end of a schedule. The potential for these savings to be made will appear when either:

1. The objective is to minimize the maximum completion time $C_{\max}$ (i.e., $\max_{j \in J}\{C_j\}$), or
2. No initial setup is necessary ($s_{0i} = 0 \; \forall i$) and the objective is to minimize the *machine occupation time* $(C + t)_{\max}$ (i.e., $\max_{j \in J}\{C_j + t_j\}$)

A relevant application of a model without initial setup times occurs in situations where machine warm-up time is greater than the setup time for any class. The setup for the first-scheduled class is consequently "absorbed" into the warm-up time, and thus saved (applicable to cases 1 and 2). Furthermore, when the teardown for the last class does not influence the makespan, choosing this last-scheduled class to be one with a "long" teardown can improve the schedule performance (case 1 only).

For the $1|s_+, s_{0i} = 0|C_{\max}$ problem (the static single machine makespan problem with additive changeovers and no initial setups), the objective is to minimize

$$\begin{aligned} C_{\max} &= p_{(1)} + t_{(1)} + \sum_{j=2}^{N=1} (s_{(j)} + p_{(j)} + t_{(j)}) + s_{(N)} + p_{(N)} \\ &= \sum_{j=1}^{N} (s_{(j)} + p_{(j)} + t_{(j)}) - (s_{(1)} + t_{(N)}) \end{aligned} \tag{2.19}$$

where $p_{(j)}$, $s_{(j)}$, and $t_{(j)}$ represent the processing time, setup time, and teardown time of the job occupying sequence position j ($j = 1, \ldots, N$), respectively.

From Eq. (2.19) it is clear that an optimal solution to this problem maximizes $s_{(1)} + t_{(N)}$, as the value of the summation term is fixed for a given instance. Let i_k be the job with the kth greatest setup time, and j_k be the job with the kth greatest teardown time ($1 \leq k \leq N$). To minimize the makespan, job i_1 should be sequenced first and j_1 sequenced last, unless $i_1 + j_1$, in which case a sequence providing the minimum makespan is either $\{i_2, \ldots, j_1\}$ or $\{i_1, \ldots, j_2\}$. Determining i_1, i_2, j_1, and j_2 occupies $O(N)$ time.

When an initial setup is necessary ($1|s_+|C_{\max}$), the solution of the problem reduces to maximizing $t_{(N)}$; $s_{(1)}$ is removed from the optimization. Similarly, if the objective is to minimize $(C + t)_{\max}$ and no initial setup is necessary, the problem reduces to maximizing $s_{(1)}$. With initial setups, and inclusion of teardowns in the makespan, any sequence of (composite) jobs is optimal. If scheduling merely with sequence-independent setup times, all $t_{(j)} = 0$; unless an initial setup is unnecessary, an arbitrary sequence of (composite) jobs is once again optimal.

Adding Dynamic Arrivals

The dynamic $1|r_j|C_{\max}$ problem is easily solved by scheduling the jobs in order of non-decreasing ready times. With general sequence-dependent setup times, the $1|r_j, s_{ij}|C_{\max}$ problem is clearly *NP*-hard, given that $1|s_{ij}|C_{\max}$ is. However, Bianco, Ricciardelli, Rinaldi, and Sassano [8] have made good progress in solving the $1|r_j, s_{ij}|C_{\max}$ problem. They report computational experience which illustrates that solution of 20 job instances is quite within the capabilities of a branch-and-bound approach.

It is interesting to note that the more restrictive ready times are, the easier it is to solve the instance. For example, Bianco et al. test their algorithm using 20 job instances where ready times are (uniformly) distributed over [0, 100], a significantly larger range than the [5,10] range used to generate sequence-dependent setup plus processing times. At any point in time, there are relatively few jobs that need be considered for scheduling in this situation. For these instances, an average of 98% of search nodes generated by the branch-and-bound procedure were eliminated by two simple dominance criteria, and computation times were considerable although not unreasonable (mean computation time of 115.3 seconds on a VAX-11/780 computer). In comparison, it took the same algorithm over 400 seconds on average to solve 15 job instances when ready times were distributed over [0, 50] and setup plus processing times were distributed over [1,10]. Only 50.5% of search nodes were eliminated by the dominance criteria in this latter case.

When additive changeovers are part of the single machine makespan problem, the computational complexity of the $1|r_{i[j]}, s_+|C_{\max}$ and $1|r_{i[j]}, s_+, pmtn|C_{\max}$ problems appear to be open, although at least special cases of these problems are efficiently solvable. It should be noted that for a dynamic problem, the jobs within a class cannot, in general, be considered a composite job. We assume that ready times govern the earliest possible starting time for job processing and do not impose restrictions on the scheduling of changeovers.

Where idle time appears in a schedule for a dynamic makespan problem, this should correspond to situations where no job is currently available for processing. It is clear that there is no benefit in purposefully delaying the processing of available jobs. When setup times are considered, inserted idle time must not delay the undertaking of setup or teardown operations if this delay eventually leads to a job starting later than otherwise possible.

In a dynamic makespan problem where all jobs in a class have a common ready time (i.e., $r_{i[j]} + r_i$, where r_i is the *class ready time*) and no individual job deadlines are imposed, there is neither need nor benefit in splitting classes, so that the problem may be simplified by forming composite jobs (unless setup times are sequence dependent and do not satisfy the triangle inequality). Algorithms for makespan problems with additive changeovers, class ready times, and dynamic arrivals can be constructed by incorporating previously introduced ideas for the $1|s_+|C_{\max}$ and $1|r_j|C_{\max}$ problems.

Starting with the $1|r_{i[j]} = r_i, s_+|(C + t)_{\max}$ problem, if the definition of a composite job a_i is modified so that composite job processing times are given by $p_i = s_i + \sum_{j=1}^{N_i} p_{i[j]} + t_i$, then the original $1|r_{i[j]} = r_i, s_+|(C + t)_{\max}$ problem is able to be solved as a $1|r_j|C_{\max}$ problem.

Where no initial setup is necessary and the makespan is taken to be the maximum completion time, (i.e., the $1|r_{i[j]} = r_i, s_+, s_{0i} = 0|C_{\max}$ problem), the objective function (2.19) used for $1|s_+, s_{0i} = 0|C_{\max}$ is not appropriate, as there exists the possibility that idle time will be scheduled. The "idea" of Eq. (2.19) holds, however, and the problem remains efficiently solvable as the optimization decision involves choosing a class i_P to be sequenced first and a class i_Q to be sequenced last ($i_P \neq i_Q$). Given a particular pair $\{i_P, i_Q\}$ selected from the $B(B-1)$ permutations available, $O(B)$ operations are required to construct a schedule *between* these two classes. This schedule construction proceeds by sequencing classes other than i_P and i_Q in order of non-decreasing class ready time. Consequently, an algorithm which evaluates the makespan for every possible pair $\{i_P, i_Q\}$ can find the optimal solution in $O(B^3)$ time.

For the problem of minimizing machine occupation time (the $1|r_{i[j]} = r_i, s_+, s_{0i} = 0|(C + t)_{\max}$ problem), it is sufficient to specify only the first-sequenced class, hence this variant of the dynamic makespan problem with class release dates can be solved in $O(B^2)$ time.

If deadlines are introduced in addition to changeovers and release dates, it is straightforward to show that the problem $1|r_{i[j]} = r_i, s_i, \bar{d}_{i[j]} = \bar{d}_i|C_{\max}$ with class ready times and class deadlines is *NP*-hard. This is possible using a restriction to either the $1|r_j, \bar{b}_j|C_{\max}$ problem without setup times, or the $1|s_i, \bar{b}_j|C_{\max}$ problem without release dates.

Makespan with Major–Minor Setup Times

The ideas of the preceding sections can be adapted for the solution of single machine makespan problems incorporating the major–minor setup times model. Recall that in this model, the setup time between two jobs is given by the sum of the major setup time s_{MJR} and minor setup time s_{MNR}. The major setup

time is zero if the two jobs belong to the same class, while the minor setup time is zero if the two jobs belong to the same sub-class. All jobs belonging to a particular sub-class $i(k)$ are members of the same class i, so that a minor setup always follows a major setup ($1 \leq i \leq B$, $1 \leq k \leq b_i$).

Typically, both the major and minor setup times are considered sequence independent. This section addresses the makespan problem where the additive changeovers model applies to both major and minor setup (changeover) times.

It was seen in preceding sections that minimizing makespan with additive (major) changeovers simply called for selection of the first-scheduled and last-scheduled classes. When changeovers satisfy the triangle inequality (as additive changeovers do) and jobs are statically available, all jobs of a class should be sequenced together. Hence, the total work (time) content of a class is fixed. The opportunity for "saving" in setup time is thus offered (1) at the beginning of the schedule if the setup for the first-scheduled class can be "absorbed" into the warm-up time on the machine (no initial setups), and (2) at the end of the schedule, if the final teardown is not included in the makespan ($C_{\max}$ objective). Essentially, the class with the greatest teardown should be scheduled last, and the class with the greatest setup should be scheduled first.

With the introduction of minor setup times, the strategy is unchanged. Both major, minor, and resulting total changeover times satisfy the triangle inequality in this model, so that to minimize makespan, all jobs belonging to any given sub-class $i(k)$ should be sequenced consecutively. Similarly, all composite jobs (sub-classes) belonging to a given class should be sequenced consecutively. In the process of developing an optimal schedule, we first determine the ordering of sub-classes within classes, and then determine the ordering of classes within the schedule.

When switching between classes, the setup time is given by $S = t_{MNR} + t_{MJR} + s_{MJR} + s_{MNR}$. If it has been determined that a particular class i is to be sequenced last, the greatest saving in makespan is afforded by scheduling last the sub-class $i(k)$ with the maximum minor teardown time of all sub-classes of i ($1 \leq k \leq b_i$). Likewise, when initial setups (major and minor) are not required, and the first-scheduled class has been predetermined, a job belonging to the sub-class with the greatest minor setup time should be sequenced first.

The strategy is then clear. If there are no initial setups, schedule first the class and sub-class that yields the highest possible value of $s_{MJR} + s_{MNR}$; and if final teardowns are not included, schedule last the class and sub-class that yields the highest possible value of $t_{MJR} + t_{MNR}$. These observations can now be included into an optimal algorithm to minimize makespan on a single machine with static job availability, additive changeovers, and no initial setups. If solving the $1|s_+{:}s_+|C_{\max}$ or $1|s_+{:}s_+, s_{0i} = 0|(C + t)_{\max}$ problems, a simplified version of the algorithm can be used.

Algorithm for $1|s_+{:}s_+, s_{0i} = 0|C_{\max}$

Step 1: For each class i, determine the sub-class $i(k_S)$ with the greatest minor setup time and sub-class $i(k_T)$ with the greatest minor teardown time ($1 \leq k_S, k_T \leq b_i$). Let S_i be the *maximum total setup time* (major plus minor) to class i, given by $S_i = s_i + s_{i(k_s)}$; and T_i be the *maximum total teardown time* from class i, given by $T_i = t_i + t_{i(kT)}$.

Step 2: Let $\alpha(k)$ be the index of the class with the kth greatest value of maximum total setup time, and let $\beta(k)$ be the index of the class with the kth greatest value of maximum total teardown time. If $\alpha(1) \neq \beta(1)$, consider class $\alpha(1)$ to be first-scheduled and class $\beta(1)$ to be last-scheduled of all classes. Otherwise, compare the values of $S_{\alpha(1)} + T_{\beta(2)}$ and $S_{\alpha(2)} + T_{\beta(1)}$; whichever is greater, consider the relevant classes to be first- and last-scheduled.

Step 3: Schedule first a job belonging to the sub-class $i(k_S)$, where i is the class selected in step 2 to be scheduled first and k_S is the sub-class index defined in step 1. Schedule last a job belonging to sub-class $j(k_T)$, where j is the class selected in step 2 to be scheduled last and k_T is the sub-class index defined in step 1. Between these jobs, form a schedule within which all jobs of any given sub-class are sequenced consecutively, and all sub-classes of any given class are sequenced consecutively.

Dealing with each class in step 1 requires $O(b_i)$ operations, as b_i minor setup times and b_i minor teardown times are "searched" in finding the maximum value of each. As $\sum_{i=1}^{B} b_i \leq N$, step 1 requires $O(N)$ operations. Step 2 and the first sub-steps of step 3 require $O(B)$ operations, while formation of

a schedule in step 3 is undertaken in $O(N)$ time. Hence, the number of operations required for the algorithm is $O(N)$.

It can be noted that as long as the "jobs-within-subclasses" and "subclasses-within-classes" rules are followed, secondary criteria can be incorporated into step 3 of the algorithm without affecting the optimality with respect to makespan. Given the limitations of the makespan objective, the ability to apply secondary criteria in this way is welcome.

Flowtime Problems with Sequence-Dependent Setups

Flowtime problems with setup times are typically more difficult to solve in comparison to makespan problems. Nevertheless, it is interesting to note that the computational complexity of the "basic" $1|s_i|\Sigma C$ and $1|s_i|\Sigma wC$ problems remains an open question. In contrast, the single machine total flowtime problem with static job arrivals and sequence-dependent setup times ($1|s_{ij}|\Sigma C$) has been shown to be strongly *NP*-hard (hence, so is its counterpart, the more general $1|s_{ij}|\Sigma wC$ problem) [71].

Ahn and Hyun [3] show that optimal sequences for the $1|s_{ij}|\Sigma C$ problem have jobs that belong to the same class appearing in SPT order. Monma and Potts [62] independently prove the stronger result that SWPT-ordering of jobs within their classes holds in optimal solutions to the $1|s_{ij}|\Sigma wC$ problem, subject to setup times satisfying the triangle inequality. An EDD-based rule for the $1|s_{ij}|L_{\max}$ problem is also established by Monma and Potts within the same proof. Theorem 4 presents their result.

Theorem 4 (Theorem 1; Monma and Potts 1989)

For the maximum lateness problem, there is an optimal schedule for which the jobs within each class are ordered by the earliest due date (EDD) rule. For the total weighted completion time problem, there is an optimal schedule for which the jobs within each class are ordered by the shortest weighted processing time (SWPT) rule. That is, for some optimal sequence, if job $a_{i[j]}$ is scheduled before $a_{i[k]}$ then

$$d_{i[j]} \leq d_{i[k]} \qquad \text{for the } 1|s_{ij}|L_{\max} \text{ problem}$$

$$\frac{p_{i[j]}}{w_{i[j]}} \leq \frac{p_{i[k]}}{w_{i[k]}} \qquad \text{for the } 1|s_{ij}|\Sigma(wC \text{ problem})$$

and these jobs are considered to be in "preference order."

Proof

Consider an optimal schedule of the form $S = \{D, k, E, j, F\}$, where jobs j and k belong to the same class i, and D, E, and F represent arbitrary (partial) sequences of jobs. We wish to show that if jobs j and k are out of preference order in S, then one of the schedules $S' = \{D, j, k, E, F\}$ or $S'' = \{D, E, j, k, F\}$ is an alternative optimal schedule.

The triangle inequality ensures that pairwise interchanges of job j, job k, and partial sequence E, to generate sequence S' or S'' from S, will not produce any additional setup time in either schedule. As a result, setups in S, and subsequently in S' and S'', can be considered as jobs with an arbitrarily large due date, zero weight, and a duration as given in schedule S. This provides a problem without setup times and hence allows us to concentrate solely on whether the interchanges that generate the sequences S' and S'' preserve optimality (use of the SWPT and EDD rules as optimality conditions for problems without setup times becomes valid).

It is useful to replace the partial sequence of jobs E, including the setup jobs s_{kE} and s_{Ej}, by a composite job e (see note below). Both S' and S'' can now be generated from an optimal solution S by interchanging j, k, and e. Since j is preferred to k, then either k is preferred to e (S' is an optimal solution, and S'' is not), or e is preferred to j (S'' is an optimal solution, and S' is not). Whichever particular case holds, it is shown that there is an optimal solution in which jobs j and k are in preference order. ■

Note: For the maximum lateness problem, a composite job h for a sequence of two jobs $\{i, j\}$ is defined by $p_h = p_i + p_j$ and $d_h = \min\{d_j, d_i + p_j\}$. For the weighted flowtime problem, it is defined as $p_h = p_i + p_j$

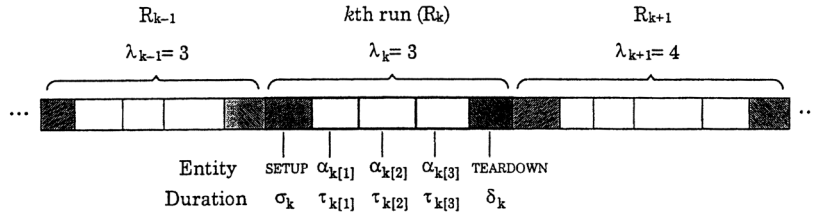


FIGURE 2.16 Runs in schedules for (static) class scheduling problems with setups and teardowns.

TABLE 2.12 Additional Notation for Class Scheduling Problems

Notation	Meaning
σ_k	Setup time for run R_k
δ_k	Teardown time for run R_k
λ_k	Number of jobs in run R_k $(1 \leq k \leq r)$
$\alpha_{k[j]}$	The jth job of run R_k
$\tau_{k[j]}$	Processing time of job $\alpha_{k[j]}$
$\varpi_{k[j]}$	Weight of job $\alpha_{k[j]}$
$\hat{r}_{k[j]}$	Ready time of job $\alpha_{k[j]}$
$C_{k[j]}$	Completion time of job $\alpha_{k[j]}$
B_k	Starting time of R_k, inclusive of setup time σ_k
T_k	Duration of run R_k, inclusive of setup and teardown $T_k = \sigma_k + \sum_{j=1}^{\lambda_k} \tau_{k[j]} + \delta_k$ for a static problem
W_k	The sum of the weights of jobs assigned to run R_k $W_k = \sum_{j=1}^{\lambda_k} \varpi_{k[j]}$

and $w_h = w_i + w_j$. A composite job may contain more than two jobs, in which case the definitions can be easily extended. Substitution of a composite job into a sequence will not change the maximum lateness, while such a substitution carried out for a flowtime problem increases the objective function by a constant (whose value depends on the weights and processing times of the jobs involved).

For the $1|s_{ij}|\Sigma C$ problem, Ahn and Hyun's proof of the SPT-within-classes rule does not require that setup times satisfy the triangle inequality. They consider two sequences $S = \{D, a_{i[k]}, E, a_{i[j]}, F\}$ and $\hat{S} = \{D, a_{i[j]}, E, a_{i[k]}, F\}$ (D, E and F as in Theorem 4) and evaluate the difference in flowtime between them in the case where $p_{i[j]} < p_{i[k]}$, showing that $\hat{S}$ is an improved sequence. The SWPT-within-classes rule represents the only strong optimality condition for the $1|s_{ij}|\Sigma C$ and $1|s_{ij}|\Sigma wC$ problems.

Runs and Batches

In a optimal schedule for a problem with class setups, it is often the case that the jobs of each class appear in two or more "groups" in the schedule. We will use the term *runs* to describe these "groups," which can alternatively be thought of as *batches*. By convention, the setup preceding a run and the teardown following a run will be thought of as "belonging" to that run (Fig. 2.16). Viewing schedules as being comprised of a series of runs is convenient and highly useful in many situations.

For a single machine problem, R_k will be used to denote the kth run in the schedule and the symbol r will denote the index of the last run in the schedule. Table 2.12 provides useful notation for class scheduling problems, which will be used in addition to that introduced so far.

A Greedy Heuristic for Sequence-Dependent Setup Times

Gupta [40] proposes a "greedy" heuristic for $1|s_{ij}|\Sigma C$. Running in $O(BN + N\log N)$ time, the heuristic builds a sequence by selecting the as-yet unsequenced job that can be completed in the shortest possible time. The *dispatching rule* followed by the heuristic can be conveniently referred to as the *shortest effective processing time* (SEPT) rule. We will consider a simple extension of this rule, the *shortest weighted*

effective processing time (SWEPT) rule, and imbed it into a version of Gupta's heuristic. The resulting heuristic takes the SWPT-within-classes rule into account and operates on the $1|s_i|\Sigma wC$ problem.

The SWEPT rule (for the $1|s_{ij}|\Sigma wC$ problem) states that at each decision point (job completion time) first determine, for each class i, the total time $\rho_{i[j]}$ required to process the job $a_{i[j]}$ with the least value of $p_{i[j]}/w_{i[j]}$ for this class. The time $\rho_{i[j]}$ is the job processing time, plus setup time if a setup will be required. Then select the job with the least value of $\rho_{i[j]}/w_{i[j]}$. The procedure given below implements this dispatching rule.

Greedy Heuristic for $1|s_{ij}|\Sigma wC$

Step 1. Arrange jobs within classes in non-decreasing order of weighted processing times (SWPT order).

Step 2. Consider each class i which contains at least one unscheduled job; let the job with the least weighted processing time of unscheduled class i jobs be $a_{i[n_i]}$. Calculate the effective processing time for job $a_{i[n_i]}$, this being given by $\rho_{i[n_i]} = s_{ki} + p_{i[n_i]}$, where k is the class of the currently last-scheduled job ($s_{kk} = 0 \forall k$). Select the job with the minimum value of ρ/w and schedule it next. Repeat this step until all jobs are scheduled.

"Greedy" heuristics cannot be expected to produce solutions of high quality. For the $1|s_{ij}|\Sigma C$ problem, Ahn and Hyun [3] have shown that the flowtime of schedules produced by Gupta's heuristic are often considerably greater than optimal flowtimes (see Section 2.3). A significant factor in this performance is the tendency for an excessive number of setups to be scheduled. Because the SEPT/SWEPT sequencing rules only "look ahead" to the next job in the schedule (i.e., they are *myopic*), they are prone to scheduling setups in situations where it otherwise would be best to continue processing the current class. As observed by Ahn and Hyun, this greedy approach leads to a solution method which effectively "ignores the group nature" of the jobs concerned.

The short running time of the heuristic is an advantage, however, particularly as it can be used as a "seed generator" for more advanced heuristics, such as that proposed by Ahn and Hyun. In addition, the SWEPT rule incorporated within the heuristic can be easily adapted to deal with more complex problems, such as the $1|r_j,s_{ij}|\Sigma wC$ problem.

Dynamic Programming Approaches

Dynamic programs for the $1|s_{ij}|\Sigma wC$ problem, and related flowtime problems, have been proposed by a number of authors. Table 2.13 provides a summary. Each of the dynamic programs noted in Table 2.13 provides an optimal solution to the problem. The terms "forwards recursion" and "backwards recursion" will be explained shortly.

TABLE 2.13 Dynamic Programming Approaches to the $1|s_{ij}|\Sigma wC$ Problem

Problem	Authors
$1\|p_{i[j]} = p_i, w_{i[j]} = w_i, s_{ij}\|\Sigma wC$ Backwards recursion, applies also to objectives other than flowtime	Psraftis, 1980 [69]
$1\|s_{ij}\|\Sigma C$ Backwards recursion, based on Psaraftis formulation	Ahn and Hyun, 1990 [3]
$1\|s_{ij}\|\Sigma wC$ Forwards recursion, adaptable to $1\|s_{ij}\|L_{\max}$ problem	Monma and Potts, 1989 [62]
$1\|s_{ij}\|\Sigma wC$ Backwards recursion, based on Psaraftis/Ahn and Hyun formulation	Ghosh, 1994 [33]
$1\|s_{ij}, B = 2\|\Sigma wC$ Forwards recursion, based upon Monma and Potts formulation	Potts, 1991 [67]
$1\|s_i\|\Sigma wC$ Forwards recursion, related DP for parallel machine problem also provided	Bruno and Sethi, 1978 [13]

Although we do not aim to convey a thorough understanding of dynamic programming to readers, by studying various dynamic programming approaches to the $1|s_{ij}|\Sigma wC$ problem we hope to outline essential dynamic programming ideas. Texts such as Dreyfus and Law [25] present a more extensive treatment.

For an instance of the $1|s_{ij}|\Sigma wC$ problem with 9 jobs (each provided with a sequence-dependent setup time), there are 362,880 schedules. Finding the optimal solution by enumerating all schedules and calculating the flowtime for each is not advisable for this instance, and is certainly not reasonable for larger problems; given $N!$ schedule, at least $O(N!)$ time would be required. Dynamic programming offers a more efficient means of determining the optimal solution.

Consider a situation where, by some means, it is established that jobs a_1, a_2, a_3, and a_4 occupy (in some order) the first four sequence positions in an optimal schedule. When filling the fifth sequence position, there are $4! = 24$ partial schedules containing jobs a_1, a_2, a_3, and a_4 that can be built onto. Now, if we know that the partial sequence $S_{2134} = a_2, a_1, a_3, a_4$ has a lower flowtime than any other partial sequence of these four jobs *with* a_4 *fourth* (e.g., a lower flowtime than S_{2314} or S_{1234}), then we should always build onto S_{2134} in preference to any of these alternatives. For each of the three other options for the fourth job (a_1, a_2, and a_3) there will also be a dominant sequence, that is, that which has the lowest flowtime compared to alternatives.

Thus, of the 24 "candidate" partial schedules (*subproblems*) containing $a_1, \ldots, a_4$ that could be extended, only four are worth extending. Furthermore, in deciding which job is best to schedule fifth (given $a_1, \ldots, a_3$ first, a_4 fourth), we are only immediately interested in (1) which are the four preceding jobs, (2) the job sequenced fourth, so we can calculate the setup time from it, and (3) the completion time of the fourth job, so that the flowtime of the fifth job can be calculated. These observations encapsulate some of the key ideas of dynamic programming.

In a dynamic program, a partial schedule is represented by a *state*. A state contains enough information to allow the *optimal value function* (in the example, the optimal flowtime of a sub-problem) to be calculated with knowledge of (1) the current state, (2) a previous state, and (3) a *recurrence relation* linking the optimal value for the current state to that of the previous state.

In the example given above, the dynamic program builds a sequence *forwards*. It is in fact more common for dynamic programs to work *backwards* from the last job to the first. The direction of progression is dictated by the form of the state and the recurrence relation, so that *forwards recursion* and *backwards recursion* become the appropriate terms.

When solving the (static) one machine class scheduling problem $1|s_{ij}|\Sigma C$ with B classes, using a backwards-recursion dynamic program, a relevant state representation is:

$$(n_1, n_2, \ldots, n_i, \ldots, n_B; i) \tag{2.20}$$

where i is the class of the first-sequenced job in the partial schedule (but not necessarily the first overall) and n_1, n_2,... are the numbers of jobs of each class 1, 2,... in the partial schedule. It is this state representation, or similar, that is used in the dynamic programs by Bruno and Sethi, Psaraftis, Ahn and Hyun, and Ghosh.

SWPT-ordering of jobs within classes (Theorem 4) tells us which particular jobs are scheduled. Let jobs-within-classes be indexed in SWPT order, so that $p_{i[j]}/w_{i[j]} \leq p_{i[j+1]}/w_{i[j+1]}$ $(i = 1, \ldots, B, j = 1, \ldots, N_i)$. Given an indexing of jobs in this way, $n_2 = 3$ implies that $a_{2[N_2-2]}$, $a_{2[N_2-1]}$ and $a_{2[N_2]}$ appear in the partial schedule, in that order although not necessarily consecutively.

Using Expression (2.20) as a state representation for a three-class problem with 5 jobs in each class, we can consider state (3, 1, 2; 1) for example. The first job in the partial sequence belongs to class 1; and as $n_1 = 3$, this job is $a_{1[3]}$, the third-last job in this class. Jobs $a_{1[4]}$, $a_{1[5]}$, $a_{2[5]}$, $a_{3[4]}$, and $a_{3[5]}$ also appear in the partial sequence in some order.

Immediately-previous states to the current state (3, 1, 2; 1) are:

- (2, 1, 2; 1), where $a_{1[4]}$ begins the previous partial sequence
- (2, 1, 2; 2), where $a_{2[5]}$ begins the previous partial sequence
- (2, 1, 2; 3), where $a_{3[4]}$ begins the previous partial sequence

Let the optimal value function V for this problem be *the total flowtime of the partial sequence*, where *time* = 0 at the starting time of the first job in the partial sequence. We wish to express the value of V for a given partial sequence (state) in terms of V for the states it is built upon.

Consider adding job $a_{3[3]}$ to the partial sequence (beginning with $a_{1[3]}$) represented by state (3, 1, 2; 1) to yield state (3, 1, 3; 3). Job $a_{3[3]}$ will be placed before the already-scheduled jobs and so will "push" the completion times of these jobs later. Between $a_{3[3]}$ and $a_{1[3]}$, a setup s_{31} will be required, this along with the processing time of $a_{3[3]}$ increasing the completion time of each scheduled job by $p_{3[3]} + s_{31}$. There are six jobs in (3, 1, 2; 1), so that the flowtime increases by $6(p_{3[3]} + s_{31}) + p_{3[3]}$, the second term corresponding to the completion time of $a_{3[3]}$ itself.

As V is to represent the optimal value of (3, 1, 3; 3), it is not correct to say

$$V(3, 1, 3; 3) = V(3, 1, 2; 1) + 6(p_{3[3]} + s_{31}) + p_{3[3]}$$

as the optimal value of $V(3, 1, 3; 3)$ might follow from a state other than (2, 1, 3; 1). What is correct is to write

$$V(3, 1, 3; 3) = \min\left\{\begin{array}{l} V(3, 1, 2; 1) + 6(p_{3[3]} + s_{31}) + p_{3[3]} \\ V(3, 1, 2; 2) + 6(p_{3[3]} + s_{32}) + p_{3[3]} \\ V(3, 1, 2; 3) + 6(p_{3[3]} + s_{33}) + p_{3[3]} \end{array}\right\} \tag{2.21}$$

$$= \min_{1 \le k \le 3}\{V(3, 1, 2; k) + 7p_{3[3]} + 6s_{3k}\} \tag{2.22}$$

as (3, 1, 2; 1), (3, 1, 2; 2), and (3, 1, 2; 3) are the three "candidate" states that may precede (3, 1, 3; 3). Expression (2.22) is in fact a recurrence relation, the "heart" of a dynamic program.

Recurrence relation (2.23) for the three-class problem as a whole is a generalization of (2.22).

$$V(n_1, n_2, n_3; i) = \min_{\{k: n_k' > 0, \quad 1 \le k \le 3\}} \left\{ V(n_1', n_2', n_3'; k) + \left(\sum_{f=1}^{3} n_f\right) p_{i[N_i - n_i']} + \left(\sum_{f=1}^{3} n_f'\right) s_{ik} \right\} \tag{2.23}$$

In (2.23), n_k' is the number of class k jobs in the preceding state. As state $(n_1, n_2, n_3; i)$ represents a class i job sequenced first, $n_i = n_i' + 1$, while $n_k' = n_k$ for all other k. The class i job "added" will be the n_ith last job in class i, $a_{i[N_i - n_i + 1]} = a_{i[N_i - n_i']}$. The minimum value is selected over the range of class indices k which can appear as the class-of-the-first-job in previous states; for example, if $n_2' + 0$, then $k = 2$ would be excluded because the state $(n_1, 0, n_3; 2)$ is impossible.

Recurrence relation (2.23) is used in order to get us from the initial state (0, 0, 0; 0), where no jobs are scheduled (the very end of the schedule) to states $(N_1, N_2, N_3; 1)$, $(N_1, N_2, N_3; 2)$, and $(N_1, N_2, N_3; 3)$ where all jobs are scheduled.

State $(N_1, N_2, N_3; i)$ corresponds to the situation where job $a_{i[1]}$ is processed first in the complete schedule, and $V(N_1, N_2, N_3; i)$ is the optimal value of flowtime in this situation. This flowtime does not include the initial setup time s_{0i}, so that recurrence relation (2.24) is required if this setup is carried out.

$$V(N_1, N_2, N_3; 0) = \min_{1 \le i \le 3}\{V(N_1, N_2, N_3; i) + N \cdot s_{ik}\} \tag{2.24}$$

$V(N_1, N_2, N_3; 0)$ is the optimal flowtime when processing starts from a bare machine. If the machine was initially configured for processing class i, then $V(N_1, N_2, N_3; i)$ is the optimal value; while if no initial setup time is required, the optimal value is given by $\min_{1 \le i \le 3}\{V(N_1, N_2, N_3; i)\}$.

Evidently, knowing the optimal flowtime value is not sufficient when the aim is to produce a schedule for a machine. To find the optimal schedule using the DP, one more element is required, this being a function $P(n_1, n_2, n_3; i)$ that gives the job scheduled after the first job $a_{i[N_i - n_i']}$ in the partial schedule. A function of this type is known as an *optimal policy function*. Refer, for example, to relation (2.21) and assume that state (3, 1, 2; 1) "provided" the optimal value for $V(3, 1, 3; 3)$. This implies that $P(3, 1, 3; 3) = a_{1[N_i-2]}$, this being $a_{1[3]}$ for the 5 jobs-per-class example. Thus, the partial schedule S described by (3, 1, 3; 3) has $a_{3[3]}$ first and $a_{1[3]}$ second. Given $P(3, 1, 2; 1)$, we can find the third job in S, and so on. This "backtracking" in order to build the optimal schedule is carried out once the final state has been generated.

As witnessed by the example at the beginning of this section, a dynamic program extends only those partial sequences which are "optimal within themselves" and hence might lead to an optimal solution for the complete schedule. In other words, a dynamic program for a machine scheduling problem operates according to the principle that "the optimal schedule has the property that, whatever the choice of the first job a_j, the remainder of the schedule from a_j onward is the optimal schedule for the problem with a_j excluded" ([25]); this often is called the *principle of optimality*. The solution-building strategy is implied by the form of the recurrence relations that are used in dynamic programs, relation (2.23) being a particular example.

The state definition (2.20) the recurrence relation (2.23) and optimal value function V, the SWPT-within-classes scheduling rule (Theorem 4), and the optimal policy function P are the dynamic program for this example problem. Moreover, this is representative of the composition of a typical dynamic program.

The Ghosh Dynamic Program for $1|s_{ij}|\Sigma wC$

The DP formulation given above for a three-class problem is a special case of the DP proposed by Ghosh for the $1|s_{ik}|\Sigma wC$ problem with an unrestricted number of classes. The recurrence relation for the (backwards-recursive) Ghosh formulation is:

$$V(n_1, n_2, \ldots, n_B; i) = \min_{\{k: n_k' > 0,\quad 1 \le k \le B\}} \{ V(n_1', n_2', \ldots, n_B'; k) + (W + w_{i[N_i - n_i']}) \cdot p_{i[N_i - n_i']} + W \cdot s_{ik} \}$$

where

$$W = \sum_{f=1}^{B} \sum_{j=1}^{n_f'} w_{f[N_f - j + 1]}$$

that is, W is equal to the combined weight of all jobs in the partial sequence represented by state $(n_1', n_2', \ldots, n_B'; k.)$ Recall that $a_{i[N_i - n_i']}$ is the job added to yield $(n_1, n_2, \ldots, n_B; i)$ (i.e., the n_ith-last job of class i). The Ghosh formulation for the $1|s_{ij}|\Sigma wC$ problem is the most recently reported DP for a single machine flowtime problem with sequence-dependent setup times.

The number of states explored by a dynamic program in finding the optimal solution depends primarily on the size of the problem and the state definition. The following straightforward method of determining (a bound upon) the number of states is often applicable; for each state variable x determine the range of possible values r_x, then find the product $\Pi_x r_x$. For the Ghosh DP with state vector $(n_1, n_2, \ldots, n_B; i)$, this approach yields $B \cdot \prod_{k=1}^{B} (N_k + 1)$ states, as i may take values 1,…, B and each n_k ranges from 0 to N_k.

When a dynamic program is running, the state information must be stored. The storage requirements for a dynamic program should be directly proportional to the number of states; thus, for the Ghosh DP, the amount of memory required will be $O(B \ldots \prod_{k=1}^{B} (N_k + 1))$, or $O(BN^B)$. It can be noted that the value of the BN^B function is highly sensitive to changes in the number of classes B; the limitations this imposes will be evident when the experimental study carried out by Ahn and Hyun is discussed shortly.

A running time bound for a dynamic program can be developed using knowledge of the number of states explored and the number of operations required to construct a state from prior states. For the

TABLE 2.14 Mean Execution Times (in Seconds) for the Ahn and Hyun Dynamic Program, Using an HP 9000 Workstation

Number of Classes (B)	Number of Jobs in Each Class (N_i)						
	2	3	4	5	10	20	50
3	0.06	0.13	0.28	0.49	3.24	23.48	401.77
4	0.29	0.98	2.54	5.43	66.51	1025.01	
5	1.40	6.46	21.01	53.82	1318.35		
6	6.40	39.60	161.26	522.02			
7	27.74	229.26	1238.16				
8	116.05	1332.23					

Ghosh DP, calculation of $V(n_1, n_2,\ldots, n_B; i)$ requires $O(B)$ operations, as there are $O(B)$ equations within the min expression, and evaluation of each should be possible in $O(1)$ time. Thus, Ghosh is able to state that the number of computations required for the proposed DP is $O(B^2 \cdot \prod_{k=1}^{B}(N_k + 1))$, or more simply $O(B^2N^B)$.

The running time bound of $O(B^2N^B)$ illustrates that the $1|s_{ij}|\Sigma wC$ problem can be solved in polynomial time if the number of classes is fixed. In practice, we may say that the problem can be efficiently solved if B is "small enough" so that the polynomial in N is not of "too large" a degree.

Dynamic Program Performance

The dynamic program developed by Ahn and Hyun for the $1|s_{ij}|\Sigma C$ problem is very similar to the Ghosh dynamic program, and shares with it both the running time bound of $O(B^2N^B)$ and the storage requirement of $O(BN^B)$. Ahn and Hyun successfully applied their dynamic program to instances containing up to 150 jobs. Their experiment involved the generation of a set of test instances with varying numbers of classes and number of jobs per class. Setup times and processing times were generated from uniform distributions with a range [1, 99], and in every instance each class was assigned the same the number of jobs ($N_{\max}$) as given to other classes (i.e., $N_i = N_{\max}\ \forall i$). The number of classes B was selected from {2,3,…,8}, the number of jobs per class N_i was selected from {2, 3, 4, 5, 10, 20, 50}, and a sample size of 20 instances was used at each tested combination of B and N_i.

The dynamic program's observed mean execution times, as reported by Ahn and Hyun, are shown in Table 2.14. Even for relatively small values of B these results show that, on a execution time basis, the use of the DP becomes increasingly prohibitive as N is increased. This is not unexpected, given the $O(B^2N^B)$ running time bound and the *NP*-hardness of the $1|s_{ij}|\Sigma C$ problem. The $O(BN^B)$ storage requirement for the algorithm is also restrictive, and Ahn and Hyun note that problems with only 8 classes and 4 jobs per class stretch the 12-MB memory capacity of the computer used; the lack of results in the lower right-hand side of the table reflects this.

As demonstrated by Ahn & Hyun's experimental results, the running time and storage requirements of the dynamic program can be prohibitive for even moderate values of number of jobs N. A more efficient method for solving the $1|s_{ij}|\Sigma wC$ problem has yet to be devised.

Ahn and Hyun's Heuristic

Ahn and Hyun [3] present a heuristic designed to provide near-optimal schedules for the $1|s_{ij}|\Sigma C$ problem. This heuristic is in some ways similar to another proposed by Gupta [39] for the problem with two classes. The Ahn and Hyun heuristic passes through a seed sequence, searching within a neighborhood of possible moves for an opportunity to improve the sequence. As such, the heuristic is an example of a local search heuristic, and may be further classified as a descent heuristic.

Passes are made both forwards (from the start of the sequence onward) and backwards (from the end of the sequence). Multiple passes can be undertaken, depending on the *stopping criterion* applied. The procedures for the forwards and backwards passes are quite similar.

In the forwards procedure, an index t corresponds to a position in the current sequence S ($1 \le t \le N$), with t being incremented one-position-at-a-time through the sequence. We will denote the job appearing

at some sequence position x ($1 \leq x \leq N$) as job $a_{(x)}$, and note that setup tasks are implied by the sequence, so that no sequence position is occupied by a setup. The partial sequence preceding job $a_{(t)}$ is labeled as σ, and is considered as the "fixed" part of the schedule. A subsequent partial sequence ρ begins with $a_{(t)}$ and contains at least the remainder of the jobs in the same run as $a_{(t)}$ (let this run be R_p). Following ρ is another partial sequence A, either an entire run or part of a run; *ρ and A will be interchanged if the flowtime of the resulting sequence $\hat{S}$ is less than that of S*. The partial sequence Γ brings up the rear of S, so that the current sequence is $S = \{\sigma\rho A\Gamma\}$ and the alternative sequence is $\hat{S} = \{\sigma A\rho\Gamma\}$.

The rules for the formation of partial sequences ρ and A are formulated in a way that ensures that SPT-ordering within classes (Theorem 4) is preserved. No class of jobs appearing in A can be allowed to appear in ρ, and vice versa—if a class appeared in both ρ and A, interchange of these partial sequences would disturb SPT-order. This immediately explains why ρ contains at least the remainder of the jobs in the same run as $a_{(t)}$. For a given value of τ, the interchange of a number of different ρ and A partial sequences can be investigated. In a manner of speaking, the neighborhood is specified by the range of different ρ and A interchanges able to be searched.

Partial sequence ρ initially contains job $a_{(t)}$ and the jobs following it in run R_p. Partial sequence A begins at sequence position r with some job $a_{(r)}$ and ends at sequence position j (job $a_{(j)}$) where $j \geq r$. Job $a_{(r)}$ will always correspond to the first job of a run, and $a_{(j)}$ belongs to the same run as $a_{(r)}$—let this run be R_A. Initially, $r = j$; that is, the first move investigates the interchange of ρ and $a_{(r)}$.

With indices $\{t, r, j\}$ giving rise to partial sequences $\{\sigma, \rho, A, \Gamma\}$ as shown in Fig. 2.17, the first comparison of $S = \{\sigma\rho A\Gamma\}$ with alternative sequence $\hat{S} = \{\sigma A\rho\Gamma\}$ is carried out. If the flowtime of $\hat{S}$, $F(\hat{S})$, is less than the flowtime $F(S)$ of S, $\hat{S}$ replaces S, index t is not incremented, and the search for improvement at this value of t is repeated. Otherwise, index j is incremented by one, expanding partial sequence A (see Fig. 2.18), and a new flowtime comparison made.

When the indexing of j causes $a_{(j)}$ to the be the first job of a run, jobs $a_{(r)}$ and $a_{(j)}$ belong to different runs. This is not allowed; thus, the partial sequence ρ expands to include the jobs up until $a_{(j-1)}$, index r is updated to $r = j$, and A is redefined so that it contains only $a_{(r)}$ (Fig. 2.19).

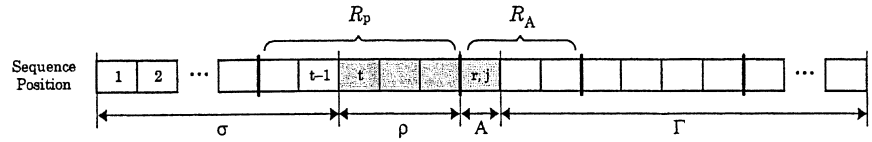


FIGURE 2.17

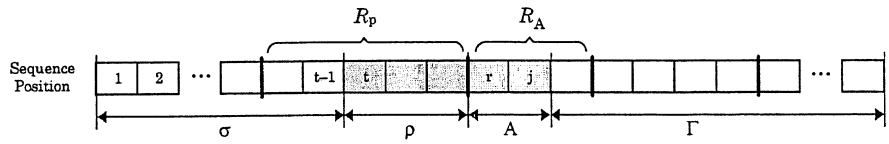


FIGURE 2.18

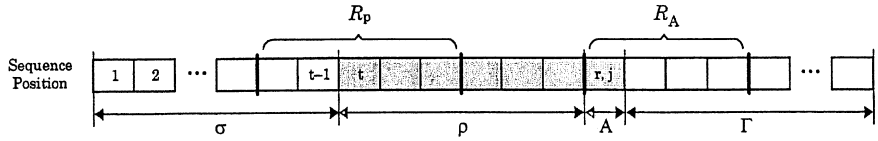


FIGURE 2.19

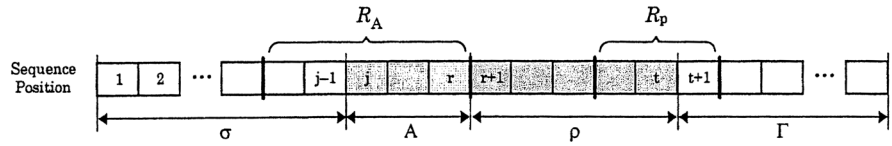


FIGURE 2.20 Sequence partitioning and Labeling for the backwards pass of the Ahn and Hyun heuristic.

If j reaches the value $N + 1$, there are no interchanges (in the neighborhood) that can improve the flowtime for this value of t. Index t is incremented by one in this case, and the search recommences.

The primary difference between the forwards and backwards passes is that for a backwards pass, index t decreases as the pass continues and, likewise, index j takes progressively lower values during a search for improvement. A block interchange for the backwards procedure can be written as $S = \{\sigma A \rho \Gamma\} \rightarrow \hat{S} = \{\sigma \rho A \Gamma\}$. With reference to Fig. 2.20, partial sequence A again corresponds to all or part of a run R_A, this time with job $a_{(r)}$ as the last job in the run and job $a_{(j)}$ earlier in R_A. Partial sequence $\rho = \{a_{(r+1)},\ldots,a_{(t)}\}$ includes job $a_{(r+1)}$ from the beginning of the run R_{A+1} following R_A, and unlike A may span more than one run. In the forwards procedure, σ is the 'fixed' part of the sequence and the flowtime of jobs in σ is not affected by a block-interchange. In comparison, Γ is the fixed part of the sequence in the backwards procedure, yet σ remains as the partial sequence of jobs that is not affected by a given block interchange.

Ahn and Hyun state a $O(B^2 N_{\max}^2)$ running-time bound for one iteration (i.e., one pass) of their heuristic, where B is the number of classes and $N_{\max}$ the maximum number of jobs in any one class. Ahn and Hyun do not provide a derivation for this running-time bound, and our own attempts at reproducing this running-time result have not been completely successful. However, if we assume that the maximum number of block-interchanges able to be undertaken at any value of t is bounded by $O(N)$, we are able to derive the bound given by Ahn and Hyun. Using this assumption, it is also possible to establish an alternative expression for the running time bound, $O(BN^2)$. Both forwards and backwards passes of the heuristic share this running-time bound; thus, for a fixed number of iterations of the heuristic, the overall running-time is $O(BN^2)$.

Although Ahn and Hyun address the total flowtime problem, their heuristic is easily extended to operate on the $1|s_{ij}|\Sigma wC$ problem, by utilizing Monma and Potts' SWPT-within-classes rule and restricting setup times to those satisfying the triangle inequality. Evaluation of the weighted flowtime form of the Ahn and Hyun heuristic, albeit with sequence-independent setup times, has been carried out by Crauwels, Potts, and Van Wassenhove [23] (see Section 2.3).

Seed Schedules, Stopping Criteria, and Experimental Performance

Ahn and Hyun's heuristic requires a "seed" sequence to be generated, this used as an input to the heuristic. We assume that Ahn and Hyun use Gupta's greedy heuristic to generate seeds; however, they do not state this, and neither do they report on investigations that may have led them to choose one seed generator over another.

Application of the Ahn and Hyun heuristic requires the imposition of a stopping criterion to determine the point at which searching should terminate. One stopping criterion proposed by Ahn and Hyun enables the heuristic to continue until no further improvement is found. An alternative stopping criterion investigated by Ahn and Hyun limits the heuristic to one forwards and one backwards pass. These can be referred to as the *unlimited-cycles* and *one-cycle* stopping criteria, respectively.

In the worst possible case, the unlimited-cycles stopping criterion can result in a sequence being generated for every flowtime value between that of the initial seed schedule to that of a final optimal solution. This leads to exponential-time worst-case behavior of the heuristic, unless all data are integers and have a magnitude bounded by some constant. In practice, the average-case behavior of the heuristic using this stopping criterion may not be so poor, yet there is no guarantee. Given that one pass of the sequence takes $O(BN^2)$ time, any stopping criterion that imposes a fixed maximum number of cycles results in polynomial-time behavior of the heuristic.

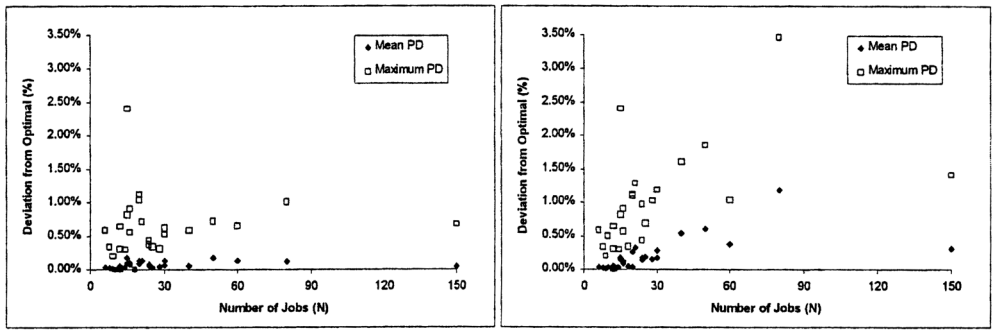


FIGURE 2.21 Mean and maximum deviations from optimal for the Ahn and Hyun heuristic; stopping criteria of unlimited cycles (left) and one cycle (right).

We have already presented some of the results of computational experiments reported by Ahn and Hyun, namely those relating to their dynamic program. When investigating the performance of the heuristic, Ahn and Hyun use the same instances and test their heuristic against (1) optimal solutions given by their dynamic program, and (2) solutions obtained using Gupta's greedy heuristic.

The results reported by Ahn and Hyun show that for the unlimited-cycles case, there is little difference between the flowtime achieved by the heuristic and the optimal flowtime. For 21 out of the 27 tested combinations of B and N_i, schedules produced by their heuristic have a flowtime within 0.10% of the optimal flowtime on average (deviations from optimal are calculated using $(F^{Huer} - F^{Opt})/F^{Opt}$). The worst observed mean deviation from optimal was 0.17% for this variant of the heuristic. Maximum percentage deviations are also low; the maximum deviation from the optimal solution was typically less than 1% for each set of test instances, the worst performance being 2.4% attained for an instance of $B = 5$, $N_i = 10$. Plots of mean and maximum percentage deviation results, for both unlimited-cycle and one-cycle termination criteria, are given in Fig. 2.21

The plot for the unlimited-cycles case shows a very slight relationship between the total number of jobs N and the deviations from the optimal solution—statistically, the correlation is poor (product-moment correlation coefficient of +0.13). This substantiates Ahn and Hyun's claim that the heuristic performance does not deteriorate when either B or N increases. For the one-cycle stopping criterion, very good results are also obtained. On average, the first iteration of the heuristic typically approaches well within 1% of the optimal solution. With this stopping criterion, it is apparent that increases in problem size do lead to an increase in both mean and maximum deviations from optimal, although the relationship is not strong.

The performance of Gupta's greedy heuristic does not compare favorably with that of the Ahn and Hyun heuristic. The mean deviation from optimal for the Gupta heuristic was between 3.5% ($B = 7$, $N_i = 3$) and 7.3% ($B = 3$, $N_i = 50$) for problem sizes of 20 jobs or more. Such deviations are well in excess of the 0.17% maximum mean percentage deviation (over all (B, N_i) combinations) attained by the Ahn and Hyun heuristic utilizing an unlimited-cycles stopping criterion.

We believe that Ahn and Hyun provide the *average time per iteration* and the *average number of iterations* for their heuristic (unlimited cycles, at each tested (B, N_i) combination). Thus, the observed average computation time (ACT) is roughly equal to the product of these two numbers (yet $E(x) = E(x/y)\ E(y)$ only if y does not vary). The average number of iterations is typically between 2 and 3, one iteration being one cycle as described above, although for problems with 40 or more jobs, this rises to a maximum average of 4.7 iterations for $B = 3$, $N_i = 50$.

Taking Ahn and Hyun's tabulated results for tests where $N = BN_i \leq 40$, estimates of ACT have been calculated and the data points plotted in Fig. 2.22. This displays both the effect of the number of classes B and of the total number of jobs N; naturally, an increase in either of these leads to an increase in ACT. A regression involving all 27 data points, of log (ACT) with log (N), gives $ACT \propto N^{2.26}$ ($r^2 = 0.987$). Interestingly, this function compares well to the $O(BN^2)$ running-time bound.

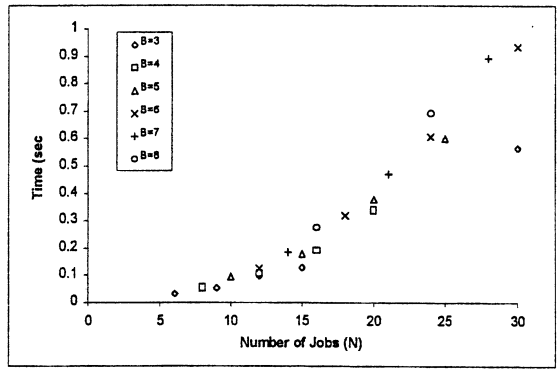


FIGURE 2.22 Plot of "approximate" ACT data for the Ahn and Hyun heuristic (unlimited-cycles stopping criterion).

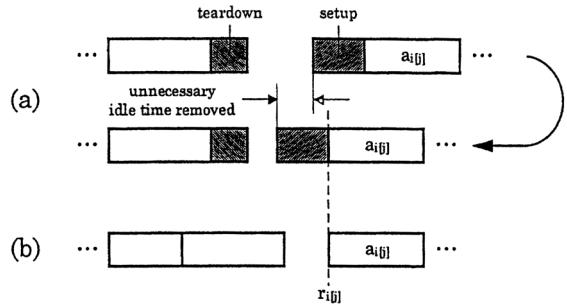


FIGURE 2.23 Idle time in regular class scheduling problems: (a) between runs, where elimination of unnecessary idle time is possible, and (b) within a run, where no unnecessary idle time is scheduled.

Additive Changeovers on a Single Machine

Although many researchers have addressed single machine problems with sequence-independent setup times, when additive changeovers on single machines are considered, we are only aware of the work of Mason and Anderson [59]. In this section we address single machine problems with additive changeovers and regular objectives, including those with release dates $r_{i[j]}$, due dates $d_{i[j]}$, and deadlines $\bar{d}_{i[j]}$ for each job $a_{i[j]}$. The class scheduling problems such as these, incorporating regular objective functions, can be termed *regular class scheduling problems*.

We will show that regular class scheduling problems with additive changeovers can be transformed and solved as problems with sequence-independent setup times only. This extension of a result developed by Mason and Anderson for the $1|s_+|\Sigma wC$ problem will allow many other problems with additive changeovers to be solved using algorithms devised for problems with sequence-independent setup times. The result significantly enhances the practical applicability of these algorithms.

The *class scheduling assumptions* listed in Section 2.2 include the assumption that inserted idle time is permitted in class scheduling problems, yet inserted idle time will appear in optimal ("reasonable") schedules for a problem with a regular objective function only when job arrivals are dynamic (i.e., where jobs have unequal ready times). Additionally, it is clear that a schedule for a dynamic problem can be improved by reduction or elimination of inserted idle time if a job commences at some instant after its ready time and idle time is scheduled between it and the preceding job in the sequence. This applies regardless of the class of the preceding job; that is, setups or teardowns can be scheduled between these jobs (Fig. 2.23). Thus, we can use the following scheduling rule for regular class scheduling problems.

- For a regular class scheduling problem with dynamic job arrivals, idle time can only be inserted between a pair of jobs if the latter job commences exactly at its ready time.

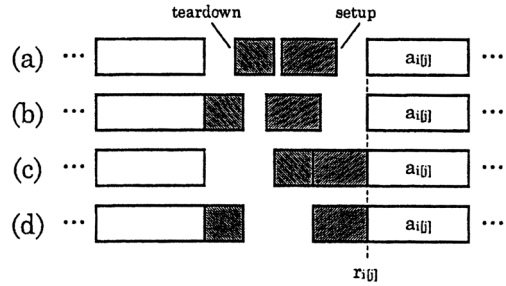


FIGURE 2.24 STS rule for problems with regular objectives: only schedule (d) satisfies the rule.

In the special case where all jobs are released simultaneously (i.e., at the *class ready time* r_i), this rule will prohibit idle time being scheduled between jobs of the same class, so that no idle time will appear within runs.

There is flexibility in the timing of the teardown and setup tasks when a job is scheduled to begin processing at its ready time, is the first job of a run, and the time available for the teardown and setup between this job and the preceding job is more than that required. Such flexibility may be welcome in practice; when the next-sequenced job might arrive early, the teardown and setup should be carried out as soon as practicable; while if there is the possibility that scheduling decisions may be changed, to cater for "hot jobs," for example, setups might be carried out at the latest opportunity. However, for the analysis of scheduling problems, it is beneficial to impose a rule relating to the scheduling of setups and teardowns in such situations. We term this rule the *setup and teardown scheduling rule for problems with additive changeovers*, or more simply, the *STS rule*.

- The STS rule states that setups must be scheduled so that they are immediately followed by the first job of a run, and teardowns must commence as soon as the last job in the run ends (Fig. 2.24).

Any schedule that does not follow this STS rule can be modified to produce a schedule that does, without changing the sequence of jobs or increasing the value of a regular objective function. Likewise, schedules that do follow the STS rule can be modified in practice without affecting the quality of the schedule.

In their analysis, Mason and Anderson conclude that for any instance P of the static single machine flowtime problem with additive changeovers and initial setups $(1|s_+|\Sigma wC)$, there is an *equivalent instance* P' with sequence-independent setup times only $(1|s_i|\Sigma wC)$. The equivalent instance P' is identical to P except that the setup and teardown times s_i and t_i of instance P are replaced by a setup time s_i' in P', with $s_i' = s_i + t_i$.

An optimal sequence for P' is also an optimal sequence for P, and a "good" sequence for P' is an equally "good" sequence for P. Consequently, it is both convenient and reasonable to solve instances incorporating additive changeovers by (1) altering the setup times according to $s_i' = s_i + t_i$, and (2) solving the instance of the $1|s_i|\Sigma wC$ problem that results. An optimal algorithm for solving P' will provide an optimal sequence for P, while a heuristic developed for P' and known to provide schedules of a certain (expected) quality can be used to provide schedules of the same (expected) quality for instances of P. It may be said that the $1|s_+|\Sigma wC$ and $1|s_i|\Sigma wC$ problems are *equivalent problems*.

The reduction of an instance of $1|s_+|\Sigma wC$ to an instance of $1|s_i|\Sigma wC$ is a special case. Mason and Anderson show that two instances of $1|s_+|\Sigma wC$, P and P', are equivalent if setup and teardown times satisfy $s_i' + t_i' = s_i + t_i$ for all classes i, and P and P' are otherwise identical (in particular, each job in P has a unique and exact equivalent in P', and vice versa). The equivalence is established by noting that application of some given sequence to both instances yields weighted flowtimes F_W (for P) and F_W' (for P') that satisfy

$$F_W' - F_W = \sum_{i=1}^{B} (s_i' - s_i) \sum_{j=1}^{N_i} w_{i[j]}$$

This difference takes a value which is independent of the schedule, so that minimization of F'_W is equivalent to minimization of F_W.

An essential element of Mason and Anderson's proof that an instance P of $1|s_+|\Sigma wC$ can be successfully transformed into an instance P' of $1|s_i|\Sigma wC$ is the property that the setup for each run begins at the same time for both instances when the same sequence is applied. This leads to the observation that the completion times $C_{i[j]}$ and $C'_{i[j]}$ of a job $a_{i[j]}$ (in P and P', respectively) satisfy the relationship $C'_{i[j]} - C_{i[j]} = s'_i - s_i$. This follows from the *transformation rule* $s'_i + t'_i = s_i + t_i$. When generalizing Mason and Anderson's result to more complicated scheduling problems, additional transformation rules will be necessary.

Consider a problem with arbitrary job release dates $r_{i[j]}$. If the processing of some job $a_{i[j]}$, which is scheduled to begin a run, commences as soon as it is released (at time $r_{i[j]}$), the setup for this run will commence at some time $T = r_{i[j]} - s_i$ in P (by the STS rule). When P is transformed into P', to allow this run to once again commence at time T requires $r'_{i[j]} - s'_i = r_{i[j]} - s_i$, thus indicating that ready times will also have to be modified in transformations from P to P' in more complicated problems. Similarly, due dates and deadlines also should be modified.

Theorem 5 provides the set of transformation rules necessary for the relationship $C'_{i[j]} - C_{i[j]} = s'_i - s_i$ to be satisfied by the completion times of all jobs in any regular class scheduling problem with ready times, due dates, and/or deadlines. The $C'_{i[j]} - C_{i[j]} - s'_i - s_i$ relationship is then combined with the transformation rules to show that (1) for every job, both the flowtime (completion time minus release date) and lateness (completion time minus due date) take the same values in both P and P'; and (2) a schedule will meet deadlines in P if and only if they are met in P'.

The notation used in the statement of Theorem 5 includes the "additional" notation given in Table 2.12 (Section 2.3). Symbols will appear primed to indicate values for instance P' with sequence-independent setups only (e.g., σ'_k is the setup for R_k in P'), and unprimed when associated with original instance P.

Theorem 5

Consider an instance P of a single machine regular class scheduling problem in which the class setup times and teardown times are given by s_i and t_i, and job ready times, due dates, and deadlines are given $r_{i[j]}$, $d_{i[j]}$, and $\bar{d}_{ij}$, respectively. Initial setup times are required, and are equal to s_i for all i. Let P' be a second instance of the same problem, P' identical to P except that these times are given by s'_i, t'_i, $t'_{i[j]}$, $d'_{i[j]}$, and $\bar{d}'_{i[j]}$. Without loss of generality, assume that all schedules satisfy the STS rule, and let the following be satisfied by instances P and P' for all i:$1 \le i \le B$ and j:$1 \le j \le N_i$.

$$s'_i + t'_i = s_i + t_i \tag{2.25}$$

$$r'_{i[j]} - s'_i = r_{i[j]} - s_i \tag{2.26}$$

$$d'_{i[j]} - s'_i = d_{i[j]} - s_i \tag{2.27}$$

$$\bar{d}'_{i[j]} - s'_i = \bar{d}_{i[j]} - s_i \tag{2.29}$$

When a given sequence of jobs S is applied to instances P and P', the completion times $C_{i[j]}$ and $C'_{i[j]}$ of any job $a_{i[j]}$ (in P and P', respectively) are related by:

$$C'_{i[j]} - C_{i[j]} = s'_i - s_i, \tag{2.29}$$

The flowtime and lateness of any job are identical in P and P', that is:

$$C'_{i[j]} - r'_{i[j]} = C_{i[j]} - r_{i[j]} \tag{2.30}$$

$$C'_{i[j]} - d'_{i[j]} = C_{i[j]} - d_{i[j]} \tag{2.31}$$

and a deadline is satisfied in P' if and only if it is satisfied in P.

Proof
By the STS rule, the completion time of the first job $\alpha_{k[1]}$ in run R_k is given by:

$$c_{k[1]} = B_k + \sigma_k + \tau_{k[1]} \tag{2.32}$$

$$c'_{k[1]} = B'_k + \sigma'_k + \tau_{k[1]} \tag{2.33}$$

where B_k is the starting time of run R_k, while the completion times of other jobs $\alpha_{k[j]}$ $(1 < j \leq \lambda_k)$ in the run will be given by:

$$c_{k[j]} = \max\{\hat{r}_{k[j]}, c_{k[j-1]}\} + \tau_{k[j]} \tag{2.34}$$

$$c'_{k[j]} = \max\{\hat{r}_{k[j]}, c'_{k[j-1]}\} + \tau_{k[j]} \tag{2.35}$$

where $\hat{r}_{k[j]}$ is the ready time of job $\alpha_{k[j]}$ in instance I. Using Eq. (2.32) and (2.33), it is found that for any $1 \leq k \leq r$,

$$B'_k = B_k \quad \text{implies} \quad c'_{k[1]} - c_{k[1]} = \sigma'_k - \sigma_k \tag{2.36}$$

Assume that

$$c'_{k[j-1]} - c_{k[j-1]} = \sigma'_k - \sigma_k \tag{2.37}$$

is satisfied for some $2 \leq j \leq \lambda_k$. Substitution of Eq. (2.26) and (2.37) into Eq. (2.35) yields

$$c'_{k[j]} = \max\{\hat{r}'_{k[j]}, c'_{k[j-1]}\} + \tau_{k[j]} = \sigma'_k - \sigma'_k + c_{k[j]}$$

so that for $2 \leq j \leq \lambda_k$,

$$c'_{k[j-1]} - c_{k[j-1]} = \sigma'_k - \sigma_k \quad \text{implies} \quad c'_{k[j]} - c_{k[j]} = \sigma'_k - \sigma_k \tag{2.38}$$

Together, Eq. (2.36) and (2.38) provide the result that, for all jobs in a run $(\alpha_{k[j]}{:}1 \leq j \leq \lambda_k)$,

$$B'_k = B_k \quad \text{implies} \quad c'_{k[j]} - c_{k[j]} = \sigma'_k - \sigma_k \tag{2.39}$$

The starting times B_1 and B'_1 of the first run in the schedule are given by:

$$B_1 = \max\{0, \hat{r}_{1[1]} - \sigma_1\}$$
$$B'_1 = \max\{0, \hat{r}'_{1[1]} - \sigma'_k\}$$

which can be used with Eq. (2.26) to show $B_1 = B'_1$. For later runs R_k $(2 \leq k \leq r)$,

$$B_k = \max\{c_{k-1[\lambda_k]} + \delta_{k-1}, \hat{r}_{k[1]} - \sigma_k\} \tag{2.40}$$

$$B'_k = \max\{c'_{k-1[\lambda_k]} + \delta'_{k-1}, \hat{r}'_{k[1]} - \sigma'_k\} \tag{2.41}$$

where δ_{k-1} and δ'_{k-1} are the teardown times of run R_{k-1} in instances P and P', respectively. If it is assumed that $B'_{k-1} = B_{k-1}$, then $c'_{k-1[\lambda_{k-1}]} - c_{k-1[\lambda_{k-1}]} = \sigma'_{k-1} - \sigma_{k-1}$ by Eq. (2.39); substitution of this,

Eq. (2.25) and (2.26) into (2.40) and (2.41), yields Eq. (2.42).

$$\begin{aligned} B'_k &= \max\{c'_{k-1[\lambda_k]} + \delta_{k-[1]}, \hat{r}'_{k[1]} - \sigma'_k\} \\ &= \max\{c_{k-1[\lambda_{k-1}]} + \sigma'_{k-1} - \sigma_{k-1} + \sigma'_{k-1}, \hat{r}_{k[1]} - \sigma_k\} \\ &= \max\{c_{k-1[\lambda_{k-1}]} + \delta_{k-1}, \hat{r}_{k[1]} - \sigma_k\} \\ &= B_k \end{aligned} \tag{2.42}$$

Thus, $B_{k-1} = B'_{k-1}$ implies $B_k = B'_k$, so that by mathematical induction it is shown that $B_k = B'_k$ for all runs R_k $(1 \le k \le r)$ as $B_1 = B'_1$. Combining this result with Eq. (2.39) gives:

$$c'_{k[j]} - c_{k[j]} = \sigma'_k - \sigma_k \quad \text{for all} \quad 1 \le k \le r, 1 \le j \le \lambda_k$$

so that (2.29) holds for all jobs.

Expressions (2.30) and (2.31) for the flowtime and lateness of a job are derived by combining Eq. (2.29) with Eq. (2.25) to (2.27). Similarly, substitution of Eq. (2.28) into Eq. (2.29) gives:

$$C'_{i[j]} - \bar{d}'_{i[j]} = C_{i[j]} - \bar{d}_{i[j]}$$

for deadlines. As the difference between the completion time and deadline of a job is the same for both instance P and instance P', a job cannot meet a deadline in one instance and fail to meet it in the other.

Theorem 5 can be used to show that many problems with additive changeovers can be solved as problems with sequence-independent setup times. To solve P using an algorithm for a single machine problem with sequence-independent setup times, it is usually sufficient to ensure that:

1. Equations (2.25) to (2.28) are satisfied by an equivalent instance P'; that is, instance P' is constructed from P according to these relationships.
2. The objective function is regular, so that the 'idle time insertion' rule and the STS rule are valid.
3. In the objective function, job completion times only appear when grouped with due dates as job lateness values $T_j = C_j - d_j$, or grouped with ready times as job flowtime values $F_j = C_j - r_j$.
4. Constraints are satisfied in P if and only if their equivalents are satisfied in P'.

In relation to item 4 above, it has been illustrated that this requirement is satisfied by deadline constraints once they are modified for use in P'. Similar analysis would be required for additionally imposed constraints.

Table 2.15 provides some examples of problems with additive changeovers that can be solved by transforming the original instance and solving it as an instance without teardowns. Clearly, we cannot provide an exhaustive list of the problems amenable to this treatment.

It is interesting to note that Mason and Anderson's original $1|s_+|\Sigma wC$ problem does not satisfy item 3 above, as job completion times are not "grouped" with due date or ready times in the objective. However, $1|s_+|\Sigma wC$ can be solved as a $1|s_i|\Sigma wC$ problem due to the equivalence described in Section 2.2 between the objectives of (1) sum of job completion times ΣwC, (2) sum of job lateness $\Sigma w(C - d)$, (3) sum of job flowtimes $\Sigma w(C - r)$, and (4) sum of job waiting times $\Sigma w(C - p - r)$. This equivalence extends the scope of Theorem 5.

TABLE 2.15 Single Machine Problems that are Solvable as Problems with Sequence-Independent Setup Times

Number of late jobs	$1\|s_+\|\Sigma wU$
Maximum lateness with deadlines	$1\|\bar{d}_j, s_+\|L_{\max}$
Tardiness with deadlines and release dates	$1\|r_j, \bar{d}_j, s_+\|\Sigma wT$
Flowtime plus maximum tardiness	$1\|s_+\|\Sigma wF + T_{\max}$

The transformation rules of Theorem 5 will ordinarily specify non-zero ready times for jobs, even if the original problem is a static problem with $r_{i[j]} = 0 \ \forall i, j$. This initially alarming result does not lead to the modified problem becoming a "truly" dynamic problem, however. A problem need only be considered dynamic if the scheduling of jobs can actually be constrained by ready times, that is, if there exists at least one job $a_{i[j]}$ with a release date $r_{i[j]}$ greater than the setup time s_i for its class. Equation (2.26) can be used to show that $r_i \leq s_i$ implies $r'_i \leq s'_i$, so that both instances will be static if the original instance is.

Although Theorem 5 deals with problems with initial setup times s_{0i} equal to class setup times s_i, problems without initial setups or with $s_{0i} \neq s_i$ can be handled simply. We will assume that all setup and teardown times must be non-negative; note that this is consistent with the view that the zero point in time corresponds to the earliest instant at which scheduling decisions can be effected. Let $\gamma_i = s_{0i} - s_i$, calculate γ_i for all classes i, and let $\gamma_{\min} = \min_{1 \leq i \leq B}\{\gamma_i\}$ and $\gamma_{\max} = \max_{1 \leq i \leq B}\{\gamma_i\}$. If $\gamma_{\min} \geq 0$, all initial setup times are no more than class setup times. In this case, we can modify initial setup times according to $s'_{0i} + t'_i = s_{0i} + t_i$, or equivalently $s'_{0i} - s'_i = s_{0i} - s_i$, and apply Eq. (2.25) to (2.28) as above in order to create an equivalent problem. The fact that $\gamma_{\min} \geq 0$ ensures that all initial setup times in P' are non-negative. Clearly, setting $t'_i = 0$ for all classes removes teardown times, this being our primary interest here. Alternatively, $\gamma_{\min} < 0$, and utilization of $s'_{0i} + t'_i = s_{0i} + t_i$ can lead to initial setup times that are less than zero. However, if we simply seek to remove teardowns when creating P', $s'_i \geq s_i$ for all classes so that $s'_{0i} \geq 0$ for all i.

Interestingly, the solution of problems with additive changeovers as problems with sequence-independent setup times is not possible with the $C_{\max}$ objective. The is due to the $C'_{i[j]} - C_{i[j]} = s'_i - s_i$ expression providing unequal change in completion times for jobs belonging to different classes. More importantly, Theorem 5 deals only with regular measures. Scope exists, however, for development of similar results for non-regular measures such the earliness or earliness/tardiness objectives.

To conclude this section, we introduce another result attributable to Mason & Anderson, this result being loosely related to the above analysis. Mason and Anderson show that an instance of the $1|s_{ij}, s_{0i} = 0, B = 2|\Sigma wC$ problem can be solved as an instance of $1|s_i, B = 2|\Sigma wC$. A sequence-dependent setup time s_{ik} from class i to k can be viewed as a sequence-independent teardown time for class i, and the $s'_i + t'_i = s_i + t_i$ rule can then be applied to develop an instance of $1|s_i, B = 2|\Sigma wC$. Theorem 5 can be utilized to extend this result; any problem with two classes, sequence-dependent setup times, no initial setups, and an objective relevant to Theorem 5 can be solved as a problem with sequence-independent setup times. If initial setups are non-zero, this transformation is not valid however.

Flowtime Problems with Sequence-Independent Setup Times

The static single machine flowtime problem with sequence-independent setup times ($1|s_i|\Sigma wC$) has been studied by a number of researchers, and a range of solution methodologies for combinatorial problems have been applied to it. We will continue our review of solution methodologies by looking at the branch-and-bound algorithms for the $1|s_i|\Sigma wC$ problem (as reported by Mason and Anderson [59] and Crauwels, Hariri, Potts, and Van Wassenhove [22]) and particular neighborhood search approaches (Mason [58] and Crauwels, Potts, and Van Wassenhove [23]). These algorithms will be investigated after the structure of the $1|s_i|\Sigma wC$ problem is surveyed.

The SWPT-within-classes rule developed by Monma and Potts for the $1|s_{ij}|\Sigma wC$ problem is clearly an optimality condition for the simpler $1|s_i|\Sigma wC$ problem. We will therefore assume without loss of generality that (1) jobs are indexed within their classes so that $p_{i[j]}/w_{i[j]} \leq p_{i[j+1]}/w_{i[j+1]}$ ($1 \leq i \leq B$, $1 \leq j < N_i$), and (2) that jobs will always be sequenced in this index order, so that $a_{i[j]}$ will always precede $a_{i[j+1]}$.

Mason and Anderson [59] establish two additional optimality conditions for the $1|s_i|\Sigma wC$ problem. These prove to be extremely useful in the development of algorithms for the $1|s_i|\Sigma wC$ problem.

For any run, the *weighted mean processing time (WMPT)* is given by the total time of the run (sum of processing and setup times) divided by the total weight in the run (sum of job weights), so that the

WMPT of a run R_k can be written as:

$$WMPT(R_k) = \frac{T_k}{W_k} = \frac{\sigma_k + \sum_{j=1}^{\lambda_i} \tau_{k[j]}}{\sum_{j=1}^{\lambda_i} \varpi_{k[j]}}$$

Mason and Anderson show that there exists an optimal sequence for the $1|s_i|\Sigma wC$ problem that has runs appearing in non-decreasing order of *WMPT*, that is, in *shortest weighted mean processing time order* (*SWMPT*).

Theorem 6 (Mason and Anderson)

In an optimal sequence for the $1|s_i|\Sigma wC$ problem, runs appear in shortest weighted mean processing time order, that is:

$$WMPT(R_k) = \frac{T_k}{W_k} \leq \frac{T_{k+1}}{W_{k+1}} = WMPT(R_{k+1})$$

for $1 \leq k < r$.

Proof

Consider two runs R_u and R_{u+1}, adjacent in a schedule and not in *SWMPT* order; that is, $WMPT(R_u) > WMPT(R_{u+1})$. Interchanging the positions of these two runs will not increase the weighted flowtime of jobs in other runs, while each job in R_{u+1} has its completion time reduced by at least T_u and each job in R_u has its completion time increased by at most T_{u+1}. Thus, the change in weighted flowtime is given by:

$$\Delta F_W \leq W_u T_{u+1} - W_{u+1} T_u$$

and as $WMPT(R_u) > WMPT(R_{u+1})$, it follows that $W_{u+1}T_u > W_u T_{u+1}$ and $\Delta F_W < 0$, this representing an improvement in the flowtime of the schedule. Consequently, a schedule that does not satisfy the SWMPT rule can be improved, and so is suboptimal.

The third optimality condition for the $1|s_i|\Sigma wC$ problem, which complements the SWPT-within-classes rule for jobs and the SWMPT rule for runs, describes a necessary relationship between the last job $\alpha_{u[\lambda_u]}$ of a run R_u and the first job $\alpha_{v[1]}$ of a following run R_v of the same class.

Theorem 7 (Mason and Anderson)

In an optimal sequence for the $1|s_i|\Sigma wC$ problem, jobs belonging to different runs of the same class satisfy

$$\frac{\tau_{u[\lambda_u]}}{\varpi_{u[\lambda_u]}} \leq \frac{\sigma_v + \sum_{k=u+1}^{v-1} T_k}{\sum_{k=u+1}^{v-1} W_k} \leq \frac{\tau_{v[1]}}{\varpi_{v[1]}} \tag{2.43}$$

where R_u and R_v are runs of the same class ($1 \leq u < v \leq r$).

Proof

Let π be the partial sequence $\{R_{u+1},\ldots,R_{v-1}\}$, that is, the sequence between jobs $\alpha_{u[\lambda_u]}$ and $\alpha_{v[1]}$, and let $T = \sigma_v + \sum_{k=u+1}^{v-1} T_k$ and $W = \sum_{k=u+1}^{v-1} W_k$, so that T is the total time scheduled between $\alpha_{u[\lambda_u]}$ and $\alpha_{v[1]}$, and W is the combined weight of jobs in π. If the left-hand-side of Eq. (2.43) is not satisfied, so that $\tau_{u[\lambda_u]} / \varpi_{u[\lambda_u]} > T/W$, we can interchange the positions of $\alpha_{u[\lambda_u]}$ and π to achieve a change in flowtime of at least

$$\Delta F_w = \varpi_{u[\lambda_u]} T - \tau_{u[\lambda_u]} W$$

which is negative and thus represents an improvement in the schedule's flowtime. If run R_u consists only of job $\alpha_{u[\lambda_u]}$, further improvement is obtained when the setup σ_u is removed from the schedule.

Where S does not satisfy the right-hand inequality, the positions of π and $\alpha_{v[1]}$ can be interchanged to improve the flowtime. The change in flowtime generated by this interchange is given by:

$$\Delta F_W = \varpi_{u[1]} T - \tau_{v[1]} W$$

which is negative when the right-hand inequality of Eq. (2.43) is not satisfied. In the case where $\lambda_v = 1$ and $v < r$ (i.e., R_v is not the last run), the flowtime is decreased by more than ΔF_W as a result of the interchange.

By viewing the work sequenced between any two runs R_u and R_v of the same class as a composite job of processing time $\sigma_v + \sum_{k=u+1}^{v-1} T_k$ and weight $\sum_{k=u+1}^{v-1} W_k$, Theorem 7 represents an *extended SWPT rule*; the jobs belonging to a given class and the total work sequenced between each pair of these jobs must follow SWPT order.

Lower Bounds

Sahney [72], Mason and Anderson [59], and Crauwels, Hariri, Potts, and Van Wassenhove [22] use lower bounds within branch and bound algorithms for the $1|s_i|\Sigma wC$ problem. Sahney's problem can be viewed as a $1|s_i|\Sigma wC$ problem with two classes, and the lower bound used by Sahney is generalized by Mason and Anderson; thus, only the latter will be discussed here.

Given an initial and predetermined partial sequence of scheduled jobs S, Mason and Anderson's procedure generates a lower bound for the flowtime of the best complete schedule formed by appending a partial sequence of currently unscheduled jobs to S. The flowtime of jobs belonging to S can be calculated directly, so that it is the determination of a lower bound on the flowtime of currently unscheduled jobs which is of interest.

The completion time of a job $a_{i[j]}$ in any feasible sequence for a static flowtime problem with setup times can be written as:

$$C_{i[j]} = C^S_{i[j]} + C^P_{i[j]}$$

where $C^S_{i[j]}$ is the contribution due to changeovers and $C^P_{i[j]}$ that due to the processing times of jobs. Given a sequence, these contributions can be easily calculated for each job. The weighted flowtime of the sequence can be written as the sum of these contributions, $F_W = F^S_W + F^P_W$. For example, we can calculate F_W from F^S_W and F^P_W for the schedule shown in Fig. 2.25 (where all weights equal one).

$$\begin{aligned} F_W &= F^S_W + F^P_W = (C^S_{1[1]} + C^S_{1[2]} + C^S_{2[1]}) + (C^P_{1[1]} + C^P_{1[2]} + C^P_{2[1]}) \\ &= (2 + 2 + 5) + (2 + 6 + 11) = 28 \end{aligned}$$

For any sequence, the value taken by the processing-times contribution F^P_W does not depend on the number or durations of setup times. Likewise, the contribution F^S_W due to changeovers does not depend on the processing times of jobs. This allows processing times to be considered as zero when calculating F^S_W, and setup times to be taken as zero when finding F^P_W.

A valid lower bound for a complete schedule for an instance of the $1|s_i|\Sigma wC$ problem can be established by finding lower bounds LB^S_W and LB^P_W for F^S_W and F^P_W, respectively, for jobs yet to be scheduled. A lower bound for the F^P_W contribution can be found by ignoring setup times for unscheduled jobs and sequencing these jobs in non-decreasing order of p/w, that is, in *shortest weighted processing time* (*SWPT*) order. LB^S_W can be established by (1) considering each class to be a *composite job* with processing time s_i and a weight w_i equal to the sum of weights of unscheduled jobs of that class, and (2) sequencing these classes in non-decreasing order of s_i/w_i. The flowtime of this sequence yields LB^S_W.

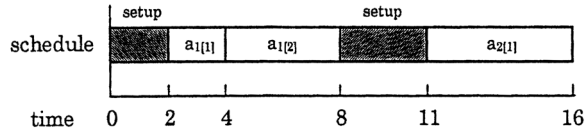


FIGURE 2.25 Example schedule for calculation of flowtime contributions F^S_W and F^P_W.

Where a partial schedule exists, the class of the currently last-sequenced job is excluded from calculation of LB_W^S. Taking account for a predetermined partial sequence also requires the incorporation of the latest known completion time into the completion time of each unscheduled job.

To calculate Mason and Anderson's lower bound requires $O(N \log N)$ time, this being the effort required to sort jobs into SWPT order (sorting of setup times by s_i/w_i takes $O(B \log B)$ time, and calculation of LB_W^S and LB_W^P requires $O(N)$ operations each). When lower bounds are calculated repetitively (e.g., within an branch-and-bound algorithm) savings in computation can be made because the jobs need only be sorted once, at an initial stage. At nodes in the search tree, the calculation of the lower bound can be completed in $O(N)$ operations (by surveying the ordered list of jobs in $O(N)$ time and updating an ordered list of setup times in $O(B)$ time).

The prime attribute of the Mason and Anderson lower bound is that it can be calculated quickly. Its on-average proximity to the optimal solution is not outstanding; for example, where setup times and processing times are distributed uniformly over [0, 100] and [1, 100], respectively, Mason and Anderson's experimental results show that the lower bound value is approximately 86% of the optimal solution value. Where the maximum setup time is reduced to be a tenth of the maximum processing time, the quality improves, however, to within about 4.5% of the optimal solution.

The lower bound developed by Crauwels, Hariri, Potts, and Van Wassenhove achieves significantly greater proximity to the optimal flowtime, at the cost of inflated lower bound computation times; the time complexity function for their lower bound is $O(NT)$, where T is an upper bound on the completion time of the last job in a schedule for the given instance of $1|s_i|\Sigma wC$. It is therefore evident that the running time of this lower bound is directly influenced by the magnitude of setup and processing time durations; hence, this time complexity function is *pseudo-polynomial*. The practical applicability of an algorithm (e.g., a branch-and-bound algorithm) incorporating this lower bound may be limited due to the effect setup and processing time durations will have on running times.

The Crauwels et al. lower bound is based upon *Lagrangian relaxation* of an integer programming formulation of the $1|s_i|\Sigma wC$ problem. We provide this formulation below; in this formulation, $\mathcal{J}$ is the set of jobs, $\mathcal{T}$ is the set of time intervals $\{1, 2, \ldots, T\}$, and $\mathcal{B}$ is the set of classes $\{1, 2, \ldots, B\}$. Crauwels et al. note that an upper bound for T is given by $\sum_{i=1}^{B} \sum_{j=1}^{N_i} (s_i + p_{i[j]})$.

minimize

$$\sum_{(i,\, j:a_{i[j]} \in \mathrm{J})} w_{i[j]} C_{i[j]}$$

subject to

$$\sum_{t \in \mathrm{T}} x_{i[j],\, t} = p_{i[j]} \qquad i, j{:}a_{i[j]} \in \mathrm{J}$$

$$\sum_{(i,\, j:a_{i[j]} \in \mathrm{J})} x_{i[j],\, t} + \sum_{b \in T} y_{bt} \leq 1 \quad t \in \mathrm{T}$$

$$s_i(x_{i[j],\, t} - x_{i[j],\, t-1} - x_{i[j-1],\, t-1}) \leq \sum_{t' = t - s_i}^{t-1} y_{bt'} \qquad (2.44)$$

$$i, j{:}a_{i[j]} \in \mathrm{J},\ t = s_i + 1, \ldots, T$$

$$x_{i[j],\, t} = \left\{ \begin{array}{l} 1 \text{ if } t \in \{C_{i[j]} - p_{i[j]} + 1, \ldots, C_{i[j]}\} \\ 0 \text{ otherwise} \end{array} \right\}$$

$$i, j{:}a_{i[j]} \in \mathrm{J}$$

$$C_{i[j]} \leq C_{i[j-1]} + p_{i[j]} \qquad i, j{:}a_{i[j]} \in \mathrm{J}, j > 1$$

$$C_{i[1]} \leq s_i + p_{i[1]} \qquad b \in \mathrm{B}$$

$$y_{bt} \in \{0, 1\} \qquad b \in \mathrm{B},\ t \in \mathrm{T}$$

It can be observed that $x_{i[j],t} = 1$ if job $a_{i[j]}$ is being processed during interval t and $y_{bt} = 1$ if a setup for class b is carried out during interval t.

Integer programming is not generally considered to be viable approach to the solution of the type of scheduling problems discussed in this chapter (i.e., the large integer programs cannot be solved in reasonable time). However, in this and a number of other cases, strong and useful lower bounds have been obtained from Lagrangian relaxation of integer programming formulations. For our current purposes, we merely note that in the case of Crauwels et al.'s lower bound for $1|s_i|\Sigma wC$, it is the machine capacity constraints (2.44) that are relaxed in the process of obtaining a lower bound. By relaxing (2.44), the Lagrangian problem decomposes into a separate sub-problem for each class, each of these sub-problems being efficiently solved using dynamic programming.

There exists another class of lower bounds that are applicable to the $1|s_i|\Sigma wC$ problem. The concept of these lower bounds is simple. In any schedule for a $1|s_i|\Sigma wC$ problem, all jobs belonging to any class i are preceded by at least one setup for that class. The contribution of class i setup times to the completion times of class i jobs is therefore no less than s_i time units. To obtain a lower bound, we *distribute* a total of s_i time units worth of processing time among class i jobs. In other words, we inflate job processing times from initial values $p_{i[j]}$ to new values $p'_{i[j]}$ according to:

$$p'_{i[j]} = p_{i[j]} + \beta_{i[j]} s_i$$

where $0 \leq \beta_{i[j]} \leq 1$ and $\sum_{i=1}^{N_i} \beta_{i[j]} = 1$. This process is carried out for all classes, and setup times are subsequently set to zero. By distributing setup times in this way, we generate an instance I' of a problem without setup times. Given that $0 \leq \beta_{i[j]} \leq 1$ and $\sum_{i=1}^{N_i} \beta_{i[j]} = 1$ are the only restrictions placed on the distribution of setup times, there will exist an infinite number of ways to distribute setup times (i.e., different *distribution schemes*, each specified by a set of $\beta_{i[j]}$ values).

If instance I' is to be solved as a $1||\Sigma wC$ problem, it can be solved optimally in $O(N \log N)$ time using the SWPT rule. Alternatively, we can recognize that in an optimal sequence for the $1|s_i|\Sigma wC$ problem jobs appear in SWPT order. Thus, after distributing the setup time, we can impose precedence constraints between jobs in I' to force jobs to appear in the same order as they would in the optimal sequence for I. These precedence constraints will be chain precedence constraints (see Fig. 2.1, Section 2.2), and I' will also be optimally solvable in $O(N \log N)$ time, as an instance of $1|tree|\Sigma wC$.

In either approach, the flowtime of the optimal schedule for I' is a lower bound for the optimal flowtime for the original instance I of the $1|s_i|\Sigma wC$ problem. Further, the "idea" of such *distributive lower bounds* can also be applied to many other flowtime problems with setup times, including the $1|s_i,r_{i[j]}|\Sigma wC$ and $R|s_i|\Sigma wC$ problems.

When I' is generated such that it is an instance of $1||\Sigma wC$, Dunstall, Wirth, and Baker [27] show that there is one distribution scheme that dominates all alternatives. This is the cascade distribution scheme, and the resulting lower bound can be termed the cascade lower bound. In the cascade lower bound, setup time is distributed with the aim of providing the same weighted processing time (p'/w) for all jobs in the class. If not enough setup time is available, the k jobs that originally had the least weighted processing time (i.e., least p/w) will receive distributed setup time. The weighted processing times of each of these k jobs are identical, as shown in Fig. 2.26.

The dominance of the cascade distribution scheme over other distribution schemes, when I' is an instance of $1||\Sigma wC$, is an interesting result that indicates the appropriateness of this distribution scheme when dealing with both the $1|s_i|\Sigma wC$ problem and more advanced problems with setup times. The cascade lower bound can be improved slightly by noting that some jobs receiving setup time will not be delayed by a full setup time to their class. An additional amount of flowtime can be added to the lower bound to account for this.

It can be shown that the "improved" cascade lower bound and the lower bound based on generation of I' as an instance of $1|tree|\Sigma wC$ are, in fact, equivalent. These "distributive" lower bounds can also be shown to dominate the Mason and Anderson lower bound. Computational analysis has illustrated that the "distributive" lower bounds clearly outperform the Mason and Anderson lower bound (see [27]).

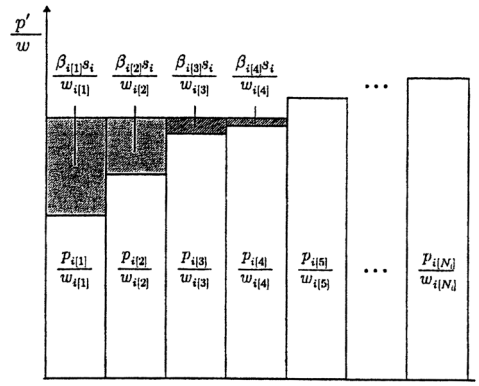


FIGURE 2.26 An illustration of the cascade lower bound.

Dominance Rules

From the three optimality conditions, Mason and Anderson develop a number of dominance relationships for sequence construction. These are given as Rules 1 through 5 below. Crauwels, Hariri, Potts, and Van Wassenhove [22] develop an additional dominance relationship, Rule 6, from the same basis. When used within branch-and-bound algorithms for the $1|s_i|\Sigma wC$ problem, these dominance rules have greatly assisted the pruning of the search tree, allowing the algorithms to return optimal solutions within reasonable time. Furthermore, they represent highly useful tools for the design and implementation of heuristics.

1. In an optimal sequence for the $1|s_i|\Sigma wC$ problem, where R_u and R_v are runs of the same class ($u < v$),

$$WMPT(R_u) \leq \frac{\tau_{v[1]}}{\varpi_{v[1]}} \tag{2.45}$$

   Thus, if extending a partial sequence with the addition of a new run of some class i, we should add jobs to this run at least until we use up all class i jobs (i.e., $a_{i[N_i]}$ is scheduled) or the next unscheduled job of this class has a weighted processing time greater than the current weighted mean processing time of the run.
2. Consider a partial sequence S within which the last run is R_v, the last scheduled job is $a_{i[j]}$, and $WMPT(R_{v-1}) > WMPT(R_v)$. If $j < N_i$, job $a_{i[j+1]}$ should be added to the end of R_v. If $j = N_i$, the partial sequence S cannot satisfy Theorem 6 and is sub-optimal.
3. In an optimal sequence for the $1|s_i|\Sigma wC$ problem, where R_u and R_v are runs of the same class ($u < v$),

$$\frac{\tau_{u+1[1]}}{\varpi_{u+1[1]}} \leq \frac{\tau_{v[1]}}{\varpi_{v[1]}}$$

   (as shown by Mason and Anderson in their Corollary 4.1). Thus, if we are extending a partial sequence S by adding a new run R_{u+1}, and S currently terminates with a run R_u of some class i, the weighted processing time $\tau_{u+1[1]} / \varpi_{u+1[1]}$ of the first job $\alpha_{u+1[1]}$ of R_{u+1} can be no greater than the weighted processing time of the next unscheduled job of class i.
4. If we wish to extend a partial sequence S by starting a new run R_v of class i, we must ensure that the two inequalities of Theorem 7 will be satisfied between R_v and the latest scheduled run R_u of the same class in S (if R_u exists).
5. If there are two or more jobs from the same class with the same $p_{i[j]}/w_{i[j]}$ ratios, then there is an optimal solution in which all such jobs appear consecutively.
6. Consider a partial sequence S, and let the last run in S be R_v, this run containing jobs of some class i. Let n_k be the number of jobs of class k which appear in S, $0 \leq n_k \leq N_k$, and assume without

loss of generality that these are the first jobs n_k jobs of class k. Let a class $k \neq i$ belong to set U if $n_k < N_k$, this being the set of classes that currently contains unscheduled jobs. If partial sequence S is such that

$$\frac{T_v}{W_v} > \min_{k \in U} \left\{ \frac{s_k + \sum_{j=n_i+1}^{N_i} p_{k[j]}}{\sum_{j=n_i+1}^{N_i} w_{k[j]}} \right\} \tag{2.46}$$

then if $n_i < N_i$, job $a_{i[n_i+1]}$ must be appended next to S. If $n_i = N_i$, partial sequence S cannot be optimal.

Some explanation of Rule 6 is necessary. The bracketed expression in (2.46) is the WMPT of a run formed of all unscheduled jobs of class k ($k \in U$). Such a run satisfies Rule 1, and has the maximum WMPT of any run of class k jobs that satisfies this rule. If $n_i = N_i$, T_v/W_v cannot be reduced by addition of class i jobs. Let k' be the class providing the minimum in (2.46). It is not possible to schedule the remaining class k' jobs so as to satisfy Rule 1 and Theorem 6; hence, S is sub-optimal.

Branch-and-Bound Approaches

The branch-and-bound approach to $1|s_i|\Sigma wC$ developed by Mason and Anderson utilizes Dominance Rules 1 to 5 in combination with their lower bound. Lower bound values are used to direct the search, and pruning of branches is undertaken with the use of the dominance rules as well as upper bounds calculated from complete schedules. Mason and Anderson study the performance of both this algorithm and a similar algorithm that makes use of the dominance rules and upper bounds alone (a depth-first search algorithm).

Crauwels, Hariri, Potts, and Van Wassenhove build upon Mason and Anderson's work and study a very similar branch-and-bound algorithm that utilizes their Lagrangian relaxation-based lower bound, Dominance Rules 1 to 6, and a further dominance rule (to be described shortly). The same authors also study a branch-and-bound algorithm that is significantly different from that of Mason and Anderson. We will not deal with this latter approach here, as data provided by Crauwels et al. illustrates that the Mason and Anderson-type approach gives superior performance. Neither will we consider the approach taken by Sahney [72] (see also [73]) for the special case where only two classes are to be scheduled.

Mason and Anderson observe that the SWPT-within-classes rule implies that when extending partial sequences of jobs, we need only choose the class of job to schedule next. The actual job scheduled will be that with the lowest value of weighted processing time of all jobs yet to be sequenced. In addition, Crauwels et al. note that two jobs $a_{i[j]}$ and $a_{i[k]}$that belong to the same class and have identical weighted processing times can be formed into a single "composite job" with weight $w_{i[j]} + w_{i[k]}$ and processing time $p_{i[j]} + p_{i[k]}$. This can be done as an alternative to direct utilization of Rule (5) in the sequence construction process. The optimal sequence for an instance with such composite jobs is also optimal for the original instance with the relevant jobs scheduled separately. We will assume that this pre-processing to form composite jobs has been carried out.

In the Mason and Anderson branch-and-bound algorithm for $1|s_i|\Sigma wC$, each node in the search tree is associated with a partial sequence corresponding to the first part of the schedule. This partial sequence is extended as the search branches from parent nodes to a child nodes. Complete sequences are generated at the lowest levels of the search tree. The elimination of branches that cannot lead to optimal sequences is enabled through comparison of lower bounds to upper bounds and by application of dominance rules. Clearly, the flowtime of a complete sequence provides an upper bound on the optimal flowtime.

In addition to upper bounds and Rules 1 to 6, Crauwels et al. also use what they term a "dynamic programming dominance rule." This rule can be applied to eliminate a node representing a partial sequence S_1 if (1) another node exists whose partial sequence S_2 contains the same jobs, (2) S_1 and S_2 share the same last job, (3) the completion time of this last job is no more in S_2 than it is in S_1, and (4) the weighted flowtime of S_2 is no greater than that of S_1.

The partial sequence at the root node of the search tree is empty, and branching from this node may be up to B nodes, one for each class in the instance. At these first-level nodes, a run of the appropriate class is formed. The run is constructed according to Rule 1, this rule governing the minimum size of a run. It is clear that a new node need not be added each time a single job is appended to a partial sequence; rather, new nodes are generated only when alternative sequencing decisions arise.

When expanding from a parent-node deeper in the search tree, formation of a child-node may be considered for each class with unscheduled jobs. "Immediate" elimination of some of these "potential" nodes may be possible, if Rules 3 and 4 are not satisfied. When branching to child-nodes not eliminated by the above, the partial sequence of the parent-node is extended, and Rules 1, 2, and 6 are relevant when extending partial sequences. If the child-node is not eliminated as sub-optimal during the extension of the partial sequence, a lower bound for a complete sequence beginning with the child-node's partial sequence can be calculated. Further elimination may be possible if the lower bound at a node exceeds the current upper bound.

Mason and Anderson explain that the branching strategy used within their algorithm can be termed "restricted flooding" and refer to the description of this approach provided by Rinnooy Kan [71]. When choosing the "path" to take when branching from a parent-node, the branch corresponding to the least lower bound is chosen and this child-node becomes the next parent-node. If a particular child-node is fathomed, the search returns to its parent-node and, if possible, expands the branch corresponding to the next-best lower bound. Otherwise, the search backtracks to the next-highest level, etc.

In Mason and Anderson's implementation, upper bounds are only calculated from complete sequences generated at the lowest levels of the search tree. In the implementation by Crauwels et al., an initial upper bound is also determined, using a heuristic solution. Furthermore, Crauwels et al. use a procedure for generating upper bounds at intermediate nodes. Potentially, this acts to further prune the search tree, as tighter upper bounds are located sooner. In the Crauwels et al. lower bound, Lagrange multipliers (for the lower bound) are obtained from feasible schedules, and the closer to optimal the feasible schedule, the tighter the lower bound can be expected to become. Thus, as the upper bound is associated with the best-known feasible schedule, calculation of upper bounds at intermediate nodes has a second important function in the Crauwels et al. implementation.

Both Mason and Anderson and Crauwels et al. provide results of computational testing. Mason and Anderson generate 100 instances for each possible combination of number of jobs $N = \{10, 20, 30\}$ and number of classes $B = \{1, 2, \ldots, 30\}$. A uniform distribution was used to give the number of jobs in each class (N_i), subject to the constraint that $\sum_{i=1}^{B} N_i = N$. In comparison, the main set of instances used by Crauwels et al. has $N = \{30, 40, 50\}$ and $B = \{4, 6, 8, 10\}$ with each class receiving either $\lfloor N/B \rfloor$ or $\lceil N/B \rceil$ jobs. The sample size utilized by Crauwels et al. was 50 instances per combination of B, N and range of setup times (see below). Further dissimilarity in test conditions is seen in choices of ranges of processing times, setup times, and weights. Mason and Anderson set all weights to one, while Crauwels et al. distribute weights uniformly over [1, 10]. Both sets of authors distribute processing and setup times uniformly, yet different ranges are utilized, as seen in Table 2.16.

Fortunately, the testing carried out by both sets of authors is extensive and a number of conclusions can be drawn directly from each set of published results. We are most interested here in the size of the search tree, measured by the average number of nodes (ANN), and the average computation times (ACT).

As noted by Mason and Anderson, the maximum possible number of nodes in a search tree for the $1|s_i|\sum wC$ problem is given by $N!/\prod_{i=1}^{N}(N_i!)$, this being the number of schedules satisfying the SWPT-within-classes rule only. By considering the potential magnitude of this number, it is evident that effective pruning of the search tree is vital. This pruning is facilitated by the dominance rules and lower bound strength.

The ANN is an excellent measure of the extent of pruning and thus the power of the dominance rules and lower bounds. ANN data reported by Crauwels et al. allows us to gauge the relative strength of the Mason and Anderson lower bound in comparison to the Crauwels et al. lower bound. The ANN for the branch-and-bound algorithm incorporating the Crauwels et al. lower bound is typically far less than that for the otherwise identical algorithm using Mason and Anderson's lower bound; Fig. 2.27 illustrates this for the "medium" range of setup times.

TABLE 2.16 Processing Time and Setup Time Ranges Used in Branch-and-Bound Algorithm Testing

Author	Processing Times	Setup Times
Mason and Anderson	[1, 100]	'Small' [0, 10] 'Medium' [0, 100]
Crauwels et al.	[1, 10]	'Small' [1, 5] 'Medium' [1, 10] 'Large' [2, 20]

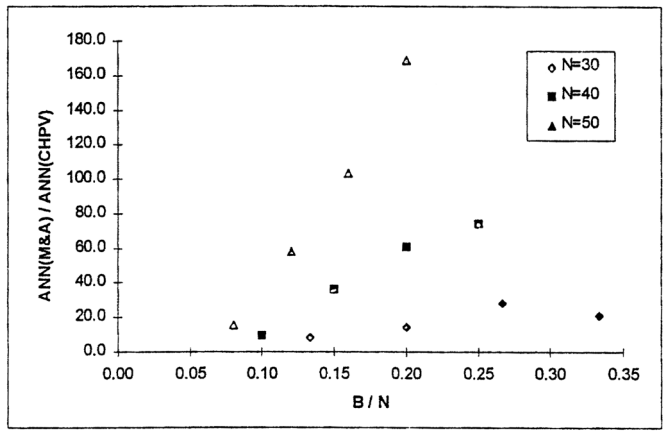


FIGURE 2.27 Ratio of branch-and-bound ANN values for the "medium" range of setup times (plotted from data provided in [22]).

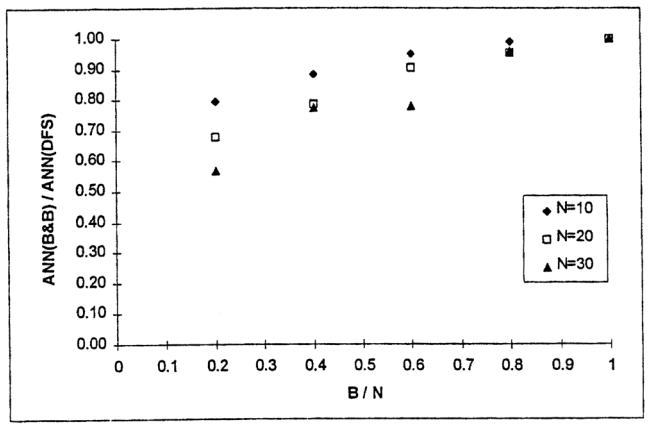


FIGURE 2.28 Ratio of branch-and-bound (B&B) and depth-first-search (DFS) ANN values (plotted from data provided in [59]).

From data provided by Mason and Anderson, it is possible to indicate the extent to which utilization of their lower bound assists in pruning the search tree, compared to the pruning possible in the depth-first search procedure, which applies dominance rules only. In Fig. 2.28 it is seen that the lower bound (in the branch-and-bound algorithm) does help to decrease the size of the search tree, yet it is also evident that the dominance rules account for much of the search tree pruning. The data in Fig. 2.28 relates to a "medium" range of setup times.

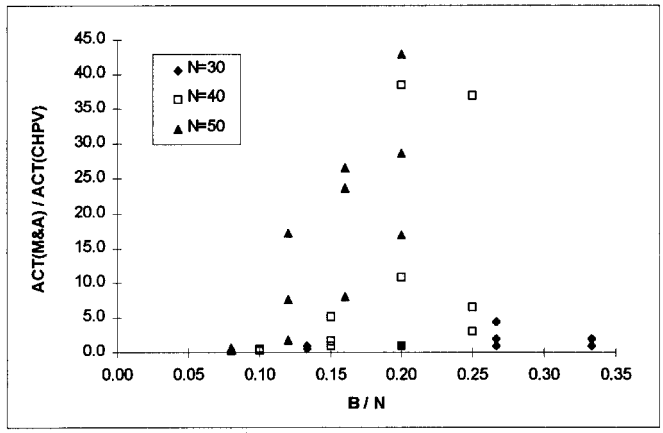


FIGURE 2.29 Ratio of branch-and-bound ACT values for the 'medium' range of setup times (plotted from data provided in [22]).

Mason and Anderson's article contains a more extensive analysis of such data. An interesting result is that for the "small" range of setup times, the difference in search tree sizes is more pronounced. Mason and Anderson conclude that the reason for this effect is that reducing the ratio of setup to processing times ($\bar{s}/\bar{p}$) weakens the dominance rules. This hampers the depth-first search procedure more severely than it does the branch-and-bound (which can rely on the lower bound for pruning). For both algorithms, the reduction in the value $\bar{s}/\bar{p}$ and the associated weakening of the dominance rules leads to increased problem difficulty, as measured by both search tree sizes and computation times.

ANN data supplied by Crauwels et al. follows the same trend: reduction in $\bar{s}/\bar{p}$ leads to an increase in ANN. As observed by Crauwels et al., when setup times are large in comparison to processing times, the "splitting" of a class into many runs is very expensive; thus, there will be a tendency to schedule most jobs of a class together and the combinatorial characteristics of the problem are reduced. In contrast, low values of $\bar{s}/\bar{p}$ make multiple runs more attractive.

Consideration of Figs. 2.27 and 2.28 together illustrate the idea that application of strong dominance rules and tight lower bounds can lead to successful branch-and-bound approaches to scheduling problems.

The increased restriction of search tree size by the Lagrangian lower bound does not necessarily lead to shorter branch-and-bound computation times, due to the greater computational effort required in calculating this lower bound. Certainly, the Lagrangian lower bound can be time-consuming when the durations of tasks in the instance are distributed over a large range (due to the $O(NT)$ bound).

However, when setup and processing times are at most 10 time units, the Crauwels et al. algorithm is on average at least as fast as the Mason and Anderson algorithm for 29 of the 36 problem sets tested with 30, 40, or 50 jobs (i.e., for all setup time ranges), according to ACT data reported by Crauwels et al.* The algorithm equipped with the Mason and Anderson lower bound is faster on average for seven of the problem sets, and each of these has the smallest number of classes in the test program (four classes). The Crauwels et al. algorithm has an ACT of less than 1 second in all except two of these seven problem sets.

Fig. 2.29 clearly illustrates the relative "explosion" of average computation times experienced by the Mason and Anderson algorithm as N or B is increased. The maximum reported ACT for the Mason and Anderson algorithm is 68.9 seconds for problems with $N = 50$, $B = 10$, and "small" setup times. In contrast, the maximum reported ACT for the Crauwels branch-and-bound algorithm when applied to a 50 job problem was 2.6 seconds ("small" setup times, $B = 8$). It is obvious that the Mason and Anderson lower

*To the best of our knowledge, the data reported here will appear in the article by Crauwels et al.

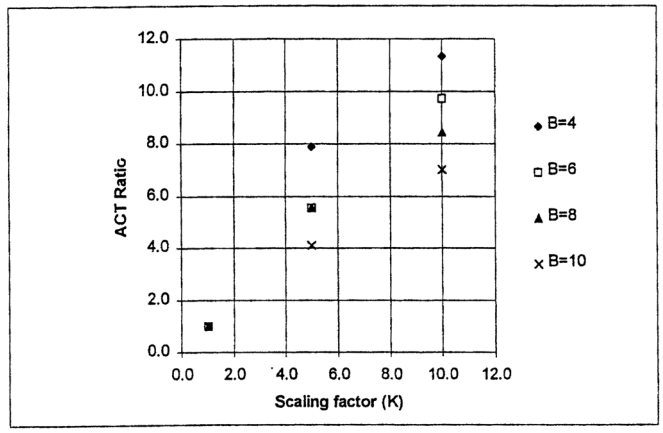


FIGURE 2.30 Effect of mean task duration scaling on branch-and-bound performance, plotted from data provided by Crauwels.

bound exerts far too little control over the search tree size as N increases. This makes the branch-and-bound algorithm incorporating this lower bound very unattractive for N larger than 50.

Distributing processing and setup times over a "wide" range will have quite an adverse effect on observed computation times for the Crauwels et al. lower bound. Essentially, an increase by some factor K in mean setup time $\bar{s}$ and mean processing time $\bar{p}$ (i.e., *mean task duration*) should lead to a proportional increase in lower bound running time. This will, in turn, affect the running time of the algorithm (measured by ACT).

Figure 2.30 has been constructed using data provided by Crauwels in a personal communication. For this plot, instances have been generated with "medium" setup times, and all times in the instance are scaled up by the chosen value of K. The ACT ratio is the value of ACT at some scaling factor K divided by the value of ACT in the case where $K = 1$. It is seen in this figure that there is indeed a strong relationship between the scaling factor and the algorithm running time, with the number of classes B also affecting the relationship. Associated ANN data (not plotted) shows that the ANN does not show significant variation with K. This suggests that problem difficulty is not affected by scaling.

Dynamic Programming Approaches

Dynamic programs proposed for the more general $1|s_{ij}|\Sigma wC$ problem can evidently be used to solve the $1|s_i|\Sigma wC$ problem, given that sequence-independent setup times are a special case of sequence-dependent setup times. Thus, each of the dynamic programs noted in Section 2.3 could be utilized for $1|s_i|\Sigma wC$, with the most efficient of these approaches requiring $O(B^2N^B)$ time.

Heuristic Approaches

Both Gupta's heuristic ([39], Section 2.3) and Ahn and Hyun's heuristic ([3], Section 2.3) can be utilized to solve the $1|s_i|\Sigma wC$ problem once minor modification is made to cater for job weights in the objective. Furthermore, a number of heuristics have been specifically developed for the problem with sequence-independent setup times. Of most interest here are the genetic algorithm devised by Mason [58] and the local search heuristics developed by Crauwels, Potts, and Van Wassenhove [23]. We will discuss each of these heuristics within this section, in order to complete our investigation into common algorithmic methods for solving machine scheduling problems with setup times.

As observed in Sections 2.2 and 2.3, local search heuristics seek optimal schedules by iteratively modifying complete sequences. The search continues until a prespecified termination criterion is fulfilled; for example, a maximum number of iterations is exceeded. It should be noted that a search heuristic can usually be credited only with a pseudo-polynomial running time bound, particularly if the maximum number of iterations is not predetermined.

Crauwels, Potts, and Van Wassenhove develop tabu search, simulated annealing, and threshold accepting heuristics for the $1|s_i|\Sigma wC$ problem. Two different neighborhoods for these heuristics are investigated by Crauwels et al., these being the *shift sub-batch* neighborhood and the *shift job* neighborhood. The shift sub-batch neighborhood is identical to that used by Ahn and Hyun within their heuristic for $1|s_{ij}|\Sigma wC$ (Section 2.3). The shift job neighborhood is a smaller neighborhood, where alternative sequences are produced by either shifting the last job of a run to the beginning of the next run of the same class, or the first job of a run to the end of the previous run of the same class. SWPT ordering of jobs is preserved by all moves in both neighborhoods, while SWMPT ordering of runs and the extended SWPT rule may not be.

Two processes are used to generate seed schedules in the Crauwels et al. local search heuristics. The first is an adaptation to the $1|s_i|\Sigma wC$ problem of Gupta's greedy heuristic for $1|s_{ij}|\Sigma wC$ (as described in Section 3.3). A schedule is produced using the modified Gupta heuristic, and then runs are sorted in SWMPT order to produce a better schedule. This schedule will satisfy both the SWPT-within-classes rule and SWMPT rule for runs (and our own computational testing shows that these seed schedules are typically within 6% of the optimal solution).

For multi-start implementations of the local search heuristics, further seeds are generated using a second process. In this, the jobs of a class are first sorted into SWPT order. Each job is considered in turn and a decision made randomly as to whether the job will constitute the last job in a run; Dominance Rule 1, presented earlier, is used to prohibit certain jobs from being the last in a run. The runs thus formed are then arranged into a sequence that satisfies the SWMPT rule and that does not contain adjacent runs of the same class.

The primary differences between the Ahn and Hyun heuristic and each of the three local search heuristics developed by Crauwels et al. are seen in the methods used to accept modified sequences. Ahn and Hyun's descent heuristic accepts the first move found that strictly improves the flowtime. Such moves may be called *improving moves*, and can be compared to *deteriorating moves*, which worsen the value of the objective function, and *neutral moves*, which generate a new sequence that has the same objective function value as the sequence it replaced. Descent heuristics have no ability to escape from local optima because they only accept improving moves. Search heuristics that can accept deteriorating moves do exhibit this ability.

For any deteriorating move, there is a corresponding improving move that reinstates the previous sequence. Thus, it is insufficient to simply alter a descent heuristic so that it accepts deteriorating moves when no improving moves are possible (as a deteriorating move will often be "undone" on the next iteration). If deteriorating moves are to be accepted, more intelligent modifications to descent heuristics, such as those found in a tabu search heuristic, are required.

Essentially, a tabu search heuristic "remembers" which deteriorating moves have been undertaken, and disallows their reversal; that is, such "reversing moves" are tabu. The memory of a tabu search heuristic is a *tabu list*. There exist many different strategies for constructing and maintaining tabu lists; some issues to be considered are the method by which moves are characterized, the length of the tabu list (i.e., how far back it remembers), and the way in which the tabu list is maintained and referenced by the heuristic. Readers are referred to the tutorial papers by Glover ([34],[35]), which discuss the features of tabu search heuristics at greater length.

Crauwels et al. investigate a number of variants of tabu search heuristics for the $1|s_i|\Sigma wC$ problem. Using some initial computational testing, they select a particular variant as most successful, and use it within their major computational testing which compares the different local search approaches. This tabu search heuristic uses the shift job neighborhood, has multiple starts ($N/3$ seed schedules), and uses a tabu list that prevents moves which shift certain jobs. After a move is carried out, the job that was moved is recorded at the beginning of the tabu list, and ordinarily cannot be moved again within a certain number of moves (when $N \leq 70$, this is approximately 7 moves, as the tabu list is 7 jobs long). The primary stopping rule used in the Crauwels et al. tabu search heuristic allows $2N$ iterations to be performed. The SWPT-within-classes rule is always preserved during the operation of the heuristic. A partial reordering of runs according to SWMPT order is undertaken after each move, yet SWMPT order will not necessarily be present throughout as a reordering of runs containing only one job might correspond to an execution of moves that are tabu.

The simulated annealing and threshold accepting heuristics use an alternative acceptance strategy. Unlike tabu search, which accepts the best improving move or a least-worst deteriorating move on any iteration, any moves can be accepted in simulated annealing (with a certain probability favoring improving moves), while deteriorating moves are undertaken in threshold accepting as long as the increase in the objective function value is not in excess of the *threshold value*. In [26], Eglese discusses many relevant aspects of simulated annealing heuristics, and Dueck and Scheuer [26] introduce threshold accepting heuristics and report on their success in solving the TSP with such an algorithm.

Acceptance probabilities for deteriorating moves in simulated annealing algorithms commonly take the form $p(\delta) = \exp(-\delta/T)$, where δ is the increase in objective function due to the move (for minimization problems) and T is the *temperature* parameter (the "simulated annealing" term literally arises from the original application, that being simulation of metal annealing processes). It is clear from this probability function that all neutral and improving moves are to be accepted.

Higher temperatures increase the chances that a particular deteriorating move will be accepted. By varying the temperature during the progress of a simulated annealing algorithm, control is gained over the balance between diversification of the search into new areas of the solution space and concentration of the search in the current area. Particular simulated annealing algorithms are characterized by the schemes used to vary the temperature. The most successful scheme for the $1|s_i|\Sigma wC$ problem appears to vary the temperature in a periodic manner (which is specified within the article by Crauwels et al.). This is in contrast to the more common procedure of decreasing the temperature throughout the process.

Computational results of Crauwels et al. also show that the shift sub-batch neighborhood, in combination with multiple seed schedules, yields the most promising performance for simulated annealing heuristics. In the preferred (multi-start) simulated annealing heuristic for $1|s_i|\Sigma wC$, up to $2N/R$ complete passes are made through the sequence (R being the number of starts, $R = 4$ being chosen for testing), and at each point all moves in the neighborhood are assessed and potentially carried out. The temperature variation scheme is such that temperature is altered on each new pass through the sequence. The threshold value used for acceptance of deteriorating moves in the threshold accepting heuristic is varied in a similar periodic manner within the Crauwels et al. implementation. The simulated annealing and threshold accepting heuristics proposed by Crauwels et al. differ only in the acceptance criteria used.

There is considerable difference between the local search heuristics of the above type and evolutionary algorithms. We will use the example of the $1|s_i|\Sigma wC$ problem to illustrate some of the issues and considerations relevant when applying the concept of evolutionary algorithms to machine scheduling problems. Although the concept of evolutionary algorithms is simple, a thorough discussion of their application requires an analysis far more extensive than that able to be undertaken here (e.g., see Mattfeld [60] or Michalewicz [61]).

A major task in the development of a genetic algorithm for a machine scheduling problem is the determination of an appropriate and effective genotype, as well as a scheme that translates a chromosome into a schedule (as a whole, we will call this an *encoding* for the problem). It can be appreciated that a chromosome definition should allow representation of at least those schedules which satisfy known dominance relationships for the problem; otherwise, we may unwittingly exclude optimal solutions from the search space. It can also be suggested that all possible schedules should be able to be represented by chromosomes. However, allowing all possible schedules to be represented can lead to an inefficient search, as much of the known structure of the problem may be ignored (in fact, it is difficult to formulate genetic algorithms that make good use of problem structure, this being a valid criticism of these methods).

Mason investigates three encoding schemes for $1|s_i|\Sigma wC$ that use dominance relationships to varying degrees. All three schedule in accordance with the SWPT-within-classes rule. Computational results provided by Mason show that, without question, the most successful coding scheme (class-run decoding) is that which more fully utilizes the known problem structure. In the class-run decoding method, all the required information for forming runs of a particular class is found in a continuous section of chromosome; gene information indicates which jobs should begin a new run of their class. Runs are formed using the encoded information and then arranged in SWMPT order to generate a schedule (i.e., the phenotype of the individual).

Dominance Rule 1 for the $1|s_i|\Sigma wC$ problem, which governs the "minimum size" of a run, plays an essential role in both the definition and decoding of a genotype in the class-run coding scheme. Rule 1 enables us to determine which jobs must be part of the first run in an optimal schedule, and further, the minimum number of jobs to be assigned to a run starting with any job $a_{i[j]}$. Define $\lambda_{\min}(a_{i[j]})$ as the minimum number of jobs in a run of class i which begins with job $a_{i[j}$ (assuming jobs are sequenced according to the SWPT-within-classes rule, and indexed in SWPT order), so that:

$$\lambda_{\min}(a_{ij}) = \min_{k \geq 1} \left\{ k: \frac{s_i + \sum_{j'=j}^{j+k-1} p_{i[j']}}{\sum_{j'=j}^{j+k-1} W_{i[j']}} \leq \frac{p_{i[j+k]}}{w_{i[j+k]}} \right\}$$

and $\lambda_{\min}(a_{ij}) = N_i = -j + 1$ if no value of k satisfies the inequality.

We can calculate $\lambda_{\min}(a_{i[1]})$ for each class i, and assign $k_{i_B} = N_i - \lambda_{\min}(a_{i[1]})$ genes in a chromosome to class i. The total length of a chromosome will be given by $\sum_{i=1}^{B} k_i$. Genes may be viewed as instructions for the construction of runs, and take binary values (i.e., 0 or 1). An allele of 0 indicates that the next unallocated job should be added to the run currently under construction, and an allele of 1 indicates that the next unallocated job should begin a new run. These instructions are followed from the start of the relevant chromosome section onward, and when dealing with some class i, we first construct a run $R_{[1]}$ beginning with $a_{i[1]}$ and containing $\lambda_2 = \lambda_{\min}(a_{i[1]})$ jobs. The first gene is then consulted. If an allele of 0 is present, the next unallocated job $a_{i[\lambda_1 + 1]}$ is added to $R_{[1]}$. Otherwise, if this allele is 1, $a_{i[\lambda_1 + 1]}$ begins a new run ($R_{[2]}$) containing $\lambda_2 = \lambda_{\min}(a_{i[\lambda_1 + 1]})$ jobs, that is, jobs $a_{i[\lambda_1 + 1]}$ to $a_{i[\lambda_1 + \lambda_2]}$. The next instruction is obtained from the following gene, etc., until all jobs have been assigned to runs.

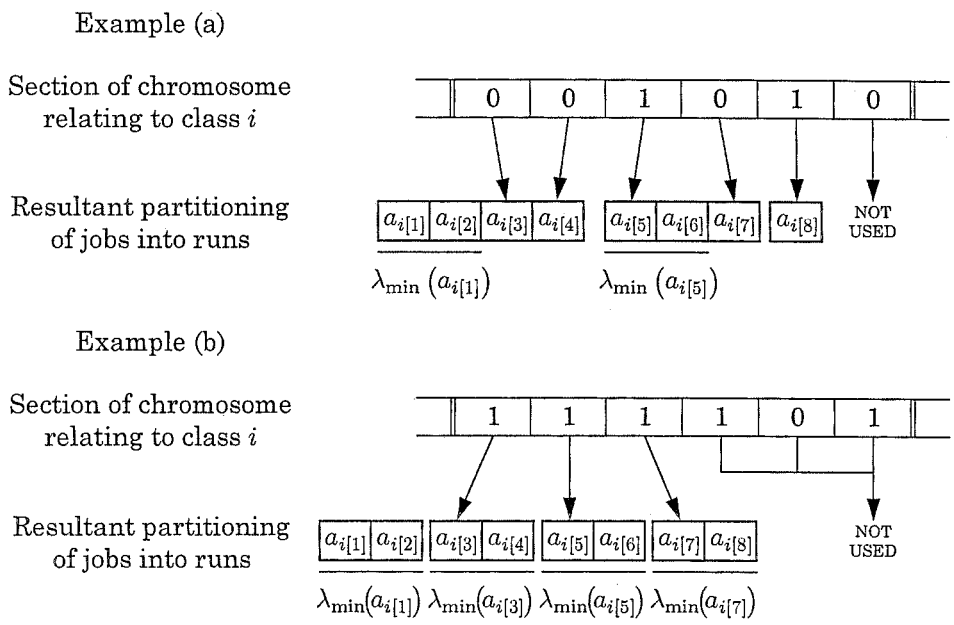
Figure 2.31 illustrates this chromosome decoding process, for an example instance given by Mason which has all job weights set to 1. We are concerned with run construction for some class i in the instance, this class having 8 jobs and a setup time of 4. As shown in the included table, $\lambda_{\min}(a_{i[1]}) = 2$, so that 6 genes of the chromosome are allocated to this class.

It is clear from Fig. 2.31 that the decoding scheme is such that genes near the end of the chromosome section may be not used in run construction on some occasions. This redundancy of genes is not ideal, and it can also be noted that the some sequences can be represented by more than one genotype. The class-run coding scheme is quite problem specific, taking advantage as it does of many particular attributes of the $1|s_i|\Sigma wC$ problem. The nature of machine scheduling problems is such that this specialization is largely unavoidable when attempting to construct efficient heuristics. Interestingly, any sequence is feasible for the $1|s_i|\Sigma wC$ problem, as long as it contains every job once and once only. When solving more complex problems, the means by which infeasible solutions are either discarded, penalized, or repaired is an important issue that needs to be addressed.

Specification of a chromosome representation and chromosome decoding is one major aspect of genetic algorithm development for machine scheduling problems. The wise selection of reproductive and mutative processes is also of great importance when developing effective genetic algorithms for these problems (we will discuss only the former here).

Reproduction in genetic algorithms for combinatorial problems can be a complex issue. Essentially, parent individuals "mate" and generate offspring, these offspring combining elements of the genotype of both parents. The basis of a reproductive process in a genetic algorithm is the *crossover operator*. Some simple and common crossover operators are shown in Fig. 2.32. For each operator shown, the crossover point (or points) are determined at random and the genetic material for the offspring copied from that of the parents (alternatively at either side of the crossover point).

Although Fig. 2.32 illustrates crossover in a case where alleles are binary, these principles also apply for the allele "alphabets" of greater cardinality that are commonly used in machine scheduling problems. However, more complex crossover operators (such as *order-based* crossover operators, devised by Davis [24]) can yield better results in these cases.



Job index j	1	2	3	4	5	6	7	8
Processing time $p_{i[j]}$	4	6	7	8	10	12	18	20
$\lambda_{\min}(a_{i[j]})$	2	3	2	2	2	1	2	1

FIGURE 2.31 Examples of chromosome decoding in Mason's genetic algorithm using the class-run coding scheme.

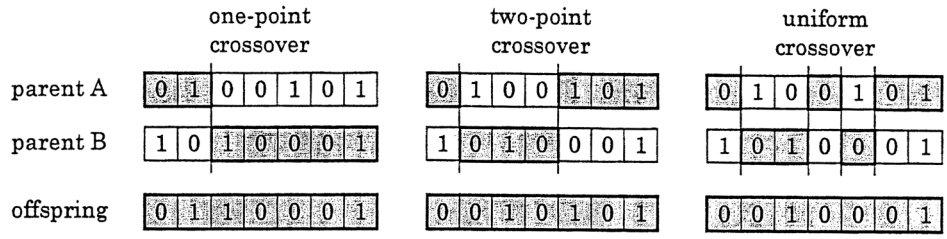


FIGURE 2.32 An illustration of three common crossover operators. The resulting offspring obtains a chromosome formed of the shaded sections in the parent chromosomes (diagram adapted from Mattfeld [60]).

Appropriate choice of crossover operator can greatly affect the performance of a genetic algorithm. Likewise, the method used to select individuals for passing on to the next generation (the "selection scheme"), as well as factors such as mutation rate, can also affect genetic algorithm performance significantly. Clearly, experimental trials can be used to identify the most promising variants of a genetic algorithm. Using this approach, Mason identifies one-point crossover as a successful operator for the $1|s_i|\Sigma wC$ problem using class-run encoding. Other important specifications of Mason's successful class-run-encoded genetic algorithm are: a population of 50 individuals, an "elitist" strategy for selecting the next generation, a mutation probability of $\rho_m = 0.001$, and a crossover probability of $\rho_c = 0.6$ (this being the probability that crossover will be applied to a selected individual; a value of 0.6 is a common choice in the literature).

TABLE 2.17 Combinations of B and N Tested by Crauwels, Potts, Van Wessenhove

Number of jobs N	40, 50	60, 70	100
Number of classes B	4, 6, 8, 10	4, 8, 15	4, 6, 8, 10, 15, 20

TABLE 2.18 Summary of Heuristic Performance for the $1|s_i|\Sigma wC$ Problem

	DS(N/3)			SASP(4)			TASP(4)			TSJJ(N/3)			GA(2)		
Problems	NO	MPD	ACT	NO	MPD	ACT	NO	MPD	ACT	NO	MPD	ACT	NO	MPD	ACT
N=40 to 70	45	0.08	8.3	45	0.1	8.2	43	0.1	8.5	48	0.05	5.1	43	0.2	8.4
N=100	39	0.06	71.1	37	0.09	57.8	27	0.01	54.1	42	0.05	44.7	31	0.19	54.4

Crauwels et al. report on extensive computational testing of the search heuristics outlined above. The branch-and-bound algorithm (of Crauwels, Hariri, Potts, and Van Wassenhove [22]) described earlier is used to provide optimal solutions. Processing times, weights, and setup times are generated for test instances in a manner identical to that followed for evaluation of the branch-and-bound algorithm (including the use of "small," "medium," and "large" ranges of setup times). Tested combinations of N and B are given in Table 2.17. At each level of B, N and range of setup times, 50 test problems were generated.

An important result made clear by the reported test data from Crauwels et al. is that each of the local search methods can be adapted to provide a heuristic for the $1|s_i|\Sigma wC$ problem that exhibits very good objective function performance at acceptable computation times. On average, it is the tabu search algorithm that is most effective over the range of instances solved. This is illustrated by the "grand averages" shown in Table 2.18, this data obtained from Tables 5 and 6 of the article by Crauwels, Potts, and Van Wassenhove [23].

In Table 2.18, NO refers to the number of optimal solutions found (out of 50 test instances), while MPD and ACT refer to the maximum percentage deviation and average computation time, respectively. DS(N/3), SASP(4), TASP(4), and TSJJ(N/3) are the previously described "preferred" versions of the Ahn and Hyun heuristic, simulated annealing, threshold accepting, and tabu search heuristics, respectively. The number of starts given to each heuristic is shown in parentheses. GA(2) is Mason's genetic algorithm with class-run encoding and 2 starts. Fig. 2.33 illustrates the performance of the search heuristics for the largest instances solved by Crauwels et al.

Crauwels et al. are able to conclude that their tabu search heuristic is the most effective when the number of classes is relatively small, while for larger values of B Mason's genetic algorithm tends to perform better. We note their concluding observation that a hybrid method that invokes tabu search for low B/N values and the GA for high B/N values should provide quality solutions over a wide range of $1|s_i|\Sigma wC$ instances.

We note at this point that much caution is required when "extrapolating" results of computational studies involving search heuristics for a particular problem to other problems, or even alternative data sets for the same problem. The performance of search heuristics is affected by neighborhood choice, choice of search parameters such as temperature and number of starts, and the structure of the problem. A search method that performs well for one problem may or may not be equally successful when adapted to a different problem.

Finally, we note other heuristic approaches to the single machine flowtime problem with sequence-independent setup times. Williams and Wirth [78] provide an $O(N^4)$ constructive heuristic for the problem, which is shown to provide reasonable performance for the $1|s_i|\Sigma C$ problem when compared to Gupta's heuristic. Descent heuristics are also provided by Baker [6], Dunstall, Wirth, and Baker [27], Mason [58], and Crauwels, Hariri, Potts, and Van Wassenhove [23]. The latter heuristic is used to provide both an initial upper bound and a set of Lagrange multipliers for use within the branch-and-bound algorithm discussed earlier.

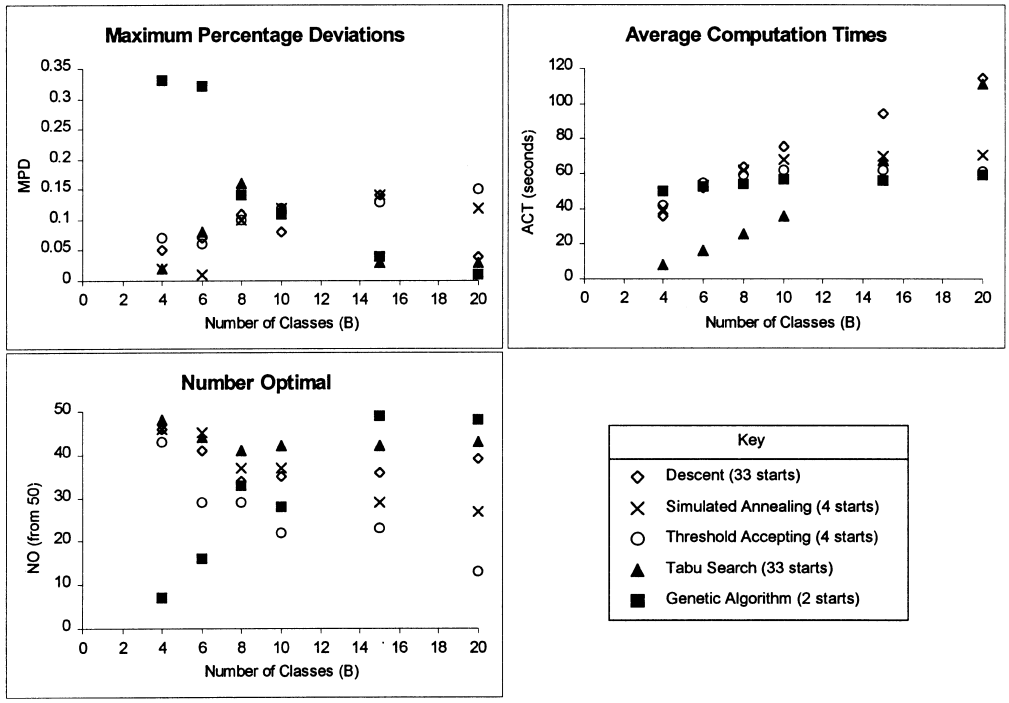


FIGURE 2.33 Search heuristic computational results for 100 job instances with a "medium" range of setup times (plotted using data published in [23]).

Flowtime Problem Extensions

Static flowtime problems on single machines with setup times have attracted attention from a number of researchers. Details of many of the important approaches to such flowtime problems with sequence-dependent and sequence-independent setup times models have been discussed.

Clearly, algorithms for the $1|s_{ij}|\Sigma wC$ problem can be used to solve any problem incorporating the major–minor setup times model. Nevertheless, Liao and Liao illustrate that the special structure of the $1|s_i{:}s_i|\Sigma wC$ problem can be exploited to provide efficient heuristics specifically tailored for this problem. These authors propose a novel heuristic approach to the problem that iteratively uses dynamic programming to solve subproblems with two classes or sub-classes only. This heuristic has a polynomial running-time bound, and over a range of test problems exhibits acceptable computation times (on the order of a few seconds) and excellent objective function performance. In addition, Nowicki and Zdrzalka [64] formulate a "general" tabu search algorithm for static single machine problems with regular objective functions, sequence-independent major setup times, and "weakly" sequence-dependent minor setup times; a minor setup time of fixed duration s is required between jobs of the same class if the job being switched to has a lower index. Nowicki and Zdrzalka report encouraging computational experience with weighted tardiness and maximum weighted lateness objectives.

The assumption of static job availability is restrictive, yet unfortunately little research has been carried out on flowtime problems with setup times and dynamic job arrivals. Research efforts in this direction will thus make worthwhile contributions to the machine scheduling field. Although the $1|r_{i[j]}, s_i|\Sigma C$ problem (for example) can be expected to be significantly more difficult to solve than the $1|r_j|\Sigma C$ or $1|s_i|\Sigma C$ problems, research into these "simplified" problems can be drawn on to assist study of the more "advanced" dynamic problem with setup times. This is a continuing area of interest for the authors. Similarly, there are a number of other aspects of practical scheduling problems that have not yet been addressed in the literature pertaining to the scheduling of single machines with setup times and the flowtime or makespan objective.

2.4 Summary

Section 2.3 considered a range of machine scheduling problems with setup times. Through the study of these problems, we have illustrated the way in which knowledge of the "structure" of machine scheduling problems can be developed using mathematical analysis, and have shown adaptations of many common algorithmic methods to the solution of problems with setup times. Each of the models of setup times introduced in Section 2.2 have been represented in the set of problems studied. Furthermore, the approaches to problem analysis and solution described in Section 2.3 are highly representative of those commonly found in scheduling research. Thus, we believe that we have achieved the goals set forth in the introduction (Section 2.1).

We have not addressed problems with multiple machines, and have not looked in detail at objective functions that are directly concerned with the due date performance of jobs. This we can accept as a criticism; yet by concentrating on single machines and a restricted set of objectives, we hope to have conveyed with increased clarity the way in which machine scheduling problems with setup times can be formulated, analyzed, and successfully solved.

There can be no doubt as to the important and necessary role that effective scheduling plays in enabling the best utilization of limited resources such as time, labor, and machine capacity. It is also clear that scheduling is a complex activity that is often very difficult to undertake effectively. Mathematical scheduling approaches provide methods and analyses that can be extremely valuable in the practical scheduling processes undertaken throughout commerce and industry.

The ongoing challenge for scheduling research is to increase the relevance of "mathematical scheduling theory" to "practical scheduling problems." Although not discussed at length in this chapter, exploration of new means of incorporating mathematical scheduling methods into processes for in-practice scheduling represents one important aspect in this quest. This relevance can also be increased by deploying mathematical scheduling techniques in the solution of problems with greater practical applicability. Setup times are prominent features of many practical scheduling problems. Thus, when attempting to bridge the gap between scheduling research and practical scheduling problems, analysis of machine scheduling problems with setup times is of crucial importance.

References

1. J. Adams, E. Balas, and D. Zawack. The shifting bottleneck procedure for job shop scheduling. *Management Science,* 34(3):391–401, 1988.
2. D. Adolphson and T. C. Hu. Optimal linear ordering. *SIAM Journal on Applied Mathematics,* 25(3):403–423, 1973.
3. B.-H. Ahn and J.-H. Hyun. Single facility multi-class job scheduling. *Computers and Operations Research,* 17(3):265–272, 1990.
4. A. D. Amar and J. N. D. Gupta. Simulated versus teal life data in testing the efficiency of scheduling algorithms. *IIE Transactions,* 18:16–25, 1986.
5. K. R. Baker. *Introduction to Sequencing and Scheduling*. John Wiley & Sons, 1974.
6. K. R. Baker. Solving the weighted completion time problem with batch setup times. Working Paper 98–118, Amos Tuck School, Dartmouth College, 1998.
7. R. Bellman. Dynamic programming treatment of the traveling salesman problem. *Journal of the ACM,* 9:61–63, 1962.
8. L. Bianco, S. Ricciardelli, G. Rinaldi, and A. Sassano. Scheduling tasks with sequence-dependent processing times. *Naval Research Logistics,* 35:177–184, 1988.
9. J. Blazewicz. Scheduling dependent tasks with different arrival times to meet deadlines. In H. Beilner and E. Gelenbe, Eds., *modeling and Performance Eualuation of Computer Systems,* 1976, 57–65.
10. P. Bratley, M. Florian, and P. Robillard. Scheduling with earliest start and due date constrainsts. *Naval Research Logistics Quarterly,* 18(4):511–519, 1971.

11. P. Brucker. *Scheduling Algorithms.* Springer-Verlag, 1995.
12. J. Bruno and P. Downey. Complexity of task sequencing with deadlines, set-up times and changeover costs. *SIAM Journal on Computing,* 7(4):393–404, 1978.
13. J. Bruno and R. Sethi. Task sequencing in a batch environment with setup times. *Foundations of Control Engineering,* 3:105–117, 1978.
14. J. L. Burbidge. *Production Flow Analysis for Planning Group Technology.* Oxford University Press, 1989.
15. G. Burns, J. Rajgopal, and B. Bidanda. Integrating group technology and TSP for scheduling operations with sequence-dependent setup times. In *2nd Industrial Engineering Research Conference Procedings,* IIE, 1994, 837–841.
16. O. Charles-Owaba and B. Lambert. Sequence-dependent machine set-up times and similarity of parts: a mathematical model. *IIE-Transactions,* 20(1):12–21, 1988.
17. C.-L. Chen and R. L. Bulfin. Complexity of single-machine, multicriteria scheduling problems. *European Journal of Operational Research,* 70:115–125, 1993.
18. T. C. E. Cheng and Z.-L. Chen. Parallel machine scheduling with batch setup times. *Operations Research,* 42(6):1171–1174, 1994.
19. G. Clarke and J. W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research,* 12:568–581, 1964.
20. R. W. Conway, W. L. Maxwell, and L. W. Miller. *Theory of Scheduling.* Addison-Wesley Publishing, 1967.
21. S. A. Cook. The complexity of theorem-proving procedures. In *Conference Record of the 3rd Annual ACM Symposium on the Theory of Computing,* 1971, 151–158.
22. H. A. J. Crauwels, A. M. A. Hariri, C. N. Potts, and L. N. Van Wassen-hove. Branch and bound algorithms for single machine scheduling with batch setup times to minimise total weighted completion time. *Annals of Operations Research,* (to appear).
23. H. A. J. Crauwels, C. N. Potts, and L. N. Van Wassenhove. Local search heuristics for single machine scheduling with batch setup times to minimize total weighted completion time. *Annals of Operations Research,* 70:261–279, 1997.
24. L. Davis. Applying adaptive algorithms to epistatic domains. In *Proceedings of the Ninth International Joint Conference on Artificial Inteligence,* 1985, 162–164.
25. S. R. Dreyfus and A. M. Law. *The Art and Theory of Dynamic Programming.* Mathematics in Science and Engineering, Academic Press, 1977.
26. G. Dueck and T. Scheuer. Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing. *Journal of Computational Physics,* 90:161–175, 1990.
27. S. Dunstall, A. Wirth, and K. R. Baker. Lower bounds and algorithms for flowtime minimization on a single machine with setup times, (in preparation).
28. R. W. Eglese. Simulated annealing: a tool for operational research. *European Journal of Operational Research,* 46:271–281, 1990.
29. F. C. Foo and J. G. Wager. Set-up times in cyclic and acyclic group technology scheduling systems. *International Journal of Production Research,* 21(1):63–73, 1983.
30. M. R. Garey and D. S. Johnson. Two-processor scheduling with start-times and deadlines. *SIAM Journal on Computing,* 6(3):416–426, 1977.
31. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness.* Bell Telephone Laboratories, 1979.
32. J. W. Gavett. Three heuristic rules for sequencing jobs to a single production facility. *Management Science,* 11(8):B166–B176, 1965.
33. J. B. Ghosh. Batch scheduling to minimize total completion time. *Operations Research Letters,* 16:271–275, 1994.
34. F. Glover. Tabu search – Part one. *ORSA Journal on Computing,* 1(3):190–206, 1989.
35. F. Glover. Tabu search – Part two. *ORSA Journal on Computing,* 2(1):4–32, 1990.

36. A. Goicoechea, D. R. Hansen, and L. Duckstein. *Multiobjective Decision Analysis with Engineering and Business Applications*. John Wiley & Sons, 1982.
37. B. Golden, L. Bodin, T. Doyle, and W. Stewart, Jr. Approximate traveling salesman algorithms. *Operations Research,* 28:694–711, 1980.
38. R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics,* 5:287–326, 1979.
39. J. N. D. Gupta. Optimal schedules for single facility with two job classes. *Computers and Operations Research,* 11(4):409–413, 1984.
40. J. N. D. Gupta. Single facility scheduling with multiple job classes. *European Journal of Operational Research,* 8:42–45, 1988.
41. M. Held and R. M. Karp. A dynamic programming approach to sequencing problems. *Journal of the Society for Industrial and Applied Mathematics,* 10(1):196–210, 1962.
42. W. A. Horn. Single-machine job sequencing with treelike precedence ordering and linear delay penalties. *SIAM Journal on Applied Mathematics,* 23(2):189–202, 1972.
43. J. R. Jackson. Scheduling a production line to minimize maximum tardiness (Management Science Project, Research Report 43). Technical report, University of California, Los Angeles, 1995.
44. A. Janiak and M. Kovalyov. Single machine scheduling subject to deadlines and resource dependent processing times. *European Journal of Operational Research,* 94:284–291, 1996.
45. J. Józefowska and J. Węglarz. Discrete-continuous scheduling problems-mean completion time results. *European Journal of Operational Research,* 94:302–309, 1996.
46. F. M. Julien, M. J. Magazine, and N. G. Hall. Generalized preemption models for single machine dynamic scheduling problems. *IIE Transactions,* 29:359–372, 1997.
47. S. Karvonen and J. Holmstrom. How to use tooling analysis to reduce setup time and improve throughput. *Production and Inventory Management Journal,* 1996, 75–80.
48. J. Labetoulle, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Preemptive scheduling of uniform machines subject to release dates. In W. Pulleybank, Ed., *Progress in Combinatorial Optimisation,* Academic Press, Canada, 1984, 245–262.
49. B. J. Lageweg, J. K. Lenstra, E. L. Lawler, and A. H. G. Rinnooy Kan. Computer-aided complexity classification of combinatorial problems. *Communications of the ACM,* 25(11):817–822, 1982.
50. E. L. Lawler. Optimal sequencing of a single machine subject to precedence constraints. *Management Science,* 19(5):544–546, 1973.
51. E. L. Lawler. Sequencing jobs to minimize total weighted completion time subject to precedence constrainsts. *Annals of Discrete Mathematics,* 2:75–90, 1978.
52. E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, Eds., *The Traveling Salesman Problem.* John Wiley & Sons, 1985.
53. E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys. Sequencing and scheduling: algorithms and complexity. In *Handbooks in Operations Research and Management Science, Volume 4: Logistics of Production and Inventory,* chap. Elsevier Science, 1993, chap. 9, 445–522.
54. J. K. Lenstra and A. H. G. Rinnooy Kan. Complexity of scheduling under precedence constraints. *Operations Research,* 26(1):22–35, 1978.
55. J. K. Lenstra, A. H. G. Rinnooy Kan, and P. Brucker. Complexity of machine scheduling problems. *Annals of Discrete Mathematics,* 1:343–362, 1977.
56. C.-J. Liao and L.-M. Liao. Single facility scheduling with major and minor setups. *Computers and Operations Research,* 24(2):169–178, 1997.
57. J. D. C. Little, K. G. Murty, D. W. Sweeney, and C. Karel. An algorithm for the traveling salesman problem. *Operations Research,* 11:972–989, 1963.
58. A. J. Mason. *Genetic Algorithms and scheduling problems.* PhD thesis, University of Cambridge, 1992.
59. A. J. Mason and E. J. Anderson. Minimizing flow time on a single machine with job classes and setup times. *Naval Research Logistics,* 38:333–350, 1991.

60. D. C. Mattfeld. *Evolutionary Search and the Job Shop.* Psysica-Verlag, Heidelberg, 1996.
61. Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs,* 3rd ed., Springer-Verlag, 1996.
62. C. L. Monma and C. N. Potts. On the complexity of scheduling with batch setup times. *Operations Research,* 37(5):798–804, 1989.
63. T. E. Morton and D. W. Pentico. *Heuristic Scheduling Systems.* John Wiley & Sons, 1993.
64. E. Nowicki and S. Zdrzalka. Single machine scheduling with major and minor setup times: a tabu search approach. *Journal of the Operational Research Society,* 47:1054–1064, 1996.
65. S. S. Panwalkar, R. A. Dudek, and M. L. Smith. Sequencing research and the industrial sequencing problem. In S. E. Elmaghraby, Ed., *Symposium on the Theory of Scheduling and its Application,* Springer-Verlag, 1973, 29–38.
66. M. Pinedo. *Scheduling, Theory, Algorithms and Systems.* Prentice-Hall, 1995.
67. C. N. Potts. Scheduling two job classes on a single machine. *Computers and Operations Research,* 18(5):411–415, 1991.
68. C. N. Potts and L. N. Van Wassenhove. Integrating scheduling with batching and lot-sizing: a review of algorithms and complexity. *Journal of the Operational Research Society,* 43(5):395–406, 1992.
69. H. N. Psaraftis. A dynamic programming approach for sequencing groups of identical jobs. *Operations Research,* 28:1347–1359, 1980.
70. G. Reinelt. *The Traveling Salesman: Computational Solutions for TSP Applications.* Lecture Notes in Computer Science. Springer-Verlag, 1994.
71. A. H. G. Rinnooy Kan. *Machine Scheduling Problems: Classification, Complexity and Computations.* Nijhoff, The Hague, 1976.
72. V. K. Sahney. Single-server, two-machine sequencing with switching time. *Operations Research,* 20(1):24–36, 1972.
73. V. K. Sahney. Errata (to single-server, two-machine sequencing with switching time). *Operations Research,* 22:1120, 1974.
74. J. B. Sidney. Decomposition algorithms for single-machine sequencing with precedence relations and deferral costs. *Operations Research,* 23(2):283–298, 1975.
75. W. E. Smith. Various optimizers for single stage production. *Naval Research Logistics Quarterly,* 3:59–66, 1956.
76. D. R. Sule. Sequencing n jobs on two machines with setup, processing and removal times separated. *Naval Research Logistics Quarterly,* 29(3):517–519, 1982.
77. C. H. White and R. C. Wilson Sequence dependent set-up times and job sequencing. *International Journal of Production Research,* 15(2):191–202, 1977.
78. D. N. Williams and A. Wirth. A new heuristic for a single machine scheduling problem with setup times. *Journal of the Operational Research Society,* 47:175–180, 1996.

3

Computer Aided 5-Axis Machining

Andrew Warkentin
University of Dalhousie

Paul Hoskins
CAMplete Solutions, Inc.

Fathy Ismail
University of Waterloo

Sanjeev Bedi
University of Waterloo

3.1 Introduction
Classification of NC Milling Machines • The Need for 5-Axis Machining • Components of a 5-Axis Machine

3.2 Kinematics of a 5-Axis Milling Machine
Forward Kinematics of a Tilt-Rotary Table Type 5-Axis Milling Machine • Inverse Kinematics of a Tilt-Rotary Table Type 5-Axis Milling Machine • Forward Kinematics of a Wrist Type 5-Axis Milling Machine • Inverse Kinematics of a Wrist Type 5-Axis Milling Machine

3.3 Surface Machining
Tool Path and Trajectory Planning • Tool Positioning • Tool Path Simulation, Verification, and Correction

3.4 Conclusion

5-Axis machining offers several advantages over the traditional 3-axis machining in producing complex sculptured surfaces; it reduces the machining time and improves the surface finish. This chapter provides a comprehensive review of the main issues in this new technology. It presents a unified approach to describing the kinematics of typical 5-axis machine configurations. It also provides a state-of-the-art review of current research issues, including tool path planning, tool positioning strategies, tool path simulation and verification, and gouge detection and correction.

3.1 Introduction

Computer numerically controlled (CNC) machining is utilized extensively in producing parts with complex sculptured surfaces. It is used indirectly to produce these parts, by machining the dies and molds in which they are drawn or cast, or directly in the production of specialized parts such as turbine blades. In all cases, the workpiece surface data, generated in a computer aided design (CAD) package is passed to a computer aided manufacturing (CAM) package or module to generate the tool path. Traditionally, these surfaces have been produced on 3-axis machines using ball nose cutters. It has been demonstrated by many researchers, including the present authors, that switching from 3-axis to 5-axis technology can result in substantial savings in machining time, coupled with improved surface finish.

Because of the potential for improved productivity, the interest in 5-axis machining has escalated in the past decade. Yet the authors are not aware of any single publication that offers a comprehensive review of this subject. This chapter is an attempt to accomplish this task. Our knowledge in the field of 5-axis machining has been gained through working experience in the field and reading as many as possible of

the numerous publications in the literature. The list of references at the end of the chapter is but a small sample of the available body of work.

The first part of this chapter provides the reader with the essentials to better understand 5-axis machining technology. It includes the salient features that distinguish 5-axis from 3-axis machining. The chapter also gives a uniform presentation of the kinematics of the two most common 5-axis machine configurations. Knowledge of these kinematics is crucial to the proper implementation of the concepts presented in this chapter and those in the literature.

The second part of the chapter reviews some of the cutting-edge research in 5-axis surface machining. These include new tool path planning and tool positioning strategies to improve surface finish and reduce production time, techniques for tool path simulation and verification, and algorithms for gouge detection and avoidance. In particular, it highlights the recent efforts at the University of Waterloo by the present authors in the above areas.

Classification of NC Milling Machines

CNC milling machines are usually described by the number of axes of motion. A 3-axis milling machine is capable of moving a cutting tool in three directions relative to the workpiece. Through the use of ball nose end mills and special fixtures, these machines are very flexible and can be used for low- and high-volume manufacturing. A 5-axis milling machine can position a tool in three-dimensional space and control its orientation. They are especially useful for low-volume, high-complexity parts. Moving from 3-axis to 5-axis machine tool technology means much more than adding two rotational axes to a 3-axis machine. 5-axis machining requires a considerable investment in the cost of the 5-axis machine itself. Equally important is the extra training required for the personnel who program and operate these complex machines.

The investment of moving from 3-axis machining technology to 5-axis technology may seem daunting at first. However, 5-axis machining provides flexibility and efficiency that cannot be obtained with 3-axis milling. A 5-axis machine can produce parts with more complex geometry using a single setup without the need for complex and expensive fixtures. Such machines can produce special geometry, eliminating the use of specialized cutters often used in 3-axis machining. A 5-axis machine can produce many parts which are impossible to produce otherwise, such as the highly twisted impellers found in aircraft turbine engines. Most importantly, 5-axis machines are substantially better at producing free-form surfaces than 3-axis machines.

The Need for 5-Axis Machining

A 5-axis machine gives the cutting tool access to most features of a part without changing the machine setup, as shown in Fig. 3.1. This is commonly known as 5-sided machining [1]. For example, a 5-axis machine can drill holes in the sides of a part by simply rotating the part. This eliminates specialized

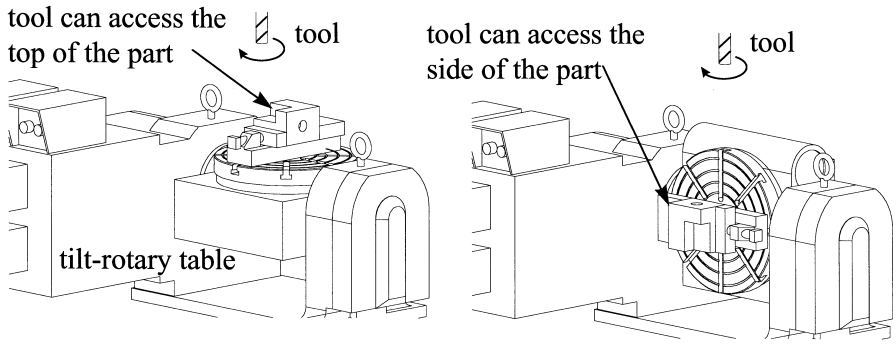


FIGURE 3.1 Five-sided machining.

fixtures or setup changes which would be required to produce such parts on a 3-axis machine. In the automotive industry, the extra positioning capability can be used to produce parts with odd geometry, such as cylinder port ducts and engine blocks. 5-axis machines can also produce flat surfaces at arbitrary orientations by tilting a standard flat end mill. A 3-axis machine would require a special form cutter to produce the same geometry.

5-axis machining originated in the aerospace industry where 5-axis machines produce many of the twisted ruled surfaces that appear in jet engine parts such as impellers. These surfaces can be machined efficiently in one setup on a 5-axis machine using the side of the tool (the flank) rather than the bottom of the tool. Thus, it is often called flank milling. A 3-axis machine would have to use the tool tip (point milling) to produce the same surface and may require several different workpiece setups to machine the entire blade. The improved accuracy of the product and substantial decrease in the production costs more than justify the additional cost associated with 5-axis machining.

5-axis machines provide a significant advantage in free-form (sculptured) surface machining. In traditional 3-axis machining, a ball nose end mill machines the surface using many closely spaced passes of the cutting tool. Each pass of the tool leaves a cylindrical impression on the workpiece. These impressions cannot reproduce a free-form surface exactly. To machine these impressions, the ball nose end mill performs much of its cutting near the bottom center of the tool where the tangential speed of the cutting surface is lowest, producing a poor surface finish. After 3-axis machining, hand grinding and polishing are usually required to achieve a smooth surface. A 5-axis machine can tilt a flat end mill with respect to the surface, which leaves an elliptical-shaped impression on the workpiece. By changing the angle of tilt, the dimensions of this impression can be altered to provide a better approximation of the intended surface. In addition, cutting takes place at the edge of the tilted tool where the cutting speed is the highest, producing a better surface finish than a ball nose cutter.

Reducing the finish machining time through fewer passes of the cutting tool reduces the overall cost of production in this very significant area in modern manufacturing. These free-form surfaces are a common component of modern consumer goods, which are usually produced using tools and dies. The manufacture of dies and molds is a $20 billion industry in the United States alone. The production of a single small die can take up to 1300 hours and, of this time, about 66% is spent on CNC finish machining and hand grinding and polishing. In this area, 5-axis machining is vastly superior to 3-axis machining because it requires less passes of the cutting tool and produces a superior surface.

Components of a 5-Axis Machine

The four integral components of a 5-axis machining system are: the type of milling machine, the machine controller, the CAD/CAM software, and the personnel. Not all 5-axis machining systems are suitable for all tasks. The selection and proper utilization of equipment is critical to achieving the gains in efficiency available through 5-axis machining.

Configurations of a 5-Axis Milling Machine

The variety of 5-axis machine configurations all have their own advantages. The most important issues to consider are the rigidity of the system, the work volume, and the system accuracy. In addition, features such as automatic tool changers, and feed and spindle speeds are as important in 5-axis machines as they are in 3-axis machines.

Rigidity is desirable in all milling machines because it increases positioning accuracy and permits higher metal removal rates. Rigidity can be a problem with 5-axis machines because rotational joints are usually more flexible than linear sliding joints. Generally, a more rigid machine will be more expensive. Determining the required rigidity depends on the types of material being cut and the size of cuts being performed.

The working volume of the machine is also important. It is defined by the range of the joint motions. This range of motion determines the maximum workpiece size and affects accessibility of the tool to certain features on the workpiece.

Accuracy is critical in a 5-axis machine. Positioning errors in the rotational axes are magnified, depending on distance from the center of rotation. As a result, the error associated with the machine varies according to the position in the work volume. This phenomenon makes it difficult to predict the expected accuracy of a workpiece. Many 5-axis machines are actually retrofitted 3-axis machines. Misalignment of the additional rotational axes can severely impact the accuracy of the resulting machined surface.

The Machine Controller

A CNC machine controller controls the machine, runs G-code programs, and provides the user interface between the machine and the operator. The controller should be capable of full 5-axis simultaneous motion. This means that during a cutting motion, all five axes travel in a linear fashion from position to position. For example, at the halfway point of the motion, all axes should be halfway. This ensures that the machine will cut in a smooth, predictable fashion. Some controllers are only capable of 5-axis positioning. In these cases, the controller will not perform linear interpolation and the axes will not move simultaneously. This makes the controller simpler, but reduces the ability of the machine to perform such operations as free-form surface machining or flank milling. The controller should also be able to process data as fast as possible. For 5-axis machining of a free-form surface, many closely spaced tool positions may be required. This could mean that over a stretch of a few millimeters, the controller may have to change the machine position 100 times. If the controller cannot process these positions fast enough, it will have to reduce the feed rate. This will slow down the machining process and alter the surface finish. Controllers should also be able to lock the positions of axes for heavy cuts during 5-sided machining operations.

CAD/CAM Software

CAD/CAM software provides the interface between the human user and the CNC machine. These machines are programmed with the required tool trajectory using a special command set called G-codes. These G-codes are a *de facto* standard in the CNC machine industry. G-code programs can be written manually for simple parts. However, in most case CAM software is used to produce G-code programs directly from CAD models. A CAM package typically produces a G-code program in two stages. First, tool paths consisting of generic cutter locations (CLDATA) are generated. The CLDATA consists of a list of tool positions in the workpiece coordinate system. The cutter locations must then be converted into G-code programs using a post-processor specific to the NC machines that will produce the part.

The selection of a suitable CAD/CAM package for 5-axis machining is important. Many CAM packages are geared to two and a half dimensions. Such packages can perform simultaneous motion in the *xy* plane but only positioning motion in the *z* direction. This is adequate for simple machined parts. However, these packages cannot produce free-form surfaces. Many CAM packages claim to be capable of 5-axis machining. However, the available features of these systems vary greatly. Most of these packages can perform 5-axis positioning that is suitable for 5-sided machining. However, they cannot perform 5-axis free-form surface machining. Some CAM packages can machine free-form surfaces in 5 axes by placing the cutter tip on the surface and aligning the tool axis with the surface normal. For concave surface areas, this will cause gouging of the tool into the design surface. These packages may allow offsetting of the tool to limit gouging but at a loss in surface accuracy. The most sophisticated CAM packages can perform tilted end milling, where the tool is tilted at an angle from the surface normal and the tool tip is placed in contact with the surface. This provides more efficient metal removal. However, the current state of 5-axis machining technology available in CAM packages is lagging behind many advanced 5-axis machining techniques presented in the literature.

In addition to CAD/CAM packages for producing tool paths, numerical verification software plays an important role in 5-axis machining. The complex nature of 5-axis machine tool motions can be simulated to detect possible interference problems between the machine, tool, workpiece, and fixtures. Software can also be used to simulate the material removal process. This is particularly useful for detecting gouges.

When gouging of the design surface is detected, the tool position must be changed to eliminate the gouge. Material removal simulation can also be used to estimate the scallop size that will be left by the cutting process. This helps in the determination of the optimum cutter size and cross feed.

Training Personnel

The final component of a 5-axis machining system is the personnel who will program and operate the machine. 5-axis machining is more complex than 3-axis machining and requires a good understanding of the machine motion. The production of parts using 5-axis machines requires good three-dimensional skills for visualizing and building appropriate tool paths. Thorough training in the operation of the CAM software is important because of the multiple approaches to 5-axis machining, especially in the areas of surface machining. Special care must be taken when setting up the machine to correctly determine tool lengths and workpiece placement relative to the rotating joints.

3.2 Kinematics of a 5-Axis Milling Machine

The benefits of 5-axis machining arise from the ability of the machine to position the cutting tool in an arbitrary orientation with respect to a workpiece. This ability is exploited by the authors of numerous papers. The reader is typically shown an illustration of a tool floating in space above a workpiece. The mechanics of actually placing a tool in the desired location is in many cases not discussed. This phenomenon is largely due to the nature of 5-axis machining and to the nature of research. 5-axis machining is highly dependent on the configuration of the target machine. Almost every 5-axis CNC machine requires a different post-processor to account for the effect of the machines rotational axes. In fact, the post-processor requires information about the workpiece setup and tooling before it can convert generic cutter location data into specific machine-dependent G-code. Even after post-processing, the same tool path executed on different CNC machines will produce noticeably different results. For these reasons, this section on machine kinematics has been included in this chapter.

5-axis milling machines are classified by the combination and order of their linear (T) and rotational (R) axes. For example, a machine with three translations and two rotations would be specified as a TTTRR machine. There are many possible combinations of these axes that can be used to produce a 5-axis milling machine. However, as Kiridena [26] points out, there are in fact only three commonly used machine configurations:

1. RRTTT: a tilt-rotary table mounted on three linear axes usually referred as a the tilt-rotary type 5-axis machine
2. TTTRR: three linear axes with the cutter oriented by two rotary axes, commonly called a wrist type or Euler type 5-axis machine
3. RTTTR: a rotary table mounted on three linear axes and a single rotary axis for the tool

These three types of 5-axis configurations are illustrated in Fig. 3.2. The other possible configurations such as TRTTR are generally not used because of the difficulty in designing a machine with a mixture of rotational and linear axes that meets the stiffness requirements for a milling machine. Each of the configurations shown has its own advantages and disadvantages. The wrist type machines are the simplest to program, can be built to accommodate very large workpieces, but tend to be less rigid than the other configurations. They are best suited to surface machining. Tilt-rotary table type machines excel at 5-sided machining and tend to be stiffer than other configurations. However, they are more prone to setup error and may not be able to accept large workpieces.

When programming a CNC machine, the motion of each joint must be specified in order to achieve the desired position and orientation of the tool relative to the workpiece. This is referred to as the inverse kinematics problem. The programmer will also need to know the resulting position and orientation of the tool for a given set of joint commands in order to verify that the tool path is correct.

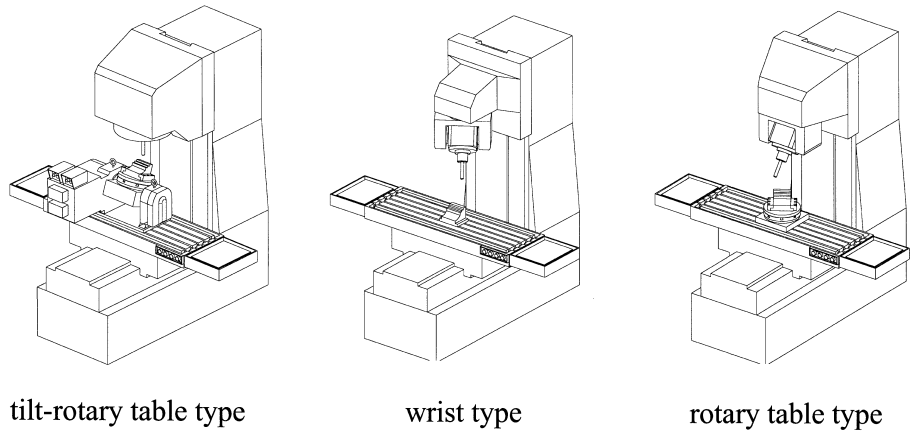


FIGURE 3.2 Typical 5-axis machine configurations.

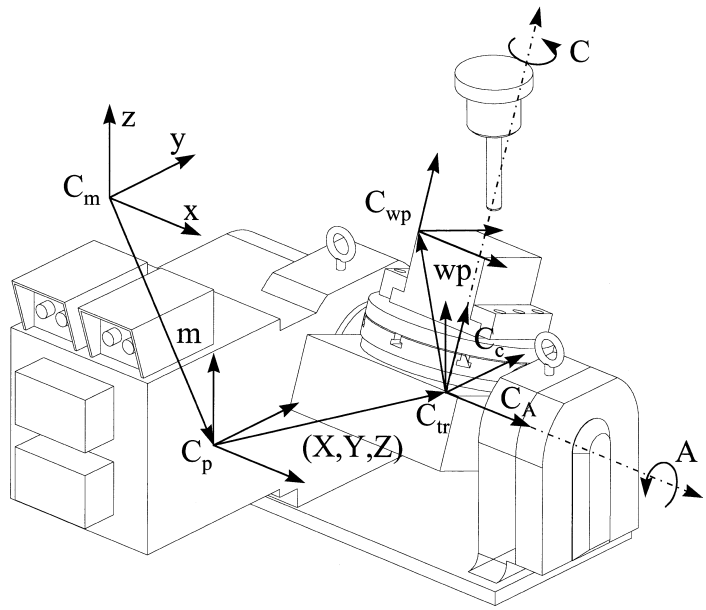


FIGURE 3.3 Kinematics of a tilt rotary table type 5-axis machine.

This is referred to as the forward or direct kinematics problem. The following sections describe the forward and inverse kinematics of the tilt-rotary table type 5-axis milling machine and the wrist type 5-axis milling machine.

Forward Kinematics of a Tilt-Rotary Table Type 5-Axis Milling Machine

When modeling a tilt rotary table type 5-axis machine, it is convenient to consider the coordinate systems illustrated in Fig. 3.3. In this figure, the machine coordinate system, $\mathbf{C_m}$, is fixed to the most positive location in the work volume of the CNC machine tool. All commands sent to the machine are in terms of the machine coordinate system. All other coordinate systems are for human convenience. The programmed coordinate system, $\mathbf{C_p}$, is located by the vector, **m**, relative to the machine coordinate system during the workpiece setup prior to machining. It is essential that this vector be set such that

the center of rotation of the tilt-rotary table is coincident with the tool tip. After this vector is set, a command sent to the controller to move to position (0, 0, 0) in the programmed coordinate system will place the tool tip at the center of rotation. The tilt-rotary table coordinate systems, $\mathbf{C_{tr}}$, and the rotating coordinate systems, $\mathbf{C_A}$ and $\mathbf{C_C}$, are attached to the center of rotation of the tilt-rotary table. Note that there are many different ways to configure a tilt-rotary table, depending on the initial position of the rotary axes. The most basic configurations assume that the table is initially horizontal or vertical. This analysis assumes that the table is initially horizontal. The workpiece coordinate system, $\mathbf{C_{wp}}$, moves with the tilt-rotary table. The workpiece offset vector, **wp**, gives the position of the workpiece coordinate system relative to the tilt-rotary table coordinate systems.

When a cutter location file is post-processed, the post-processor uses the workpiece offset vector, **wp**, to convert the cutter location data into G-codes. Each G-code position command consists of *X*, *Y*, *Z*, *A*, and *C* components that describe the tool position relative to the programmed coordinate system. The tilt-rotary table coordinate systems are translated by the *X*, *Y*, and *Z*, commands relative to the programmed coordinate system, and the workpiece coordinate system will be rotated by the *A* and *C* commands about the x and z axes in tilt-rotary table coordinate systems. The CNC controller converts commands given in the programmed coordinate system to the machine coordinate system using the machine offset vector, **m**.

To model the kinematics of the CNC machine, homogeneous transformations are used to establish the relationship between the defined coordinate systems. For this exercise, a point, $\mathbf{p}^{\mathbf{wp}}$, will be ultimately transformed from the workpiece coordinate system into the machine coordinate. The superscript on the point will refer to the coordinate system in which the point is defined and the subscripts indicate a particular component of the vector. The position of a point, $\mathbf{p}^{\mathbf{wp}}$, in the workpiece coordinate system, $\mathbf{C_{wp}}$, expressed in the tilt-rotary table coordinate system, $\mathbf{C_{tr}}$, is given by:

$$\begin{bmatrix} p_x^{tr} \\ p_y^{tr} \\ p_z^{tr} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(A) & -\sin(A) & 0 \\ 0 & \sin(A) & \cos(A) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(C) & -\sin(C) & 0 & 0 \\ \sin(C) & \cos(C) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & wp_x \\ 0 & 1 & 0 & wp_y \\ 0 & 0 & 1 & wp_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x^{wp} \\ p_y^{wp} \\ p_z^{wp} \\ 1 \end{bmatrix}$$

Next, the point, $\mathbf{p}^{\mathbf{tr}}$, now in the tilt-rotary table coordinate system, $\mathbf{C_{tr}}$, is transformed into the programmed coordinate system, $\mathbf{C_p}$, as follows:

$$\begin{bmatrix} p_x^{p} \\ p_y^{p} \\ p_z^{p} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & X \\ 0 & 1 & 0 & Y \\ 0 & 1 & 1 & Z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x^{tr} \\ p_y^{tr} \\ p_z^{tr} \\ 1 \end{bmatrix}$$

Finally, the point, $\mathbf{p}^{\mathbf{p}}$ in the programmed coordinate system, $\mathbf{C_p}$, is translated into the machine coordinate system, $\mathbf{C_m}$, using the machine offset vector, **m**:

$$\begin{bmatrix} p_x^{m} \\ p_y^{m} \\ p_z^{m} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & m_x \\ 0 & 1 & 0 & m_y \\ 0 & 0 & 1 & m_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x^{p} \\ p_y^{p} \\ p_z^{p} \\ 1 \end{bmatrix}$$

where $\mathbf{p}^{\mathbf{m}}$ is the point in the machine coordinate system. By combining all the transformation matrices together, a point in the workpiece coordinate system, $\mathbf{p}^{\mathbf{wp}}$, can be expressed in the machine coordinate system by:

$$\begin{bmatrix} p_x^m \\ p_y^m \\ p_z^m \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(C) & -\sin(C) & 0 & X + m_x \\ \cos(A)\sin(C) & \cos(A)\cos(C) & -\sin(A) & Y + m_y \\ \sin(A)\sin(C) & \sin(A)\cos(C) & \cos(\mathrm{A}) & Z + m_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x^{wp} \\ p_y^{wp} \\ p_z^{wp} \\ 1 \end{bmatrix}$$

This transformation determines the relationship between a point in the workpiece coordinate system and the machine coordinate system. This relationship is required in order to simulate the metal removal on a specific CNC machine. The post-processor needs the inverse of this relationship to determine the joint positions required to place the tool at the correct location in the workpiece coordinate system.

Inverse Kinematics of a Tilt-Rotary Table Type 5-Axis Milling Machine

The tool path used in 5-axis machining will consist of a set of tool positions, **tpos**, and a corresponding set of tool orientation vectors, **taxis**, in the workpiece coordinate system. The post-processor must convert this information into angular (A, C) and linear (X, Y, Z) components to place the tool in the correct orientation and position relative to the workpiece in the programmed coordinate system. Because the tool orientation in a tilt-rotary type machine is fixed on the z-axis in the programmed coordinate system, the correct orientation is achieved by rotating the workpiece about the A and C axes until the tool orientation vector lines up with the z-axis. In other words, the tool orientation vector, **taxis**, is [0, 0, 1] in the tilt-rotary coordinate system. In this way, the rotations A and C can be found by solving for A and C in the transformation matrix.

$$\begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(C) & -\sin(C) & 0 & 0 \\ \cos(A)\sin(C) & \cos(A)\cos(C) & -\sin(A) & 0 \\ \sin(A)\sin(C) & \sin(A)\cos(C) & \cos(A) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} taxis_x \\ taxis_y \\ taxis_z \\ 1 \end{bmatrix}$$

However, because these are transcendental equations, it is difficult to determine the correct sign of the results. For instance, the first row in the matrix can be used to conveniently solve for the C rotation.

$$C = \pm \tan^{-1}\left(\frac{taxis_x}{taxis_y}\right)$$

Care is needed to determine the correct sign of the result. Furthermore, a 5-axis machine can position and orient the tool correctly in two different ways using a negative or positive A rotation. For example, a tool orientation vector, **taxis**, of [0.577, 0.577, 0.577] can be achieved by A and C rotations of (45°, 54.731°) or (−135°, −54.731°). For these reasons, it is better to calculate the magnitude of the rotations first and then determine the correct signs of the rotations based on the quadrant of the tool orientation vector. The following algorithm determines the correct A and C values assuming the tool orientation vector always points upward and the A rotation is always negative. Using this approach, the angle between the tool orientation vector and the positive z-axis is:

$$a = \cos^{-1}\left(\frac{\sqrt{taxis_x^2 + taxis_y^2}}{\sqrt{taxis_x^2 + taxis_y^2 + taxis_z^2}}\right)$$

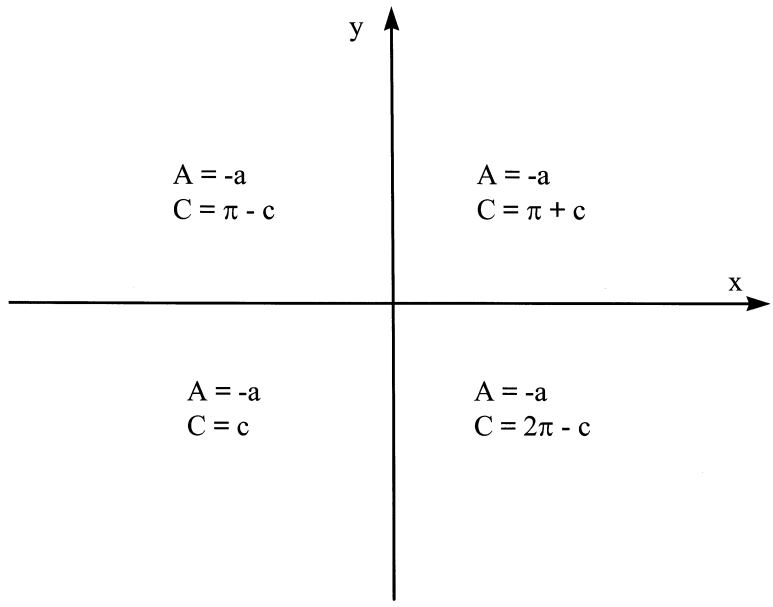


FIGURE 3.4 Selection of quadrant for rotations.

and the angle between the tool orientation vector and the *y*-axis is:

$$c = \cos^{-1}\left(\frac{|taxis_y|}{\sqrt{taxis_x^2 + taxis_y^2}}\right)$$

The correct angles can now be determined based on the quadrant of the tool orientation vector in the workpiece coordinate system as shown in Fig. 3.4.

After the rotations have been determined, the translation can be found by transforming the tool position into the workpiece coordinate system and rotating the resulting vector by *A* and *C*. The resulting displacements are the axis commands *X*, *Y*, *Z* needed to place the tool at the correct location on the rotated workpiece and can be determined as follows:

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(A) & -\sin(A) & 0 \\ 0 & \sin(A) & \cos(A) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(C) & -\sin(C) & 0 & 0 \\ \sin(C) & \cos(C) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & wp_x \\ 0 & 1 & 0 & wp_y \\ 0 & 0 & 1 & wp_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} tpos_x \\ tpos_y \\ tpos_z \\ 1 \end{bmatrix}$$

Forward Kinematics of a Wrist Type 5-Axis Milling Machine

The kinematics of a wrist type 5-axis machine are easier to model because the workpiece coordinate system is never rotated. Figure 3.5 shows the coordinate systems used when modeling a wrist type 5-axis milling machine. As always, the machine coordinate system, $\mathbf{C_m}$, is fixed to the most positive location in the work volume. The programmed coordinate system, $\mathbf{C_p}$, is located by the vector, **m**, such that the center of rotation of the wrist is initially coincident with workpiece coordinate system. The wrist coordinate systems $\mathbf{C_w}$ and the rotating coordinate systems $\mathbf{C_A}$ and $\mathbf{C_C}$ are attached to the wrist's center of rotation. The tool offset vector **t** gives the position of the tool tip relative to the wrist coordinate system.

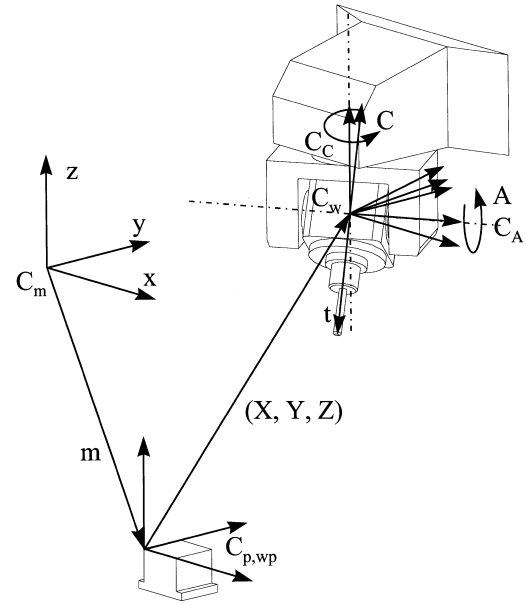


FIGURE 3.5 Kinematics of a 5-axis wrist type milling machine.

Cutter location data will be converted to G-code by the post-processor once the tool length is known. A G-code position command sent to the 5-axis machine controller will consist of *X*, *Y*, *Z*, *A*, and *C* components used to command the joints. The wrist is translated by the *X*, *Y*, *Z*, position command relative to the programmed coordinate system and the tool is rotated about the wrist by the *A* and *C* commands. These commands are converted to the machine coordinate system by the CNC controller.

Using vector algebra, the position of the tool tip, **t**, in the wrist coordinate system, $\mathbf{C_w}$, is given by:

$$\begin{bmatrix} t_x^w \\ t_y^w \\ t_z^w \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(C) & -\sin(C) & 0 & 0 \\ \sin(C) & \cos(C) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(A) & -\sin(A) & 0 \\ 0 & \sin(A) & \cos(A) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} t_x \\ t_y \\ t_z \\ 1 \end{bmatrix}$$

Similarly, the tool tip, $\mathbf{t^w}$, in the wrist coordinate system, $\mathbf{C_w}$, can be transformed into the programmed coordinate system, $\mathbf{C_p}$, as follows:

$$\begin{bmatrix} t_x^p \\ t_y^p \\ t_z^p \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & X \\ 0 & 1 & 0 & Y \\ 0 & 0 & 1 & Z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} t_x^w \\ t_y^w \\ t_z^w \\ 1 \end{bmatrix}$$

Finally, the tool tip, $\mathbf{t^p}$, in the programmed coordinate system, $\mathbf{C_p}$, can be transformed into the machine coordinate system, $\mathbf{C_m}$, by:

$$\begin{bmatrix} t_x^m \\ t_y^m \\ t_z^m \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & m_x \\ 0 & 1 & 0 & m_y \\ 0 & 0 & 1 & m_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} t_x^p \\ t_y^p \\ t_z^p \\ 1 \end{bmatrix}$$

By combining all the transformation matrices together, a tool tip can be expressed in the machine coordinate system by:

$$\begin{bmatrix} t_x^m \\ t_y^m \\ t_z^m \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(C) & -\sin(C)\cos(A) & \sin(C)\sin(A) & X + m_x \\ \sin(C) & \cos(C)\cos(C) & -\cos(C)\sin(A) & Y + m_y \\ 0 & \sin(A) & \cos(A) & Z + m_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} t_x \\ t_y \\ t_z \\ 1 \end{bmatrix}$$

This transformation determines the location of the tool tip in the machine coordinate system.

Inverse Kinematics of a Wrist Type 5-Axis Milling Machine

In the same manner as for the tilt-rotary table, the generic tool paths must be post-processed into suitable joint commands. Again, the vector **tpos** stores the tool tip position and the vector **taxis** stores the tool orientation. For the wrist type machine, the correct orientation is achieved by rotating the tool about the A and C axes until the tool lines up with the tool orientation vector. These values can be determined by solving for A and C in the set of equations given below.

$$\begin{bmatrix} taxis_i \\ taxis_j \\ taxis_k \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(C) & -\sin(C)\cos(A) & \sin(C)\sin(A) & 0 \\ \sin(C) & \cos(A)\cos(C) & -\cos(C)\sin(A) & 0 \\ 0 & \sin(A) & \cos(A) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{t}_x \\ \hat{t}_y \\ \hat{t}_z \\ 1 \end{bmatrix}$$

where $\hat{t}$ is the normalized tool tip vector in its initial position. However, difficulties arise in this method because the equations are transcendental and two possible sets of rotations can be used to position the tool. Instead, the same approach outlined in the section on tilt-rotary table will be used. The angle between the tool negative z-axis in the workpiece coordinate system is:

$$a = \cos^{-1}\left(\frac{\sqrt{taxis_x^2 + taxis_y^2}}{\sqrt{taxis_x^2 + taxis_y^2 + taxis_z^2}} \right)$$

and the angle between the tool orientation vector and the y-axis is:

$$c = \cos^{-1}\left(\frac{|taxis_y|}{\sqrt{taxis_x^2 + taxis_y^2}} \right)$$

The correct angles can now be determined based on the quadrant of the tool orientation vector in the workpiece coordinate system as shown in Fig. 3.6.

After the rotations have been determined, the translation can be found by translating the wrist coordinate system along the tool axis by the tool offset length, **t**, from the tool position in the workpiece coordinate system.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} tpos_x \\ tpos_y \\ tpos_z \end{bmatrix} + |\mathbf{t}| \begin{bmatrix} taxis_x \\ taxis_y \\ taxis_z \end{bmatrix}$$

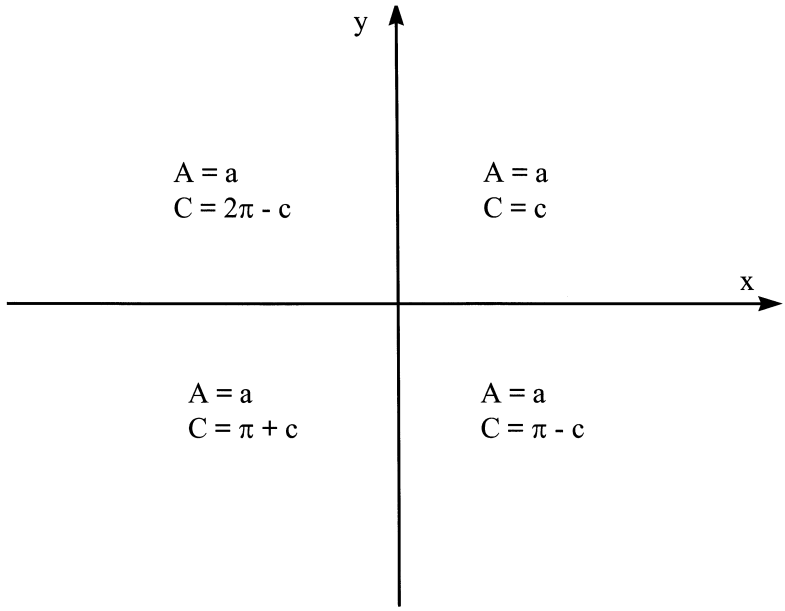


FIGURE 3.6 Selection of quadrant for rotations.

3.3 Surface Machining

Sculptured surfaces are generally produced in three stages: roughing, finishing, and benchwork. Roughing cuts are used to remove most of the material from a workpiece while leaving the part slightly oversized. Finish machining of a sculptured surface removes material from the roughed-out workpiece and attempts to machine the part to its final dimensions. The resulting surface is left with a large number of scallops, as shown in Fig. 3.7. Benchwork consisting of grinding and polishing is used to remove these scallops. The time spent on finishing and benchwork is dependent on the size of these scallops. The scallop size is related to the tool pass interval, also known as the cross-feed; reducing the tool pass interval will decrease the scallop size at the expense of increased machining time. A recent survey by LeBlond Makino of Mason, Ohio [2], stated that a small mold will typically require 57 hours of roughing, 127 hours of finishing, and 86 hours of grinding and polishing. Over 78% of the total production time is spent on finishing, grinding, and polishing. Clearly, there is a need for faster machining techniques that produce smaller scallops, and hence require little or ultimately no benchwork.

The tool path used to machine a surface is generally produced in three stages. First, tool path planning is used to determine the path the tool will take as it machines a surface. Tool path planning research is primarily concerned with the spacing between points on the tool path and determining the tool pass interval. Second, tool positioning strategies are used to determine the cutter location and orientation at specific points on the tool path. The objective of a tool positioning strategy is to minimize the material remaining between the tool and the design surface as the tool moves along the tool path. Finally, gouge detection and correction algorithms are used to determine if the tool has penetrated the desired surface and eliminate this penetration.

Tool Path and Trajectory Planning

An NC tool path used to machine a sculptured surface consists of a set of tool positions. The NC controller interpolates sequentially between these points as the tool moves from point to point. The tool path is usually designed so that the tool makes several tool passes across the surface. The tool pass interval, or cross-feed, affects the size of the scallops. Furthermore, the interpolation between individual tool positions

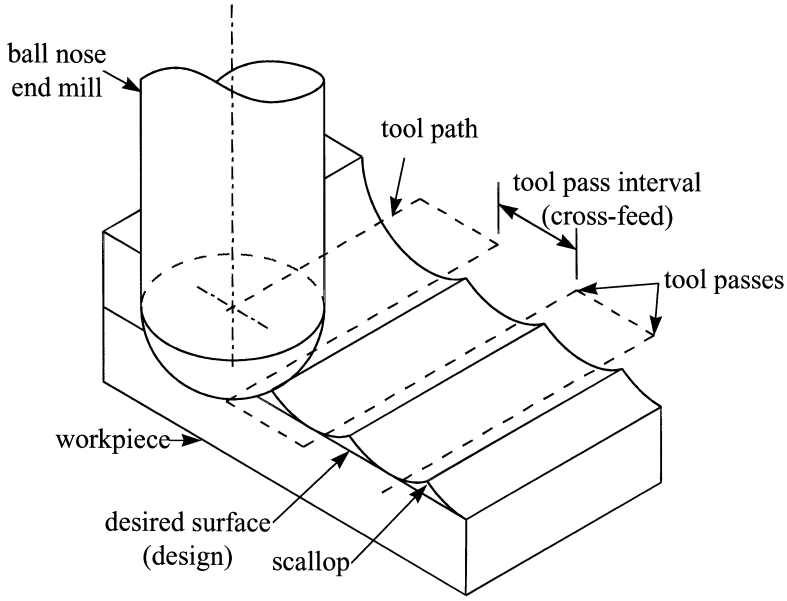


FIGURE 3.7 Scallops left after machining.

on a tool pass may cause the tool to leave the design surface. An ideal tool path will result in a surface with uniform and small-sized scallops evenly distributed across the surface. The size of the scallops will have been determined before machining. In the next section, the tool pass interval will be considered, followed by a discussion of the spacing between individual tool positions along a tool pass.

Tool Pass Interval Determination

The tool pass interval along with the tool type and surface characteristics determine the size of the scallop left on the surface. For the most part, research in this area has focused on tool pass determination for ball-nosed end mills. Very little work has been done in tool pass interval calculations for more complex tool geometries which are becoming more prevalent in 5-axis surface machining. Fortunately, many of the ideas formulated for ball-nosed tools can be extended to other types of tools.

The tool path generation algorithms for parts designed with sculptured surfaces can be broadly characterized as either constant or non-constant parameter techniques. Much of the initial work in tool path planning concentrated on constant parameter tool paths; see, for example, [47]. A constant parameter tool path is usually generated from a parametric surface description of the form shown below:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} f(u, v) \\ g(u, v) \\ h(u, v) \end{bmatrix}$$

where u and v are the parameters of the surface definition. By maintaining one of the parameters constant while varying the other parameter, a tool path consisting of a number of constant parameter curves on the surface can be defined. This approach is computationally efficient because the tool path is easily determined from the surface definition. However, the relationship between the parametric coordinate and the corresponding physical (Cartesian, (x, y, z)) coordinates is not uniform [5]. Therefore, the accuracy and efficiency of a constant parameter tool path may vary with the surface description. Figure 3.8 shows a typical fan-shaped example of this type of tool path. At the top of the part, the tool paths are far apart and will produce a poor surface finish. At the bottom of the part, an unnecessarily large number of tool motions were used to machine the part to a specified surface tolerance.

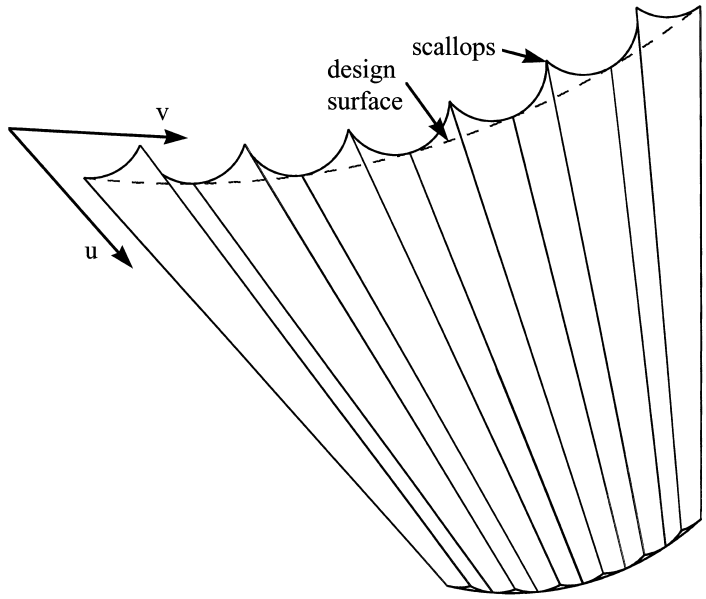


FIGURE 3.8 Constant parameter tool path.

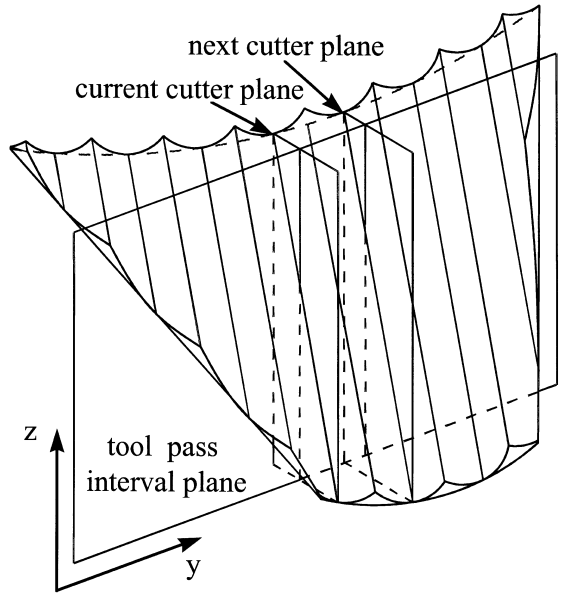


FIGURE 3.9 Non-constant parameter tool path.

Most current research in tool path planning is focused on developing non-constant parameter tool paths with tool pass intervals determined by an estimate of the required scallop height. Choi et al. [7] and Huang and Oliver [17] performed tool path planning in the *xy* plane in the Cartesian coordinate system. Cutting curves are defined by intersections of a group of parallel cutter planes. An example of this type of tool path is shown in Fig. 3.9. This tool path does not suffer from the problem of divergence.

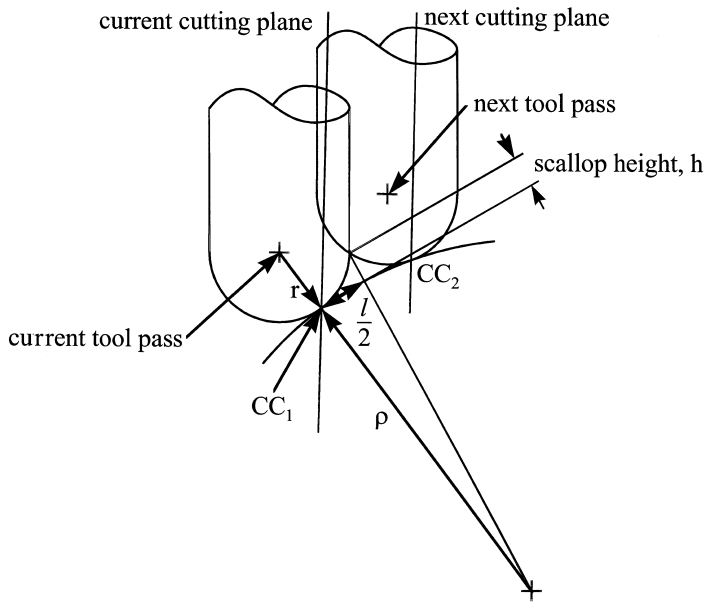


FIGURE 3.10 Tool pass interval calculation for a convex surface.

The tool pass interval was calculated by considering a plane called the tool pass interval plane, which is perpendicular to the cutter planes containing two adjacent tool positions as shown in the Fig. 3.10. The methodology Huang and Oliver [17] use for calculating the pass interval used for a concave surface is briefly explained here. The intersection curve of the surface with the tool pass interval plane can be approximated as a circular arc [8, 48] between the two cutter contact points CC_1, and CC_2 whose radius, ρ, is the radius of curvature of the surface at CC_1 in the tool pass interval plane. From the geometry of the figure, it can be deduced that the tool pass interval, l, is:

$$l = \frac{\rho\sqrt{4(\rho + r)^2(\rho + h)^2 - (\rho^2 + 2rh + (\rho + h)^2)^2}}{(\rho + r)(\rho + h)}$$

where h is the scallop height. The calculation of the tool pass interval is carried out for every cutter contact point on a tool pass. The smallest tool pass interval is used to calculate the next cutter plane. All scallop heights will be within tolerance provided the scallop height approximation is close and the smallest tool pass interval is selected. However, some sections of the surface may have scallops which are smaller than necessary.

Recent work in tool path planning has concentrated on non-constant tool pass interval techniques that maintain a constant scallop height between pass intervals. Tool path planning methods that keep a constant scallop height between tool passes have been developed by Suresh and Yang [49] and by Lin and Koren [33]. Both approaches start with an initial master tool pass. In most cases, the boundary of a surface patch or an iso-parametric curve on the design surface is selected as the master tool pass. Succeeding tool passes are generated by calculating a pass interval that will result in a specified scallop height for every point on the master tool path. This new set of points forms the next tool pass, which in turn becomes the new master tool pass. The main difference between the two techniques lies in the methods used to calculate the pass interval. Using differential geometry, Suresh and Yang [49] have developed a set of equations based on the tool radius, scallop height and surface curvature in the direction perpendicular to the tool pass cutting plane to calculate the pass interval for a specified scallop height. Their method results in a set of complex equations that must be solved numerically. Lin and Koren [33] base their tool pass

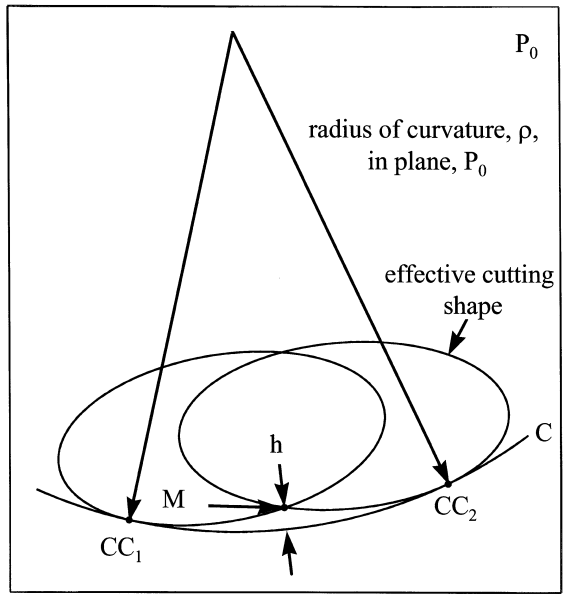


FIGURE 3.11 Scallop height estimation when surface machining in 5-axis with a flat end mill.

calculation on the work of Vickers and Quan [53]. The resulting expression approximates the tool pass interval based on the desired scallop height, the tool radius, and the curvature at the cutter contact point.

Most of the research into tool pass interval calculation has focused on surface machining using a ball-nosed cutter. However, there is still a need for constant scallop height tool paths for other types of tools for use in 5-axis machining. The main stumbling block is a lack of methods for calculating scallop heights for these more complicated situations. Bedi et al. [4] address the issue of scallop height estimation with a toroidal and flat end mill. The authors note that a toroidal cutter can be approximated by a ball nose end mill of the appropriate radius at the cutter contact point. They then use the scallop height expressions developed by Vickers and Quan [53] to approximate the scallop heights produced by flat and toroidal end mills.

Lee and Chang [29] address the more difficult problem of scallop height estimation during 5-axis surface machining with a flat end mill. The authors approach the problem by considering tool positions on adjacent tool passes. The cutting edge of the tool at positions CC_1 and CC_2 is projected onto a plane, P_o, normal to the feed direction containing both cutter contact points, as illustrated in Fig. 3.11. Because the cutting edge of an end mill is a circle, the projected shape will be an ellipse. Lee and Chang [29] refer to this ellipse as the effective cutting shape. The intersection point, *M*, between the two effective cutting shapes is first found. The intersection curve, *C*, between the design surface and the plane, P_o, is approximated as a circular arc using the curvature of the surface in the plane. The scallop height, *h*, is found by taking the distance between the point M and the intersection curve. Lee and Chang [29] used this methodology to estimate the scallop height between tool passes to ensure that the scallops were within a specified tolerance.

To date, most research on scallop height has considered tool and surface geometry. The missing ingredient is tool motion. The assumption that the scallop height can be determined based on static tool positions may be reasonable in 3-axis machining where motions are generally piecewise linear. However, tool paths used to machine surfaces tend to be nonlinear. Effective cutting profiles need to be swept along the tool path rather than projected onto a plane.

Spacing of Tool Positions

The second important issue in tool path planning is to determine the spacing between successive tool positions along a tool pass. The NC machine controller receives the tool path as a sequence of joint commands, for example (*X*, *Y*, *Z*, *A*, *C*). The NC controller performs linear interpolation between a set

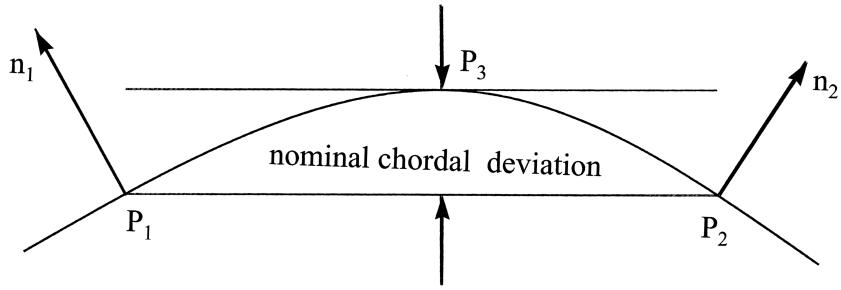


FIGURE 3.12 Nominal chordal deviation.

of (X, Y, Z, A, C) points to form the actual tool path. The tool path will not be a set of line segments because a 5-axis milling machine contains rotational axes. The tool path planner must ensure that the tool remains within tolerance with the design surface as it moves between cutter location points by selecting a small enough spacing between tool positions. If, however, the tool positions are too closely spaced, the time needed to generate the tool path and the storage requirements for the data may be excessive. More importantly, if the rate of data transfer between the NC controller and NC machine is too slow, the milling machine may be forced to wait for the next positioning command. This data starvation problem can result in reduced machining rates and jerky motion.

There has been a considerable body of research devoted to determining the best spacing between tool positions in 3-axis surface machining. The objective of that research is to determine the maximum allowable separation between tool positions that will maintain a required tolerance between the tool and the surface as the tool moves from one position to the next. All available work in that area assumes that the tool moves linearly between successive tool positions. Many tool path planning techniques use the nominal chordal deviation as a measure of the machining error [15, 17] as shown in Fig. 3.12.

A straightforward method of determining the nominal chordal deviation is to calculate P_3, by taking the halfway point between P_1 and P_2 in parametric space and determining the perpendicular distance between P_3 and the line defined by the chord. This method will produce a reasonable approximation of the chordal deviation provided the surface has fairly uniform parametric variation. However, if the parametric variation is non-uniform, the error in this approximation can be significant. Loney and Ozsoy [34] and Wysocki et al. [61] developed numerical techniques for calculating the nominal chordal deviation based on a curve subdivision technique and a cast-and-correct method, respectively.

As Huang and Oliver [17] point out, using the nominal chordal deviation as an estimate of machining error can lead to an underestimation of the true machining error if the normals n_1 and n_2 are not parallel and if both normals are not perpendicular to the chord. The true machining error can only be determined by considering the trajectory of the entire tool as it moves along the path as shown in Fig. 3.13.

As the tool moves from P_1 to P_2, the true machining error occurs between the surface and the common tangent joining both tool positions. The technique developed by Huang and Oliver [17] uses the nominal chordal deviation as an initial estimate of the true machining error. Iterations of the orthogonal projection method [41] are then used to converge to the true machining error.

The trajectory error is the result of the inability of present day NC controllers to do more than linear interpolation between data points. Another approach to solving this problem is to modify the NC controller to perform more sophisticated methods of interpolating between tool positions. Chou and Yang [10] have developed the equations to trace any arbitrary curve in Cartesian space for a wrist type 5-axis milling machine given that the NC programmer can control the velocity and acceleration of each joint continuously in time. They fit the tool path with a polynomial, resulting in a parametric description of the tool path $X(u)$, $Y(u)$, $Z(u)$, $A(u)$, $C(u)$. Expressions for velocity and acceleration in terms of u are then developed to produce the desired tool path for a given feed rate. Chou and Yang have also developed expressions for the jerk that may be of use for predicting vibrations, excessive wear, and tracking errors. They used these equations to simulate the position, velocity, acceleration, and jerk of a tool path described by a 3rd-order parametric space curve.

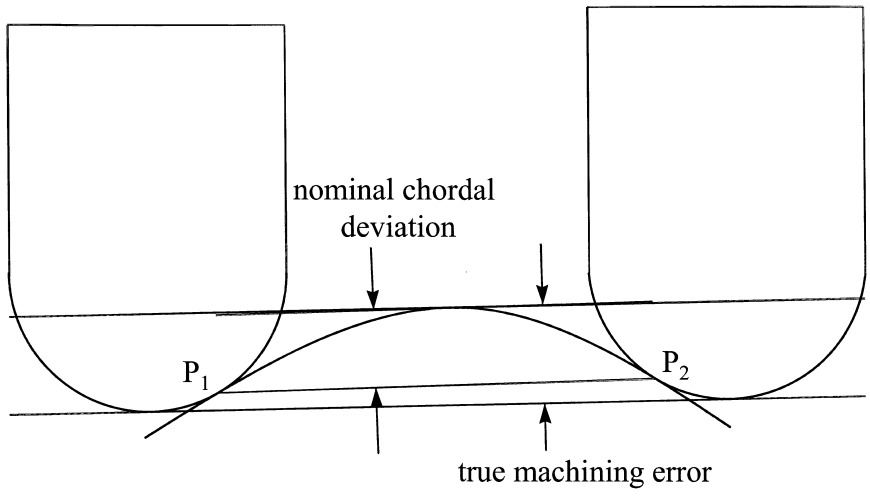


FIGURE 3.13 True machining error vs. nominal chordal deviation.

Other than the work by Chou and Yang [10], we are not aware of any research into trajectory error that takes into account the nonlinear trajectories produced by a 5-axis milling machine as it interpolates between tool positions. At present, 5-axis part programmers must rely on their judgment to determine appropriate spacing between tool positions.

Tool Positioning

Tool positioning strategies are used to determine how a tool will be placed relative to the design surface. The main objective of these strategies is to remove as much material from the workpiece as possible without cutting into the desired surface (gouging) and therefore minimizing the amount of material remaining on the desired surface. Finish machining and benchwork can consume up to 76% of the time needed to produce a surface [2]. Improvements in tool positioning strategies can result in significant cost-savings in the production of complex surfaces.

A number of tool positioning strategies have been developed to improve the placement of different types of tools relative to the design surface such that the material left behind for subsequent polishing is minimized. The most commonly used tools in the machining of molds and dies are shown in Fig. 3.14; they are the ball nose cutter with radius r; the flat bottom end mill of radius R; and the toroidal cutter which is characterized by two radii, the radius of the insert r and the radial distance between the center of the tool and the center of the insert R. As the tools rotate, the cutting surfaces of the above cutters are a sphere, a cylinder, and a torus corresponding to the ball nose, the flat, and the toroidal cutters, respectively. A review of the most relevant tool positioning techniques is presented next.

Ball Nosed

Historically, sculptured surfaces have been machined using ball-nosed end mills. The easiest way to position a ball-nosed end mill is by offsetting the tool center a distance equal to the cutter's radius along the surface normal, as shown in Fig. 3.15. Provided the minimum radius of curvature of the surface is greater than the radius of the ball-nosed end mill, no gouging will occur. The tool orientation has no effect on the geometry of the cutting surface of the ball relative to the design surface. In other words, scallop size reduction cannot be accomplished by changing the orientation of the tool. A surface machined in 3-axis with ball will have the same-sized scallops a surface machined in 5-axis for the same tool pass interval.

However, benefits from 5-axis machining with a ball-nosed tool are still possible when the problems of center cutting and accessibility are considered. Center cutting occurs when the cutting edge near or

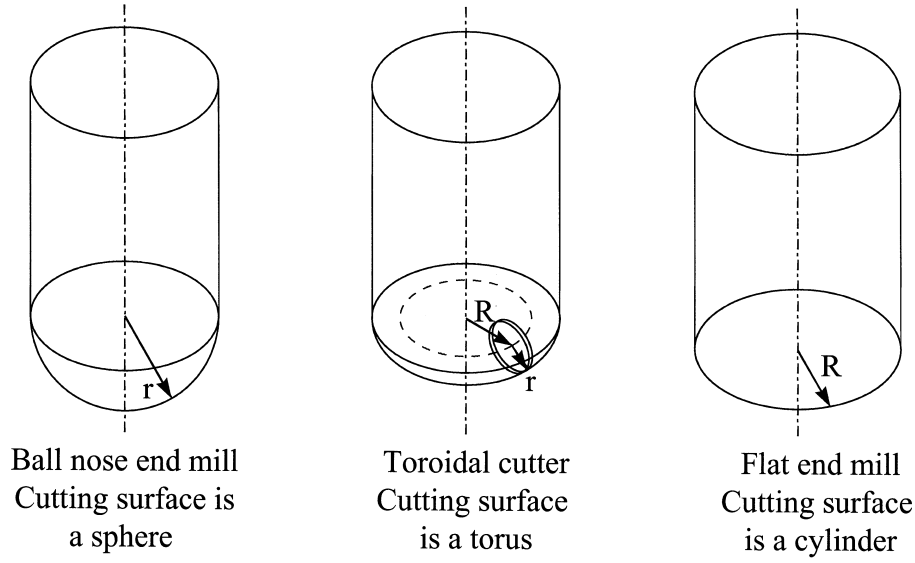


FIGURE 3.14 Cutting tools typically used for 5-axis surface machining.

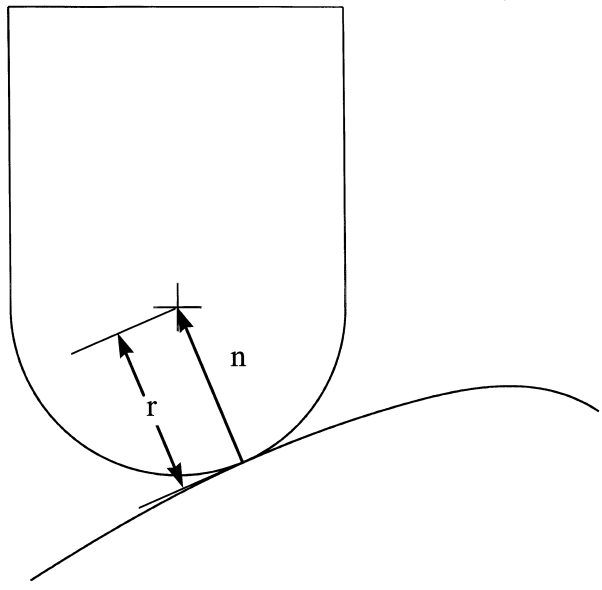


FIGURE 3.15 Positioning a ball nosed end mill on a surface.

at the tool axis is required to cut material. The cutting speed in this region of the tool approaches zero. This results in high cutting forces when operations such as plunging and ramping down are required. In 5-axis, the tool can simply be inclined in the feed direction to eliminate center cutting. In many instances, the tool may not be able to reach some location in a workpiece. For example, in 1991, Tekeuchi and Idemura [50] addressed the problem of accessibility using a solid modeling approach. Their approach assumed a fixed inclination angle for machining. The machining process was then simulated using solid modeling techniques. At every cutter location, interference checking was performed. The operator was prompted to reposition the tool in the appropriate manner if interference was detected. This approach allowed the authors to machine exceedingly complex shapes such as a complete impeller.

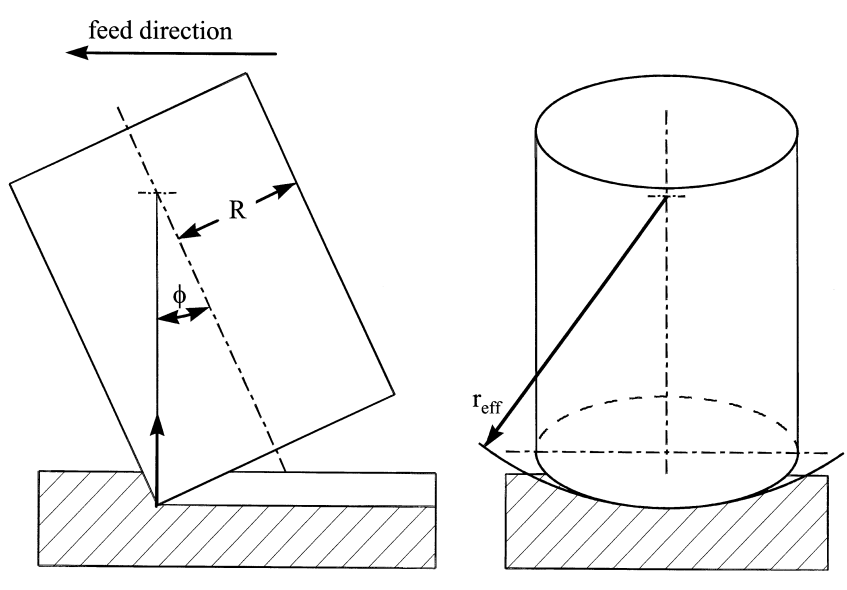


FIGURE 3.16 Machining with an inclined tool.

Inclined Tool

Most current research on 5-axis tool positioning strategies is focused on machining a surface with a flat or toroidal end mill inclined relative to the surface normal as shown in Fig. 3.16. This inclination angle is often referred to as the Sturz angle(ϕ). This approach was made popular by Vickers and Bedi [52] and Vickers and Quan in 1989 [53] when the authors pointed out that an inclined end mill could approximate a surface profile. By varying the inclination of the tool, ϕ, the effective radius, r_{eff}, of the tool could be varied as shown in the following equation.

$$r_{eff} = \frac{R}{\sin\phi}$$

A ball-nosed end mill, on the other hand, only has a constant effective radius. The cross-feed for an inclined tool was calculated in the same manner as a ball-nosed end mill by approximating the inclined end mill as a ball-nosed end mill with a radius equal to the end mill's effective radius. The authors performed a number of cutting tests on flat surfaces and ship hull molds, demonstrating that machining with an inclined end mill was considerably faster than machining with a ball-nosed end mill. One reason for the improvements was a higher effective cutting speed. A flat end mill always cuts at its periphery where cutting speeds are highest. A ball-nosed end mill cuts with a portion of the tool closer to its axis where speeds are lower. This approach has been shown in numerous papers to be highly effective and has been adopted by many high-end commercial CAD/CAM systems [28]. However, the two main drawbacks to this method are the arbitrary method used to select an inclination angle and the use of a constant inclination angle for an entire surface. Subsequent investigations were primarily concerned with these two issues. These investigations are reviewed below.

Cho et al. [6] modified the inclination angle in 1993 using a Z-map technique. In the Z-map technique, the *XY* plane is represented by a discrete set of (x, y) points. The tool and workpiece are represented by discrete z values at each (x, y) location. Interference was checked by detecting if any of the z values for the desired surface were above the z values of the tool. Tool inclination adjustments were performed in semi-automatic fashion by rotating the tool about the cutter contact point based on the weighted average

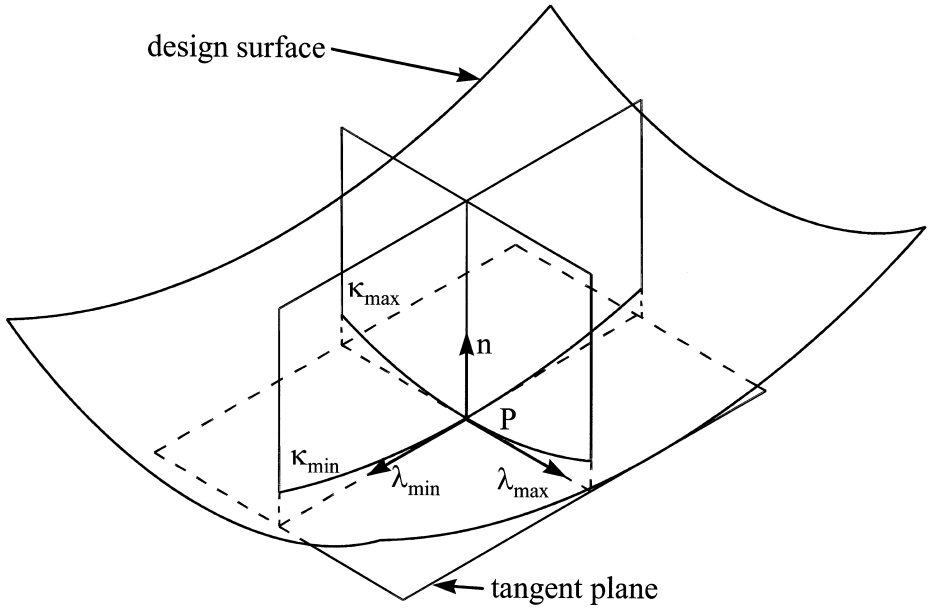


FIGURE 3.17 Curvatures of a point on a surface.

of the interfering points. This process was repeated twice at every cutter location point. If interference still occurred, the programmer was prompted to manually adjust the tool orientation. In 1994, Li and Jerard [30] pointed out that representing solids as discrete sets of point is very inefficient and that high model resolution was not possible because of data size restrictions. Li and Jerard's approach was to represent the tool and workpiece as planner models that could represent the tool and workpiece far more accurately. By considering interference between points, lines, and planes, Li and Jerard were able to adjust the tool orientation automatically.

In 1992, Jensen and Anderson [22] proposed a method for calculating an optimal tool angle based on local surface curvature. The local geometry of a surface near a point, *P*, is characterized by its minimum and maximum curvatures, κ_{min} and κ_{max}, respectively as shown in Fig. 3.17. These curvatures describe circles of radii ρ_{min} and ρ_{max} in two perpendicular planes. The directions of minimum and maximum curvature, λ_{min} and λ_{max}, form a right-handed coordinate system with the surface at *P*. See, for example, Faux and Pratt [13] or Farin [12] for a complete description of the differential geometry of surfaces.

Curvature matching as shown in Fig. 3.18. matches the effective radius of a point on the surface with the radius of maximum curvature. The tool is placed on the surface such that the feed direction lines up with the direction of minimum curvature on the surface. The tool is inclined in the direction of minimum curvature such that the effective radius of the tool at the cutter location equals the minimum radius of curvature of the surface. The authors also noted that the profile of a torus is a 4th-order curve, while the profile of an end mill is only a 2nd-order curve. Therefore, a better match between the surface and a toroidal end mill should be possible.

In 1993, Jensen et al. [23] extended this work to the toroidal end mill and developed a numerical procedure for calculating cross-feeds (see Fig. 3.19). However, the authors made no attempt to machine an actual workpiece. Had they done so, they would have realized that it is not always practical to line up the feed direction with the directions of minimum curvature because these lines tend to follow irregular curves, producing impractical tool paths [28]. Rao et al. [43, 44] developed a similar technique in 1994 that they called the principal axis method (PAM). They used their technique to machine various surface patches and investigated the effect of tool path direction on the technique.

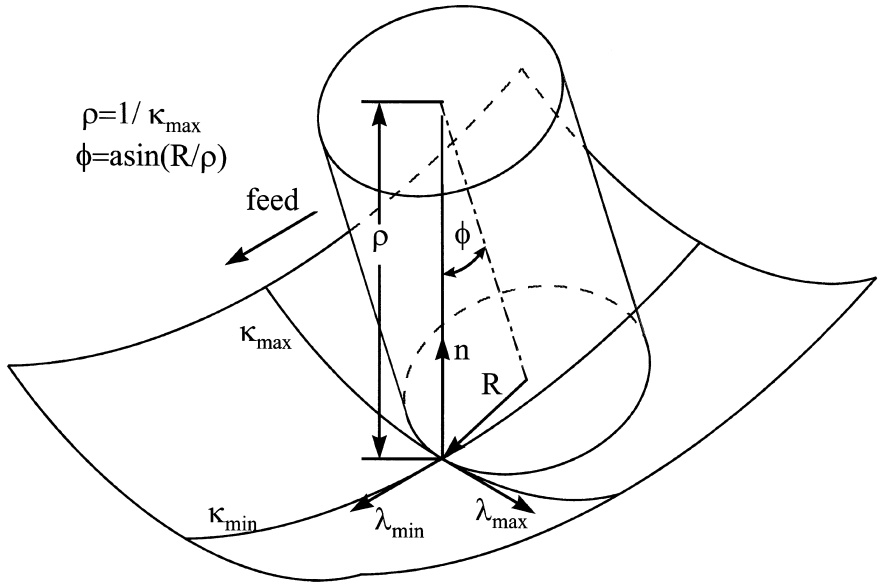


FIGURE 3.18 Curvature matching with a flat end mill.

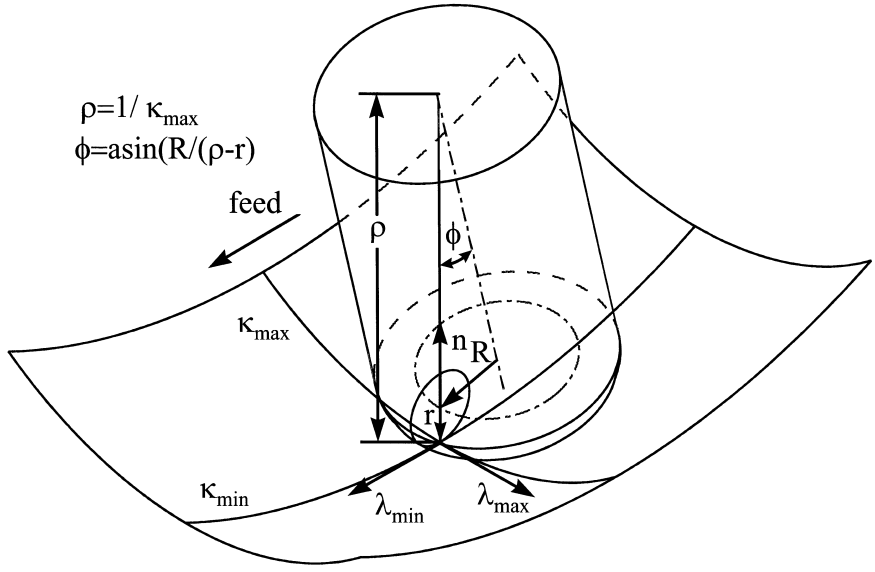


FIGURE 3.19 Curvature matching with a toroidal end mill.

In 1994, Kruth et al. [28] used curvature matching as a first approximation for their tool inclination calculation. The authors recognized the importance of the workpiece global geometry, not just local curvature. Even with curvature matching, gouging can still occur when the surface curvature is changing radically. Kruth et al. checked to see if any portion of the cutting tool was penetrating the desired surface by numerically approximating the distance between the tool and the surface. The tool inclination angle was altered based on the location and depth of gouging.

Another tool positioning strategy called curvature catering was proposed by Wang et al. [57, 58] in 1993 for a cone-shaped tool shown in Fig. 3.20. The authors derived their theory by intersecting the plane formed

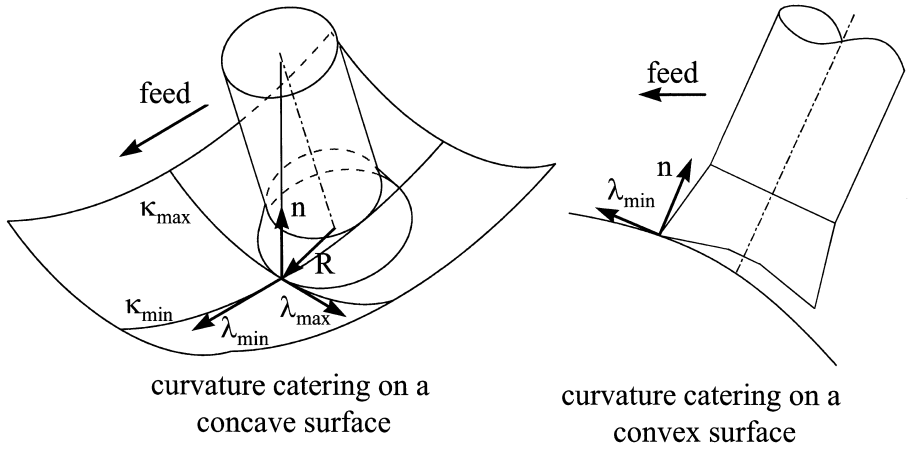


FIGURE 3.20 Curvature catering technique.

by the bottom of the tool and a 3rd-order Taylor's series approximation of the surface. The circle formed by the bottom of the tool and the resulting intersection line is matched as closely as possible. The results of this analysis are exactly the same as those obtained by Jensen and Anderson [22] for a flat end mill. The best inclination angle occurs when the effective radius of the tool is matched to the minimum radius of curvature of the surface. Li et al. [31, 32] deal with the issues of tool path generation, tool pass interval calculation, and gouging when using curvature catering. The main advantage of this technique over curvature matching with a toroidal end mill is the ability to machine convex surfaces as well as concave surfaces.

Multi-point Machining

All the previously discussed tool positioning strategies attempt to maximize metal removal by considering the local geometry of a point on the surface and a point on the tool. In 1995, Warkentin, Bedi, and Ismail [59] proposed a tool positioning strategy called multi-point machining (MPM), which matches the geometry of the tool to the surface by positioning the tool in a manner that maximizes the number of contact points between the surface and the tool. The authors demonstrated the potential of the idea using it to machine spherical surfaces. This idea is best explained using the "drop the coin" concept. A coin is placed in a spherical surface. For stability, the coin must touch the surface tangentially at every point on the coin's circumference. As the coin slides along the surface, the surface is generated along the entire circumference of the coin. If the coin is now replaced by the bottom of a milling cutter, the surface will be generated along a circle of contact between the tool and the surface. The authors were able to machine spherical surfaces with virtually no scallops in a fraction of the time required by conventional machining techniques.

In 1996, multi-point machining was extended to general concave surfaces with a toroidal end mill by Warkentin, Ismail, and Bedi [60]. The intersection theory described by Markot and Magedson [35] and Krieziz [27] was used to examine the nature of multi-point contact between a torus and a concave parametric surface described by the following set of equations. These findings were used to develop an efficient technique to find multi-point contact tool positions.

$$S(u,v) = \begin{bmatrix} S_x \\ S_x \\ S_x \end{bmatrix} = \begin{bmatrix} 80u - 20u^2 \\ 120v - 20v^2 \\ 15 + 25v^2 - 30u - 30uv^2 + 50u^2 + 30u^2v^2 - 30u^2v \end{bmatrix}$$

$$0 \geq u \geq 1$$

$$0 \geq v \geq 1$$

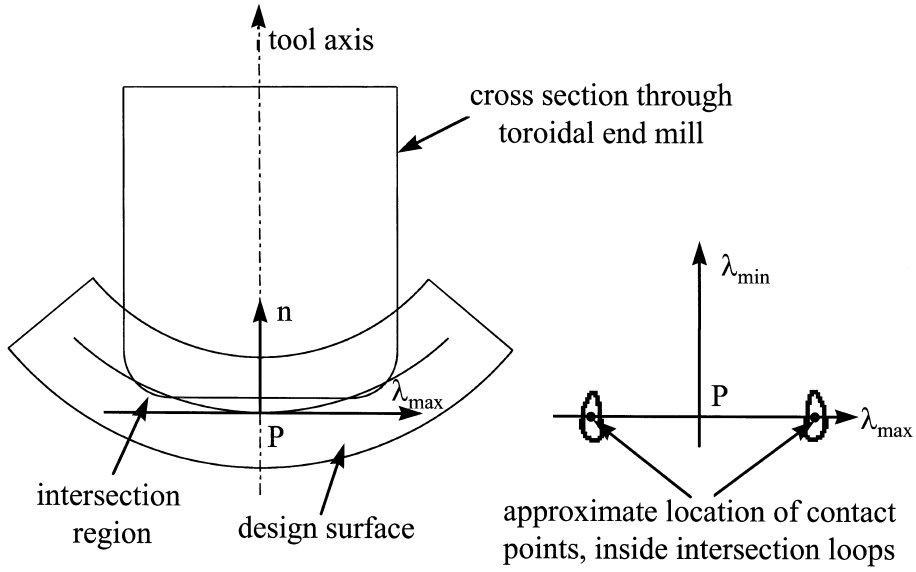


FIGURE 3.21 Location of a multi-point tool position.

A point, *P*, on the surface was selected to form a local coordinate system consisting of the surface normal, *n*, and directions of minimum and maximum curvature, λ_{min} and λ_{max}, at *P*. The tool's cutting surface was represented by a torus, and a concave surface was intersected for different tool positions, as shown in Fig. 3.21. At each trial tool position, the intersection produced loop(s) of intersection or no intersection. Points of tangency are the boundary solution between these two possibilities. Therefore, points of tangency were assumed to occur in the center of small loops of intersection. Multi-point tool positions were located when a trial tool position produced more than one tangent point.

By experimentation it was found that, at most, two contact points existed between the cutting tool and a concave surface. Figure 3.22 shows the pattern of loops of intersection between a torus with $R = 7.9375$ mm and $r = 4.7625$ mm and the test surface as the tool inclination was increased in the direction of minimum curvature at the point $u, v = (0.5, 0.5)$. The center of each loop marks the approximate location of a tangent point. These contact points were arranged symmetrically in pairs about the direction of minimum curvature. The two loops of intersection on the absisca correspond to the case when the tool axis was aligned with the surface normal at point *P*. As the inclination angle increased, the separation between contact points decreased until the two cutter contact points merged into a single cutter contact point.

Figure 3.23 shows the effect of the cutter contact separation on the surface deviations. These surface deviations were produced by determining the difference between the surface and a tool position using the mow-the-grass technique. The mechanics of the mow-the-grass technique are discussed later in the section on verification. The individual surface deviations were then projected onto the *yz* plane. The curves shown in the figure consist of the silhouettes of the projected deviations. These deviation profiles vary significantly from those produced by single-point machining techniques. Typically, single-point machining techniques produce deviation profiles that have a single minimum at the cutter contact point. Multi-point deviation profiles contain as many minima as cutter contact points, with maxima between the cutter contact points. In the present case, two minima and a single maximum produce "W"-shaped deviation profiles. As the cutter contact points separation *w* increases, the maximum deviation from the design surface underneath the tool increases.

The cutter contact separation adds an additional complexity to tool pass interval calculations. The tool path planner must also determine a cutter contact separation that will keep the center portion of the deviation profiles in tolerance with the surface, as illustrated in Fig. 3.24. There are two mechanisms producing scallops in a multi-point tool path. The larger rounded scallops are produced by the area of

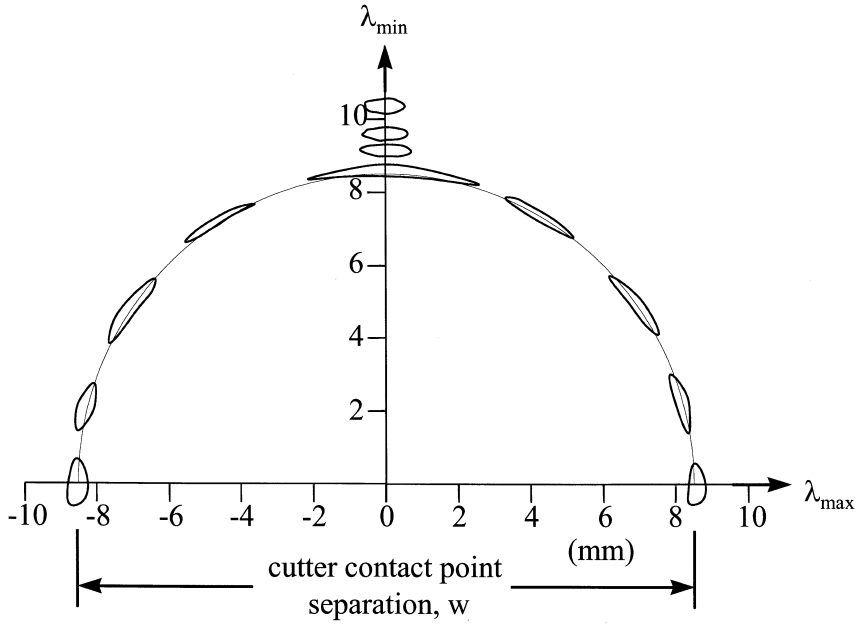


FIGURE 3.22 Arrangement of cutter contact points between a toroidal cutter and a concave surface.

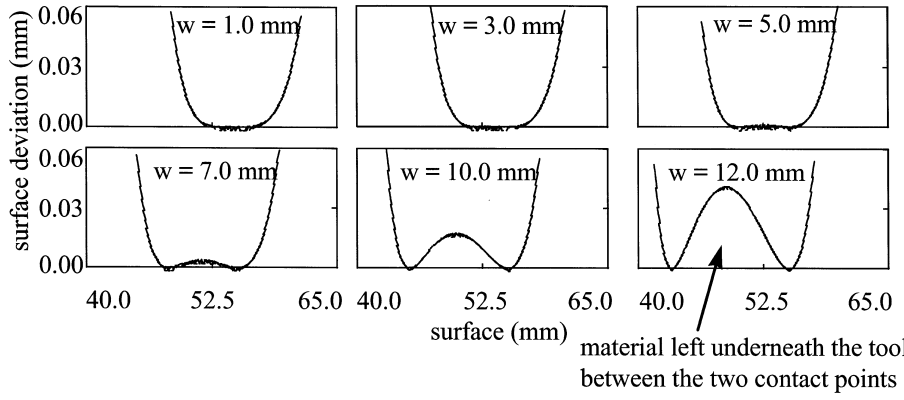


FIGURE 3.23 Effect of cutter contact point separation w on surface deviation for a toroidal end mill with $R = 7.9375$ mm and $r = 4.7625$ mm.

the tool between the contact points, and the smaller sharper deviations are produced by the periphery of the tool. Unfortunately, at the present time, a method for predicting multi-point scallop heights prior to machining or simulation does not exist. Preliminary results suggest that a cutter contact point separation of between 60% and 80% of the tool pass interval will maintain the same height for both types of scallops.

Intersection theory was used to investigate the nature of contact between the tool and the design surface. The results of the investigation led to the development of an algorithm for determining multi-point tool positioning. This algorithm consists of two parts. In the first part, an approximate tool position and orientation is calculated based on the geometry of the two contact points, CC_1 and CC_2. The resulting tool configuration places the tool in tangential contact with CC_1. However, because the curvature of surface under the tool is not constant, there is no guarantee that there is tangential contact at CC_2.

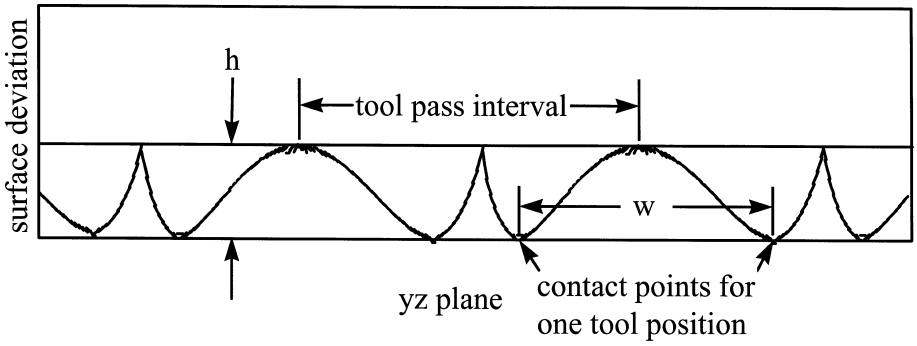


FIGURE 3.24 A multi-point scallop profile.

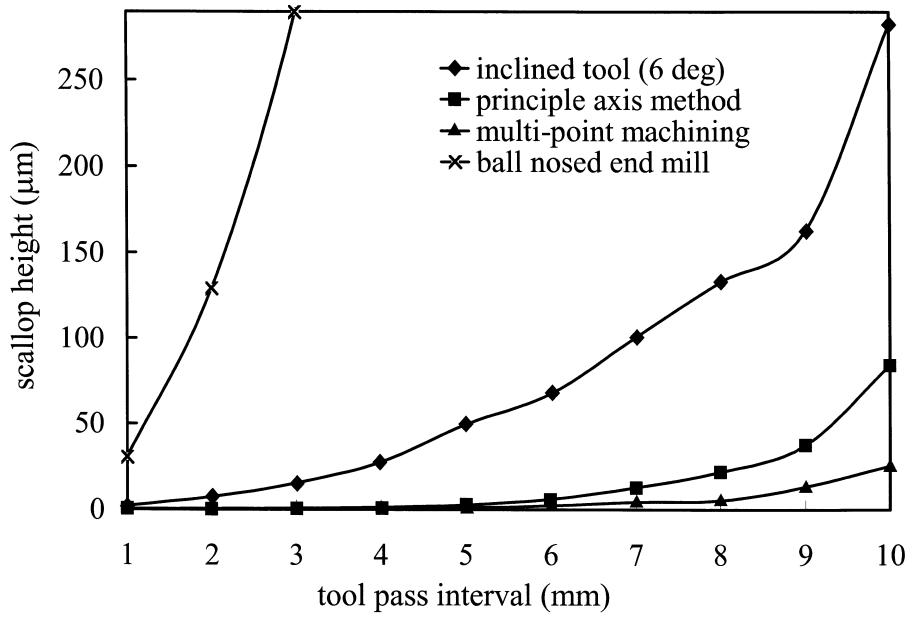


FIGURE 3.25 Comparison of maximum scallop heights produced by different tool positioning strategies.

The second part of the method refines the initial solution. The tool is then rotated while maintaining tangential contact at CC_1 until tangential contact occurs at CC_2.

Machining simulations of the test surface have been performed to compare the performance of the discussed tool positioning strategies. Figure 3.25 summarizes the results of these simulations. A toroidal cutter was used for multi-point, principle axis, and inclined tool machining simulations. The tool dimensions were $R = 5.0$ mm and $r = 3.0$ mm. A 16.0-mm diameter ball-nosed tool was used for the ball-nosed machining simulations. A 6° inclination angle was selected for the inclined tool machining because it was the minimum angle required to avoid gouging the surface. The cutter contact separation for the multi-point machining was 70% of the tool pass interval. Simulations were performed for tool pass intervals ranging from 1 mm to 10 mm in 1-mm increments. At every tool pass interval, ranking of the different strategies in terms of minimum scallop height was conducted and it was found, in descending order, to be: multi-point, principle axis, inclined tool, and lastly machining with a ball-nosed cutter. For example, if the surface tolerance was 0.01 mm, the required tool pass intervals would be 8.7, 6.6, 2.4, and 0.5 mm for the multi-point, principle axis, inclined tool, and ball-nosed techniques, respectively. For this particular surface, multi-point machining is clearly superior.

However, like the principle axis technique, increased performance comes at the expense of increased computational effort and surface requirements. It took 121, 58, and 53 seconds on a 166-MHz Pentium processor to perform the tool positioning computations for the multi-point, principle axis, and inclined tool methods, respectively, for a tool path with maximum scallop heights of 0.01 mm. Ironically, tool positioning for the ball-nosed tool required 256 seconds due to the large number of tool passes required. Furthermore, the multi-point and principle axis techniques require surfaces for which the curvature must be calculated accurately, while the inclined tool and ball-nosed techniques can be implemented with only surface normal information.

Tool Path Simulation, Verification, and Correction

As with any other type of program, NC codes must be debugged to avoid milling errors such as gouging, undercutting, collision with fixtures, etc. The two traditional methods used to check NC code depend heavily on human observation and intuition. The most common method is to perform trial cuts on a soft, inexpensive material under the observation of a skilled technician. The trial workpiece can then be measured manually or with a coordinate measuring machine (CMM) to determine the acceptability of the trial piece. Generally, several modifications and trial pieces are needed. This is both a labor and capital intensive process. The second common approach is to visually check a drawing of a tool path. The judgment of the acceptability of a tool path is largely dependent on the skill of the part programmer. A visual inspection of the tool path is generally followed with trial cuts.

Recognizing the need to automate the process of checking NC code, a large body of research has been devoted to this task. Jerard et al. [21] defines the terms simulation, verification, and correction to describe the important aspects of software designed to validate NC code. To this list we would add gouge detection. Simulation is used to model the geometric aspects of metal removal. Verification is used to determine if the simulated surface meets the requirements of the design. Gouge detection is the process used to determine if a tool position has penetrated the design surface. The techniques used for simulation and validation are often used for this purpose. If an error is found in a tool path, a correction strategy is invoked to modify the offending tool position.

Simulation and Verification

Solid modeling is the classic approach used for simulation and verification. Voelcker and Hunt [54] did an exploratory study on the feasibility of using constructive solid modeling for the simulation of NC programs, and Fridshal et al. [14] modified the solid modeling package, TIPS, to perform NC simulations. The general procedure used in most solid modeling-based NC simulators can be summarized in three steps. First, the user constructs solid models representing the workpiece and the swept volume of the tool as it moves between tool positions. The swept volume of the tool is then subtracted from the workpiece model using a Boolean operation. This "as-milled" part is then compared to the model of the desired part using Boolean operations for verification. The difficulties with this process lie in the mathematics of swept volumes and the number of Boolean operations required.

The mathematical description of a swept volume can be calculated by invoking the theory of envelopes [13]. When a tool undergoes simultaneous motion in 5-axis, the resulting path of the tool tip describes a highly nonlinear path in space and time, $P(x, y, z, t) = 0$. Coordinates x, y, z are used to describe the shape of the tool at any instance and t describes the position of the tool on the tool path. The envelope or swept volume can then be calculated by determining the surface that is tangent to $P(x, y, z, t) = 0$ at every instant in time. Due to the complexity of the result, these equations are impractical to solve for all but the simplest cases. The problem is further compounded by the number of volumes needed to simulate a tool path that may contain tens of thousands of individual tool positions.

For these reasons, researchers have turned to more approximate techniques of simulating and verifying tool paths. View-based methods have been proposed by Wang and Wang [55, 56], Van Hook [51], and Atheron et al. [3]. These methods employ a variation of the z-buffer method used in computer graphics, as illustrated in Fig. 3.26. In these methods, a vector normal to the computer graphics screen is drawn

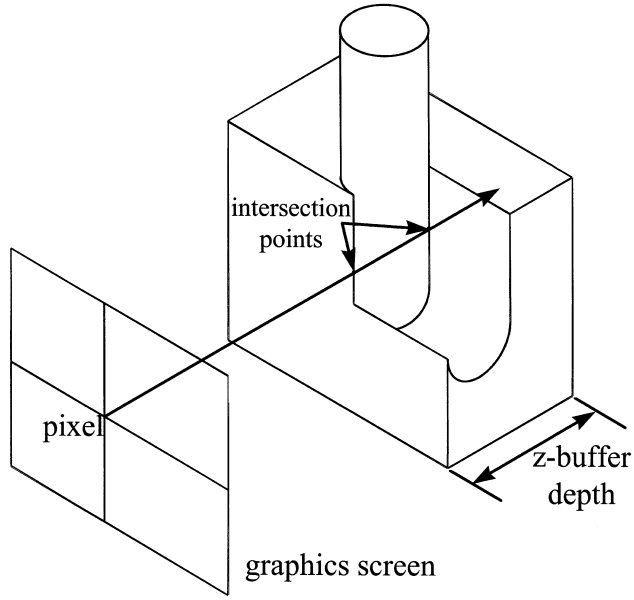


FIGURE 3.26 View space-based simulations.

through each pixel. The intersection points of these vectors with the workpiece model are stored in a z-buffer. Metal removal is simulated by performing Boolean operations between the tool and the z-buffer.

View-based simulations are very fast and interactive. The user can interact with the simulation by panning, zooming, and rotating the image. However, it is not always easy to detect and quantify errors. Errors not visible in the graphic screen cannot be detected. Generating a new view requires rerunning the entire simulation. Furthermore, the user must rely on his/her eye to detect the error and determine its magnitude. Kim et al. [25] suggest a modified z-buffer method they call a z-map for performing simulation and verification that eliminates this problem. In the z-map techniques, the simulation and display of the results are separated. The workpiece is modeled as a set of discrete columns in Cartesian space as shown in Fig. 3.27. Each z-column is stored as a number in an array called the z-map. Simulation is performed by intersecting lines defined by the z-columns with the tool motions. After each intersection, the value in the z-map is compared with the intersection result. If the intersection result is smaller than the z-map value, the value in the z-map is replaced by the intersection value. When the simulation is completed, the z-map can be displayed to visually inspect the results and then compared with the design surface to determine the magnitude of surface deviations.

The drawback with this type of simulation and any other z-buffer-based simulation is that the resolution of the simulation depends on the size of the z-buffer. For example, a 16-bit z-buffer can hold up to 65,536 levels. To overcome this problem, a number of methods based on the mow-the-grass concept have been proposed and described by Drysdale et al. [11] and Jerard et al. [19–21]. In this method, shown in Fig. 3.28, vectors extend (grow) from the desired surface at regular intervals. During the simulation, these vectors are intersected with the tool motions and the length of the vector is reduced to the intersection point. An analogy can be made to mowing a field of grass. As the simulation continues, the blades of grass represented by the vectors are "mowed down." On completion, the amount of material left above the surface or gouged out of the surface can be computed from the final length of the grass. This information can be displayed by color mapping the grass height onto the design surface.

Although the approximate methods of simulation and verification are computationally less demanding than solid modeling approaches, they still require considerable computer resources. They involve intersecting lines with objects in order to model metal removal. To have a reasonable representation of the final machined surface, a large number of vectors (blades of grass) or positions in a z-buffer are required, which

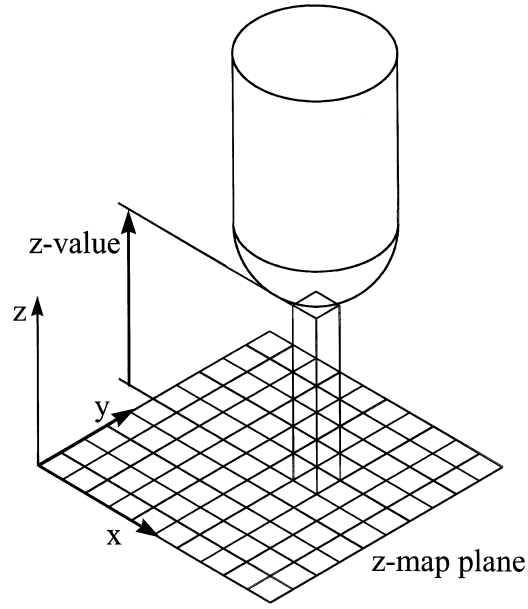


FIGURE 3.27 The Z-map technique.

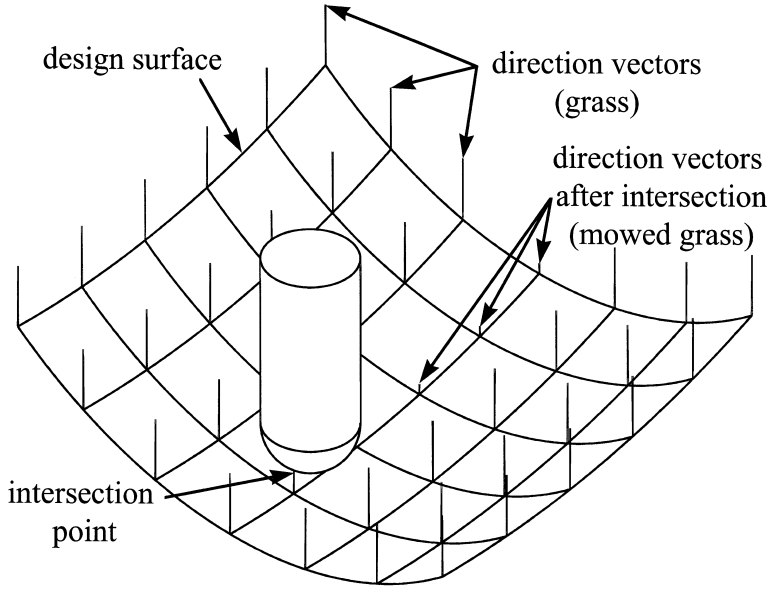


FIGURE 3.28 The mow-the-grass concept.

can quickly lead to large memory requirements. The number of these vectors depends on the size of the workpiece being modeled and the size of the expected surface deviations. In our experience, if scallop height and gouging are the phenomena of interest, a spacing between vectors of at least 10 times smaller than the tool pass interval will be required for a good representation of the machined surface. For example, a workpiece 0.5 m by 0.5 m milled with a 25.4-mm ball-nosed end mill with a tool pass interval of 0.5 mm will require a vector every 0.05 mm, for a total of 1 million evenly spaced vectors. In addition to the memory requirements of the model, considerable time must be spent on performing the intersections.

The number of intersections depends on the number of tool positions and the number of vectors underneath the tool at each position. Given that that tool positions are typically spaced at least every 1.0 mm along a tool path, the example workpiece will require 500,000 tool positions. Each tool position will have approximately 2000 vectors in its shadow. The resulting simulation will require 10^9 intersection calculations.

Gouge Detection and Correction

Often, NC programmers do not require or have the time for a full simulation and verification cycle. They are not concerned with producing a model of the machined surface. Instead, they concentrate on checking to see that each tool position is in tolerance of the surface. The result of their simulation will not tell the users the expected size of the scallops but will ensure that the surface is not gouged, undercut, or within tolerance. Examples of research in this area include the works of Takeuchi and Idemura [50], Li and Jerard [30], Rao et al. [44], Kruth and Klewais [28], and McLellan et al. [37]. These authors have all adopted different strategies for 5-axis gouge detection and correction.

Takeuchi and Idemura [50] use a boundary representation (B-rep) for the simulation of the tool motion over the design surface. A set of checkpoints is defined on the tool. At every tool position, an inside/outside test is performed between the checkpoints and the model of the workpiece to detect if gouging has occurred. Automatic correction is accomplished by rotating the tool about the cutter contact point. At the gouging position, the checkpoints are used to determine the direction and magnitude of the rotation. If this process fails, the user is prompted to manually correct the offending tool position. Li and Jerard [30] use a similar approach to detect gouging when generating tool paths for 5-axis machining with a flat-bottom end mill.

Rao et al. [44] use a variation of the mow-the-grass technique in the immediate vicinity of a cutter contact point to perform gouge detection when using the principle axis method. This tool positioning strategy relies on the minimum radius of curvature of the surface to calculate a tool inclination angle. A smaller radius of curvature produces a larger tool inclination angle. Theoretically, gouging should not occur when using this tool positioning strategy, provided that the curvature of the design surface does not change in the region underneath the tool. A gouge implies that the curvature underneath the tool has increased. Therefore, the algorithm developed by Rao et al. [44] artificially increases the curvature used in the tool positioning calculations incrementally until gouging is eliminated.

McLellan et al. [37] have developed a geometric algorithm for determining how close the tool is to a surface or how far the tool has penetrated into the design surface. The algorithm is loosely based on the theory of intersections discussed by Markot and Magedson [35]. Surfaces in close proximity contain so-called characteristic points. These points always occur in pairs, one point on each surface. The surface normals at these points are collinear. In a local sense, the closest points between two non-intersecting surfaces are the characteristic points. Similarly, in a region were two surfaces intersect, the characteristic points are the points where maximum penetration occurs. It should be noted that surfaces can have many different pairs of characteristic points. The algorithm developed by McLellan et al. [37] basically searches the tool's cutting surface and the design surface for characteristic points. If a characteristic point results in gouging, the location is stored for correction purposes. When a gouge is detected, the user has the option of using three different gouge avoidance strategies. The preferred method is to incline the tool. Based on the position and depth of the gouge, an optimal tilting direction and angle are calculated. If this strategy fails, the tool may also be lifted in the direction of the tool axis or the normal to the surface. Although lifting the tool is the surest way to eliminate gouging, McLellan et al. [37] note that unwanted material will be left on the surface in the region of the cutter contact point.

The difficulties with this gouge detection algorithm based on characteristic points arise when there is more than one characteristic point. Such a situation arises when the tool is touching a surface tangentially at a cutter contact point and gouging the surface at another point. The algorithm may not converge to the point of interest (gouge point). Instead, the algorithm may converge on the cutter contact point and the presence of gouging will not be detected. McLellan et al. [37] have solved this potential problem by restarting the algorithm at different points on the tool. If the algorithm converges to the cutter contact point from every start point, there is assumed to be no gouging.

3.4 Conclusion

5-axis machining has been demonstrated to be an efficient way of producing sculptured surfaces. It reduces the machining time and improves the surface finish. Tool path planning focuses on determining the tool pass interval and spacing between tool positions such that predictable scallops are evenly distributed across the machined surface. The main challenge of this research is to develop methods of predicting scallop height based on tool pass interval. For the most part, this issue has been resolved for ball-nosed end mills, but is still a problem when using a flat or toroidal end mill. Research efforts are underway to develop new strategies for tool positioning that match the tool profile closer to the design surface and thus reduce the need for further finishing operations. Tool positioning strategies that use curvature information, such as curvature matching, the principle axis method, and multi-point machining method in particular, were found, for the limited tests conducted, to be superior to currently used strategies. A final verdict, however, will require extensive testing on other surfaces. Research efforts are also underway to develop faster techniques for tool path simulation and verification. The ultimate objective of these techniques is to help detect and avoid gouging and interference—a major concern that casts skepticism on 5-axis machining. The current techniques require considerable investment in computer hardware to be of practical use in industry. Even with today's fast computers, a simulation and verification cycle can take several hours. More research is needed to develop and implement techniques for simulation and automatic tool path validation and correction.

Industrial acceptance of 5-axis machining will materialize only if the above research efforts lead to satisfactory solutions that could be amalgamated into existing CAM packages. In addition to software, acceptability will also depend on building more rigid machines with controllers capable of more sophisticated methods of interpolating between points on a tool path.

Acknowledgment

The authors express their appreciation for the financial support of the Natural Sciences and Engineering Research Council of Canada.

References

1. Altan, T., Lilly, B. W., Kuth J. P., Konig, W., Tonshoff, H. K., van Luttervet, C. A., Delft, T. U., and Khairy, A. B., Advanced techniques for die and mold manufacturing, *Annals of the CIRP,* 1993, 42(2), 707–716.
2. Anonymous, Technology trends, *American Machinist,* 1994, 138, 16.
3. Atherton, P. R., Earl, C., and Fred, C., A graphical simulation system for dynamic five-axis NC verification, *Proc. Autofact, SME,* Dearborn, MI, 1987, 2–12.
4. Bedi, S., Ismail, F., Mahjoob, M. J., and Chen Y., Toroidal versus ball nose and flat bottom end mills, *International Journal of Advanced Manufacturing Technology,* accepted Aug. 1996.
5. Bobrow, J. E., NC machine tool path generation from CSG part representations, *Computer-Aided Design,* 1985, 17(2), 69–76.
6. Cho, H. D., Jun, Y. T., and Yang, M.Y., Five-axis CNC milling for effective machining of sculptured surfaces, *International Journal of Production Research,* 1993, 31(11), 2559–2573.
7. Choi, B., Lee, C., Huang, J., and Jun, C., Compound surface modeling and machining, *Computer-Aided Design,* 1988, 20(3), 127–136.
8. Choi, B. K. and Jun, C. S., Ball-end cutter interference avoidance in NC machining of sculptured surface, *Computer-Aided Design,* 1989, 21(6), 371–378.
9. Choi, B. K., Park, J. W., and June, C. S., Cutter-location data optimization in 5-axis machining, *Computer-Aided Design,* 1993, 25(6), 377–386.
10. Chou, J. J. and Yang, D. C. H., On the generation of coordinated motion of five axis CNC/CMM machines, *Journal of Engineering for Industry,* 1992, 114, 15–22.

11. Drysdale, R. L., Jerard, R. B., Schaudt, B., and Hauck, K., Discrete simulation of NC machining, *Agorithmica,* 1889, 4(1), 33–60.
12. Farin, G., *Curves and Surfaces for Computer Aided Geometric Design,* 4th ed., Academic Press, New York, 1997.
13. Faux, I. D. and Pratt, M. J., *Computational Geometry for Design and Manufacture,* Ellis Horwood Limited, Chichester, 1979.
14. Fridshal, R., Cheng, K. P., Duncan, D., and Zucker, W., Numerical control part program verification system, *Proc. Conf. CAD/CAM Technology in Mechanical Engineering,* MIT, Cambridge, MA, 1982.
15. Groover, M. P., *Automation, Production Systems, and Computer-Integrated Manufacturing,* Prentice-Hall, Englewood Cliffs, NJ, 1987.
16. Huang, Y. and Oliver, J. H., NC milling error assessment and tool path correction, *Computer Graphics,* SIGGRAPH 94, Orlando, FL, 24–29 July, 1994, 287–294.
17. Huang, Y. and Oliver, J. H., Non-constant parameter NC tool path generation on sculptured surfaces, *International Journal of Advanced Manufacturing Technology,* 1994, 9, 281–290.
18. Hui, K. C. Solid sweeping in image space—application in NC simulation, *The Visual Computer,* 1994, 10(6), 306–316.
19. Jerard, R. B., Drysdale, R. L., and Hauck, K., Geometric simulation of numerical controlled machining, *Proc. ASME Int. Computers in Engineering Conf.*, San Francisco, 1988, 2, 129–136.
20. Jerard, R., Drysdale, R., Hauck, K., Schaudt, B., and Magewick, J., Methods for detecting errors in numerically controlled machining of sculptured surface, *IEEE Computer Graphics and Applications,* 1989, 9(1), 26–39.
21. Jerard, R. B., Hussaini, S. Z., Drysdale R. L., and Schaudt, B., Approximate methods for simulation and verification of numerically controlled machining programs, *The Visual Computer,* 1989, 5, 329–348.
22. Jensen, C. G. and Anderson, D. C., Accurate tool placement and orientation for finished surface machining, *Journal of Design and Manufacture,* 1993, 3(4), 251–261.
23. Jensen, C. G., Anderson D. C., and Mullins S. H., Scallop elimination based on precise 5-axis tool placement, orientation, and step-over calculations, *Advances in Design Automation ASME,* 1993, 65(2), 535–544.
24. Kawabe, S., Kimura, F., and Sata, T., Generation of NC commands for sculptured surface machining from 3-coordinate measuring data, *CIRP Ann.,* 1981, 30(1), 369–372.
25. Kim, C. B., Park, S., and Yang, M. Y., Verification of NC tool path and manual and automatic editing of NC code, *International Journal of Production Research,* 1995, 33(3), 659–673.
26. Kiridena, V. and Ferreira, P. M., Mapping the effects of positioning errors on the volumetric accuracy of five-axis machine tools, *International Journal of Machine Tools and Manufacture,* 1993, 33(3), 417–436.
27. Krieziz, G. A., Algorithms for rational spline surface intersections, Ph.D thesis, Department of Ocean Engineering, MIT, 1990.
28. Kruth, J. P. and Klewais P., Optimization and dynamic adaptation of cutter inclination during five-axis milling of sculptured surfaces, *Annals of the CIRP,* 1994, 43(1), 443–448.
29. Lee, Y. S. and Chang, T. C., Machine surface error analysis for 5-axis machining, *International Journal of Production Research,* 1996, 34(1), 111–135.
30. Li, S. X. and Jerard, R. B., 5-axis machining of sculptured surfaces with a flat-end cutter, *Computer-Aided Design,* 1994, 26(3), 165–178.
31. Li, F., Wang, X. C., Ghosh, S. K., and Kong, D. Z., Gouge detection and tool position modification for five-axis NC machining of sculptured surfaces, *Journal of Materials Processing Technology,* 1995, 48, 739–745.
32. Li, F., Wang, X. C., Ghosh, S. K., and Kong, D. Z., Tool-path generation for machining sculptured surfaces, *Journal of Materials Processing Technology,* 1995, 48, 811–816.

33. Lin, R. S. and Koren, Y., Efficient tool-path planning for machining free-form surfaces, *Journal of Engineering for Industry,* 1996, 118(1), 20–28.
34. Loney, G. and Ozsoy, T., NC machining of free form surfaces, *Computer-Aided Design,* 1987, 19(2), 85–90.
35. Markot, R. P. and Magedson, R. L., Solutions of tangential surface and curve intersections, *Computer-Aided Design,* 1989, 21(7), 421–429.
36. Martin, R. R., The geometry of the helical canal surface, *The Mathematics of Surface IV,* 1994, 17–32.
37. McLellan, E., Young, G. M., Goult, R. J., and Pratt, M. J., Interference checking in the 5-axis machining of parametric surfaces, *Computer-Aided Surface Geometry and Design, The Mathematics of Surfaces IV,* Clarendon Press, Oxford, 1994, 439–453.
38. Menon, J. P. and Voelker, H. B., Towards a comprehensive formulation of NC verification and as mathematical and computational problem, *Concurrent Engineering,* 1992, PED59. 147–164.
39. Oliver, J. H., Graphical verification of numerically controlled milling programs for sculptured surfaces, Ph.D. thesis, Michigan State University, U.S.A., 1986.
40. Paul, R. P., *Robot Manipulators: Mathematics, Programming and Control,* MIT Press, Cambridge, MA, 1981.
41. Pegna, J. and Wolter, F.-E., Designing and mapping trimming curves on surfaces using orthogonal projection, *ASME Advances in Design Automation, Vol. 1, Computer Aided and Computational Design,* 1990, DE-Vol. 23(1), 235–245.
42. Press, W. H., Teukolsky S. A., Vetterling W. T., and Flannery B. P., *Numerical Recipes in C. The Art of Scientific Computing,* 2nd ed., Cambridge University Press, 1992.
43. Rao, N., Bedi S., and Buchal, R., Implementation of the principal-axis method for machining of complex surfaces, *International Journal of Advanced Manufacturing Technology,* 1996, 11, 249–257.
44. Rao, N., Ismail, F., and Bedi, I., Tool path planning for five-axis machining using the principle axis method, *International Journal of Machine Tool and Manufacture,* accepted 1997.
45. Ruegg, A., A generalized kinematics model for three- to five-axis milling machines and their implementation in CNC, *Annals of the CIRP,* 1992, 41(1), 547–550.
46. Sakamoto, S. and Inasaki, I., Analysis of generating motion for five-axis machining centers, *Transactions of NAMRI/SME,* 1993, 21, 287–293.
47. Sata, T., Kimura, F., Okada, N., and Hosaka, M., A new method of NC interpolator for machining the sculptured surface, *CIRP Ann.,* 1981, 30(1), 369–372.
48. Suh, Y. S. and Lee, K., NC milling tool path generation for arbitrary pockets defined by sculptured surfaces, *Computer-Aided Design,* 1990, 22(5), 371–378.
49. Suresh K. and Yang. D., Constant scallop-height machining of free-form surfaces, *Journal of Engineering for Industry,* 1994, 116, 273–283.
50. Takeuchi, Y. and Idemura, T., 5-axis control machining and grinding based on solid model, *Annals of the CIRP,* 1991, 40(1), 455–458.
51. Van Hook, T., Real-time shaded NC milling display, *Computer Graphics, Proc. SIGGRAPH,* 20(4), 15–20.
52. Vickers, G. W. and Bedi, S., The definition and manufacture of compound curvature surfaces using G-Surf, *Computers in Industry,* 1985, 6, 173–183.
53. Vickers, G. W. and Quan K. W., Ball-mills versus end-mills for curved surface machining, *Journal of Engineering for Industry,* 1989, 111(22), 22–26.
54. Voelcker, H. B. and Hunt, W. A., The role of solid modeling in machine process modeling and NC verification, *Proc. 1981 SAE International Congress and Exposition,* Detroit, MI, 1981.
55. Wang, W. P. and Wang, K. K., Real time verification of multiaxis NC programs with raster graphics, *IEEE Proc. of 1986 International Conference on Robotics and Automation,* San Francisco, April 1986, 166–171.

56. Wang, W. P. and Wang, K. K., Geometric modeling for swept volume of moving solids, *IEEE Computer Graphics Applications,* 1986, 6(12), 8–17.
57. Wang, X. C., Ghosh, S. K., Li, Y. B., and Wu, X. T., Curvature catering—a new approach in manufacturing of sculptured surfaces (Part 1. Theorem), *Journal of Materials Processing Technology,* 1993, 38, 159–176.
58. Wang, X. C., Ghosh, S. K., Li, Y. B., and Wu, X. T., Curvature catering—a new approach in manufacturing of sculptured surfaces (Part 2. Methodology), *Journal of Materials Processing Technology,* 1993, 38, 177–194.
59. Warkentin, A., Bedi, S., and Ismail, F., 5-Axis milling of spherical surfaces, *International Journal of Machine Tools and Manufacture,* 1995, 36(2), 229–243.
60. Warkentin, A., Ismail, F., and Bedi, S., Intersection approach to multi-point machining of sculptured surfaces, *Computer Aided Geometric Design,* submitted 1996.
61. Wysocki, D. A., Oliver, J. H., and Goodman, E. D., Gouge detection algorithms for sculptured surface NC generation, *ASME Computer-Aided Design and Manufacture of Cutting and Forming Tools,* 1989, PED 40, 39–44.

4

Finite Element Method and Thermo-Viscoplastic Cutting Model in Manufacturing Systems

4.1 Introduction

4.2 Review of Cutting Model
Shear Plane Model • Slip Line Field Method • FEM Applied to Cutting • Basic Problems in Analyzing the Machining Process

4.3 Thermo-Viscoplastic Cutting Model
Material Behavior at High Strain Rate and Temperature • Governing Equations • Finite Element Discretization • Special Numerical Features • Boundary Condition Problems • Effective Strain Calculation • Computational Procedures

4.4 Application
Material Properties and Friction Forces Determination • Numerical Simulation Example

4.5 Conclusions

Kug Weon Kim
Chonan National Technical College

Hyo-Chol Sin
Seoul National University

4.1 Introduction

Today, the analysis of machining processes is more important than ever before. The increasing use of numerically controlled machine tools, along with increasingly sophisticated computer control methods, makes the selection of cutting conditions, which ensures effective machining, much more pressing. Experimentally obtained machining data is another important ingredient, but this can be extremely costly in terms of both time and material.

Predicting forces and temperatures is one of the most common objectives for modeling machining processes. Forces exerted by the tool on the workpiece can cause deflections that lead to geometric errors or difficulty in meeting tolerance specifications. Reaction forces of the workpiece on the tool can, if large enough, cause failure of the tool. The product of the force and velocity vectors is used to predict power requirements for sizing a new machine tool, or for estimating the production rates. Temperature predictions are used to estimate a tool's useful life or a change in mechanical properties of the workpiece. Both of these considerations are important for economical operation of the process, as well as safety and performance of the machined product.

Since 1906 when Taylor's paper was first published, many researchers have investigated the basic mechanics of the machining process. Despite their efforts, the basic mechanics aspects of the machining process are still not clearly understood, primarily due to the difficulty in modeling the contact, and

work-material deformation with large plastic strain and friction, high temperature and strain rates, and their coupling effects.

In the past decade, there has been a considerable amount of research applying finite element modeling to predict chip flow, stresses, and temperatures in metal machining. A thermo-viscoplastic cutting model (Kim and Sin, 1996) is one among various possible cutting models, and only a first step toward developing an ideal cutting model that enables one to predict machining performance without resorting to cutting experiments. With the goal of developing an efficient and accurate cutting model, and promoting the finite element method for machining process modeling, this chapter introduces the basic concepts of a thermo-viscoplastic cutting model.

4.2 Review of Cutting Model

Shear Plane Model

The shear plane model, which was the first proposed cutting model, assumes that the plastic deformation appears only in a shear plane, and the remainder undergoes rigid body motion (Merchant, 1944). The goal of this method is to predict the shear angle, the chip thickness, and the force generated. Minimizing the cutting energy with respect to the shear angle yields the direction of the shear plane. Various research has been carried out, especially for predicting the shear angle, and several useful equations have been proposed (Shaw et al., 1953; Nakayama and Arai, 1976; Wright, 1982).

Slip Line Field Method

The slip line field solution is more inclusive in the sense that the deformation zone is not a simple shear plane but a finite region. Using plasticity theory for the plane strain case, slip line fields are constructed around the primary shear zone from experiments and the shear plane can be found to be in the direction of maximum shear stress. Some headway has been made in considering cutting temperature with the flow stress of the workpiece allowed to vary with strain and strain rate. These analyses have, however, largely been dependent on the experimental observations. Moreover, major attention has focused on orthogonal machining, in which a single straight-edged cutting tool removes a chip under plane strain conditions. Among the studies in this area are the works of Kececioglu (1958), Morcos (1980), and Oxley (1989).

FEM Applied to Cutting

With the advent of digital computers, the finite element method (FEM) became a powerful tool that could overcome the computational difficulties associated with modeling the machining process. One of the first analyses on metal cutting using the FEM was presented by Klamecki (1973), in which he simulated the incipient chip formation process of a three-dimensional cutting process with the assumption of an elastic-plastic bi-linear strain hardening workpiece. Lajczok (1980) developed a simplified model for orthogonal cutting without the chip region to calculate the residual stresses in the workpiece. Tool forces were measured experimentally and utilized in the model, with the stress distributions on the chip-tool interface based on the Zorev model. Usui and Shirakashi (1982) used the FEM to simulate steady-state cutting, while the shear angle and chip geometry were assumed in advance. They showed that the prediction of orthogonal cutting process without any actual cutting experiments was possible if the flow properties and the friction characteristics were available. The analysis, however, was limited to a rate-independent deformation behavior. Iwata, Osakada, and Terasaka (1984) developed a rigid-plastic finite element model for orthogonal cutting in a steady-state condition with very low cutting speeds, wherein fracture of the chip was predicted by employing a criterion based on stress history.

Recently, there has been increased research on analyzing the cutting process using the FEM. Most cutting models using the FEM can be classified into two categories: the chip formation cutting model,

and the steady-state cutting model. More recently, as demand for ultraprecision machining increased, studies on microcutting using molecular dynamics simulation has also been conducted (Ikawa et al., 1991; Shimada et al., 1993). In these works, however, only a feasibility study is provided on nanometric or the ultimate accuracy attainable in the mechanical material removal process.

Chip Formation Cutting Model

Principal advantages of the chip formation cutting model are that the tool can be simulated from incipient to steady-state cutting and that the chip geometry and residual stresses in the workpiece can be predicted. One of the disadvantages, however, is that the model requires large computational time to reach steady-state conditions. In addition, a material failure and parting mechanism must be provided to allow the chip to separate from the workpiece. This necessitates an additional material failure criterion for chip separation.

The first model for orthogonal cutting utilizing simulated chip formation from the incipient stage to the steady state was due to Strenkowski and Carroll (1985). In their study, no heat was assumed to be conducted between chip and workpiece. The values of the chip separation criterion based on effective plastic strain were used to simulate the cutting process, without comparison to experiments. When it exceeded a specified value at the nodes closest to the tool tip, the nodes would be separated to form the machined surface and chip underneath. They found that different values selected for chip separation criterion based on the effective plastic strain affect the magnitude of the residual stresses in the machined surface of the workpiece. Strenkowski and Mitchum (1987) presented the modified cutting model. They analyzed the transition from indentation to incipient cutting and used their results to evaluate the values for the chip separation criterion. Their results showed that the criterion value, based on the effective plastic strain, increases with depth of cut.

Lin and Lin (1992) developed a thermo-elastic-plastic cutting model. The finite difference method was adopted to determine the temperature distribution within the chip and the tool. In their model, a chip separation criterion based on the strain energy density was introduced. They verified that the chip separation criterion value based on the strain energy density was a material constant and was independent of uncut chip thickness. With this cutting model, Lin and Pan (1993) simulated an entire orthogonal cutting process with tool frank wear from incipient stage to the steady state. The agreement between simulation and experimental results for cutting forces, friction force, the chip thickness, and contact length was found to be acceptable. Lin and Liu (1996) analyzed an orthogonal finish machining using tungsten carbide and diamond tools that had different coefficients of thermal conductivity. Comparing the cutting forces predicted by the model with those obtained by experiment for orthogonal finish machining, they showed that the model could simulate the orthogonal finish machining process for different tool materials.

Ueda and Manabe (1993) simulated the oblique cutting process as the first step in the three-dimensional deformation analysis of cutting. With a displacement-based criterion, the chip formation process is continuously simulated from the beginning of the cutting until the steady state. When the distance between the tool tip and the nodal point located immediately in front exceeded a predefined critical value of separation, the nearest element was separated. However, the physics of a displacement-based chip separation criterion has not been clearly characterized. The results of the rigid plastic finite element analysis are qualitatively compared with *in situ* scanning electron microscope observation of the cutting experiments.

Zhang and Bagchi (1994) developed conditional link elements handling chip separation from the workpiece. A displacement-based chip separation criterion was used to selectively null the link elements as the tool progressed. The validity of this model was examined by comparing the calculated cutting force, feed force, shear angle, and plastic deformation with those from experiments. But because this model ignored the temperature and strain rate effects, it was only useful for low-speed machining.

Hashemi et al. (1994) presented the first cutting model that did not assume a predefined cutting path. A unique chip fracture algorithm was implemented to permit tool travel between any set of nodes and to allow segmentation of the chip. Although a number of important features such as friction,

temperature effects, and workpiece strain hardening were neglected and an overly coarse mesh that restricted the number of potential tool pathways was used, this model was thought to be a considerable achievement.

Steady-State Cutting Model

Developing a steady-state cutting model may be a more effective approach, from the viewpoint of computational efficiency. This approach, primarily using an Eulerian coordinate frame, has been successfully applied to other manufacturing processes, such as metal forming. Although this cutting model requires that the boundaries of the chip free surface be known in advance or adjusted interactively during the simulation, it may be used to study steady-state cutting without the need to simulate the lengthy transition from incipient to steady-state cutting conditions, and without the need for a chip separation criterion. Other advantages are that fewer elements are necessary to discretize the chip and workpiece, and that the location of the chip separation line need not be known in advance.

The first application of the approach to metal cutting using a viscoplastic material model was reported by Strenkowski and Carroll (1986). This model is the so-called Eulerian cutting model. As this approach considers each element to be a fixed control volume, such that the mesh does not move with the flowing material as with the Lagrangian approach, the boundaries of the chip must be known in advance. These researchers later (1988) investigated the correlation between the Lagrangian and Eulerian frames. Both approaches showed reasonably good correlation with experimental results for cutting forces, residual stresses and strains, and chip geometry. With the Eulerian approach, Strenkowski and Moon (1990) analyzed steady-state orthogonal cutting with the capability to predict chip geometry and chip-tool contact length for purely viscoplastic materials.

In 1991, Komvopoulos and Erpenbeck investigated steady-state cutting effects of cratered tools with built-up edges. They examined the effects of material constitutive behavior for rate-independent elastic-perfectly plastic materials and rate-sensitive elastic-plastic isotropically strain hardening material. The analysis assumed an initially stress-free steady-state chip geometry and modeled a relatively short progression of the tool. A displacement-based chip separation criterion was used, and good correlation with experimental results was achieved.

Moriwaki et al. (1993) developed a rigid plastic finite-element model to simulate the orthogonal micro-cutting process and examined the effects of the tool edge radius to depth of cut in the micro-cutting process. They also analyzed the flow of cutting heat and temperature distribution. In analyzing temperature, however, they did not consider the variation of the flow stress with temperatures and the velocities in workpiece and chip and, hence, their study was for very low cutting speeds.

Shih and Yang (1993) developed a plane strain steady-state cutting model using a displacement-based chip separation criterion. Consideration of strain rate, temperature, and friction effects was combined with a clever remeshing scheme to allow efficient analysis of very long cutting lengths. To improve the accuracy of residual stresses, workpiece cooling was considered. Recently, a more realistic stick-slip friction model was incorporated, together with an unbalanced force reduction technique that stabilized the chip separation process (Shih, 1995). Contour plots and an Eulerian description of the material deformation were also presented to gain better understanding of the metal cutting mechanics (Shih 1996).

Joshi et al. (1994) calculated the strain rates and stresses in the primary shear deformation zone and compared them with the known experimental results for orthogonal steady-state machining. The material behavior in the metal cutting process was modeled by a viscoplastic constitutive equation.

Wu et al. (1996) developed a thermo-viscoplastic model of the orthogonal machining process based on a three-field, mixed, finite element method. This method was highly accurate for coarse meshes, computationally very cheap, and did not suffer from locking for incompressible materials. It also satisfied the nontrivial stress boundary condition better than the compatible displacement model. From this model, detailed information about the stress, strain rate, temperature, cutting forces, chip thickness, shear angle, contact length, chip geometry, and heat conduction could be obtained. Simulations were performed for aluminum 6061 alloy and pure titanium.

Kim and Sin (1996) developed a cutting model by applying the thermo-viscoplastic FEM to analyze the mechanics of the steady-state orthogonal cutting process. The model was capable of dealing with free chip geometry and chip-tool contact length. The coupling with thermal effects was also considered. In analyzing temperature distributions, the "upwind" scheme was employed and hence it was possible to analyze high-speed metal cutting. For 0.2% carbon steel, good correlation between experiments and simulations was found.

Basic Problems in Analyzing the Machining Process

Flow Stress of Work Material

The problem of modeling material behavior in the cutting process is very important, and previous studies can be divided into the following categories: rigid-plastic (Iwata et al., 1984; Moriwaki et al., 1993; Ueda and Manabe, 1993), elastic-plastic (Klamecki, 1973; Lajczok, 1980; Usui and Shirakashi, 1982; Strenkowski and Carrol, 1985; Komvopoulos and Erpenbeck, 1991; Zhang and Bagchi, 1994), and viscoplastic (Strenkowski and Carrol, 1986; Strenkowski and Moon, 1990; Joshi et al., 1994). Temperature effects have also been considered in some models, which include thermo-elastic-plastic (Lin and Lin, 1992; Lin and Pan, 1993; Lin and Liu, 1996), thermo-elastic-viscoplastic (Shih and Yang, 1993; Shih, 1995; 1996), and thermo-viscoplastic (Wu et al., 1996; Kim and Sin, 1996) materials.

In conventional machining, deformation of the workpiece takes place at high temperatures, and in this case material properties can vary considerably with temperatures. Elevated temperatures can cause the change of flow stress of the workpiece material, and have a dominating influence on the rate of wear of the tool and on the generation of a thermo-deformation layer in the machined surface. Thus, the consideration of temperature effects in the analysis of metal cutting is very important. Because materials at elevated temperatures are usually rate sensitive, a complete analysis of the machining process requires two considerations: the effect of the rate sensitivity of materials, and the coupling of the metal flow and heat transfer analyses. One of the methods of obtaining the flow stress in machining is by using machining test directly (Stevenson and Oxley, 1970; 1971). Alternatively, high-speed compression tests (Oyane et al., 1967) or high-speed torsion tests (Stevenson, 1975) can be used.

Up to now, attention has mainly been limited to plain carbon steel work material. It will be necessary to extend this work, obtaining high strain, high strain rate, high temperature flow stress properties, to other materials for analyzing the machining process. The lack of material data limits the application of finite elements in the quantitative analysis of machining.

Friction Force in Chip-Tool Interface

As the friction force is strongly related to the chip formation process and tool performance, it is essential to consider the chip-tool contact phenomenon in developing an accurate cutting model. Stress distributions (Usui and Takeyama, 1960; Chandrasekar and Kapoor, 1965; Kato et al., 1972; Doyle et al., 1979; and Bagchi and Wright, 1987) from the photoelastic technique, split tool techniques, or transparent sapphire tool technique show a nonuniform stress distribution on the rake face. A typical stress distribution on the rake face is shown in Fig. 4.1. The normal stress increases monotonically toward the tool edge, but the shear stress increases first and then reaches a nearly constant value. That is to say, there are two distinct regions on the rake face: sliding and sticking. In the sliding region, the normal stress is relatively small and dry friction is preserved. In the sticking region, the normal stress is so high that the real and apparent areas of contact are equal and the frictional stress is approximately constant. Based on experimental research

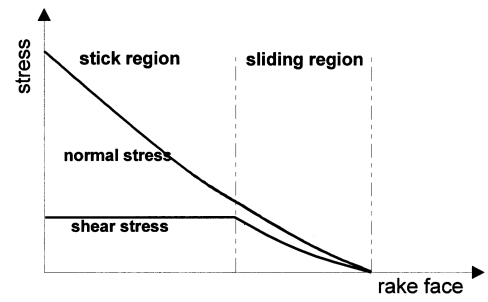


FIGURE 4.1 A typical stress distribution on the rake face.

results, a constant coefficient of friction (i.e., Coulomb's friction law) is used in the sliding region, and a constant frictional stress in the sticking region as an approximation. This can be represented as:

$$\tau_f = \mu\sigma_n \qquad \text{when} \qquad \tau_f < k \tag{4.1}$$

$$\tau_f = k \qquad \text{when} \qquad \tau_f \geq k \tag{4.2}$$

where τ_f is the frictional stress, σ_n is the normal stress, μ is the coefficient of friction, and k is the shear stress of the chip material.

The above friction model has been generally used for most cutting models applying the FEM. With this approach, constant frictional stress can be obtained from the flow stress of the chip. However, it is very difficult to determine the coefficient of friction because the frictional conditions in the sliding region are different from those of the conventional frictional test. The bottom surface of the chip is a newly formed surface with high strain hardening; the hardness of the chip can be twice as high as that of the workpiece because of the plastic deformation in the chip. This hardness variation may cause changes in the friction coefficient.

There are other ways to determine frictional stresses; for example, machining tests (Shirakashi and Usui, 1973) and specially designed tests (Ikawa et al., 1984). These methods give us nearly true information for frictional stresses on the rake face, but only few such experimental data are available for specific conditions.

4.3 Thermo-viscoplastic Cutting Model

Material Behavior at High Strain Rate and Temperature

Material behavior that exhibits rate sensitivity is called viscoplastic. According to the von Mises criterion, the deviatoric stress component is related to the strain rate component by the following equation (Malvern, 1969):

$$s_{ij} = \dot{\lambda}\dot{\varepsilon}_{ij} \tag{4.3}$$

$$\dot{\lambda} = 2(\mu + \sigma_Y / 3\dot{\bar{\varepsilon}}) \tag{4.4}$$

Here, $\dot{\lambda}$ is the proportional constant, μ is the viscosity, σ_Y is the yield stress in tension, and $\dot{\bar{\varepsilon}}$ is the second invariant of the strain rate tensor. For a perfectly plastic material, the first term in Eq. (4.4), which represents the viscosity of the material, is zero; while for a viscous liquid, the plasticity term (i.e., the second term) is zero, as here there is no critical stress for yielding. Equation (4.3) can be rewritten as follows:

$$\bar{\sigma} = \frac{3}{2}\dot{\lambda}\dot{\bar{\varepsilon}} \tag{4.5}$$

where $\bar{\sigma}$ is the second invariant of the deviatoric stress tensor. The second invariants of the strain rate tensor and the deviatoric stress tensor are defined as follows:

$$\dot{\bar{\varepsilon}} = \sqrt{\frac{2}{3}\dot{\varepsilon}_{ij}\dot{\varepsilon}_{ij}} \tag{4.6}$$

$$\bar{\sigma} = \sqrt{\frac{3}{2}s_{ij}s_{ij}} \tag{4.7}$$

For a von Mises material model, $\dot{\bar{\varepsilon}}$ and $\bar{\sigma}$ are also called the effective (flow) strain rate and the effective (flow) stress, respectively.

Governing Equations

The governing equations for deformation of viscoplastic material are formally identical to those of plastic materials, except that the effective stress is a function of strain, strain rate, and temperature.

Velocity Equation

Consider a viscoplastic deformation of material with surface S and volume V under prescribed surface traction f over the surface S_f and prescribed velocities v^* over the surface S_v. If we choose the velocity field as the main unknown, the problem to solve in the general case is the following:

Equilibrium equation:
$$\sigma_{ij,j} = 0 \tag{4.8}$$

Constitutive equation:
$$\dot{\varepsilon}_{ij} = \frac{3}{2}\frac{\dot{\bar{\varepsilon}}}{\bar{\sigma}} s_{ij} \tag{4.9}$$

Boundary conditions:
$$v_i = v_i^* \quad \text{on} \quad S_v \tag{4.10}$$
$$f_i = f_i^* \quad \text{on} \quad S_f$$

Compatibility condition:
$$\dot{\varepsilon}_{ij} = \frac{1}{2}(v_{i,j} + v_{j,i}) \tag{4.11}$$

Temperature Equation

The equation describing the heat transfer processes occurring during orthogonal cutting is the steady, two-dimensional energy equation:

$$k_c T_{,ii} - \rho C_p \frac{dT}{dt} + \dot{Q} = 0 \tag{4.12}$$

Boundary conditions:
$$\begin{aligned} T &= T_b \\ -k_c \frac{\partial T}{\partial n} &= q \\ -k_c \frac{\partial T}{\partial n} &= h(T - T_o) \end{aligned} \tag{4.13}$$

where k_c, ρ, and C_p are, respectively, the thermal conductivity, density, and specific heat; $\dot{Q}$ is the rate of heat generation per unit volume. S_T, S_q and S_h are surfaces where, respectively, the temperature, heat flux, and heat transfer coefficient are defined, and n is the outward normal to the boundary. T_b is the stationary temperature on the boundary and T_o is the ambient temperature. $\dot{Q}$ can be written as

$$\dot{Q} = \alpha \cdot \sigma_{ij} \cdot \dot{\varepsilon}_{ij} \tag{4.14}$$

where the heat generation efficiency α represents the fraction of mechanical energy transformed into heat, and is usually assumed to be 0.9.

Finite Element Discretization

Velocity Equation

With a weak form of the equilibrium and the incompressibility constraint given by $\dot{\varepsilon}_{ii} = 0$, the basic equation for the finite element discretization is given by:

$$\int_V s_{ij}\delta\dot{\varepsilon}_{ij}dV + C_K\int_V \dot{\varepsilon}_{ii}\delta\dot{\varepsilon}_{ii}dV - \int_{S_f} f_i^*\,\delta v_i dS = 0 \tag{4.15}$$

where C_K is the penalty constant and is a very large number (10^5 to 10^6).

The unknown velocity field **v** is discretized in terms of the interpolation functions:

$$\mathbf{v} = \sum_n \mathbf{v}^n N_n(\mathbf{x}) \tag{4.16}$$

where $\mathbf{v}^n$ is the velocity vector at node n with components v_i^n, and N_n is a global interpolation function that takes the value 1 at node n and 0 for any other node. The strain rate tensor can be evaluated from Eq. (4.16)

$$\dot{\varepsilon}_{ij} = \frac{1}{2}\sum_n \left(v_i^n \frac{\partial N_n(x)}{\partial x_j} + v_j^n \frac{\partial N_n(x)}{\partial x_i} \right) \tag{4.17}$$

and the **B** operator is defined from Eq. (4.17):

$$\dot{\varepsilon}_{ij} = \sum_{nk} B_{ijnk} v_k^n \tag{4.18}$$

or in matrix form as

$$\dot{\varepsilon} = \mathbf{B}\mathbf{v} \tag{4.19}$$

The effective strain rate can be represented in matrix form as follows:

$$(\dot{\bar{\varepsilon}})^2 = \dot{\varepsilon}^{\mathrm{T}}\mathbf{D}\dot{\varepsilon} = \mathbf{v}^{\mathrm{T}}\mathbf{B}^{\mathrm{T}}\mathbf{D}\mathbf{B}\mathbf{v} \tag{4.20}$$

The diagonal matrix **D** for plane strain problem is:

$$\mathbf{D} = \begin{bmatrix} \frac{2}{3} & 0 & 0 \\ 0 & \frac{2}{3} & 0 \\ 0 & 0 & \frac{1}{3} \end{bmatrix} \tag{4.21}$$

The volumetric strain rate is given by

$$\dot{\varepsilon}_v = \dot{\varepsilon}_x + \dot{\varepsilon}_y + \dot{\varepsilon}_z = \mathbf{C}^{\mathrm{T}}\mathbf{v} \tag{4.22}$$

and

$$\mathbf{C}^{\mathrm{T}} = [1\ \ 1\ \ 1\ \ 0]\mathbf{B} \tag{4.23}$$

Then, the basic equation, Eq. (4.15), is discretized and a set of nonlinear simultaneous equations is obtained:

$$\mathbf{Kv} = \mathbf{R} \tag{4.24}$$

where

$$\mathrm{K} = \int_V \frac{\bar{\sigma}}{\dot{\bar{\varepsilon}}} \mathbf{B}^{\mathrm{T}} \mathbf{D} \mathbf{B} \, dV + C_k \int_V \mathbf{C} \mathbf{C}^{\mathrm{T}} \, dV \tag{4.25}$$

$$\mathbf{R} = \int_{S_f} \mathbf{N}^{\mathrm{T}} \mathbf{F} \, dS \tag{4.26}$$

After the usual finite element assembly procedure, the set of global unknowns (velocities) must be calculated iteratively. Iteration continues until the nonlinear simutaneous equations converge, which is to require that

$$\frac{\left(\sum_i \Delta v_i^2 \right)^{1/2}}{\left(\sum_i v_i^2 \right)^{1/2}} < E \tag{4.27}$$

The convergence criterion E is problem dependent.

Temperature Equation

The temperature T at any point in a finite element can be expressed in terms of the corresponding nodal point values as:

$$T = \mathbf{NT} \tag{4.28}$$

where **N** is the matrix of shape functions and **T** is the vector of nodal temperatures. The Galerkin method with the weighting function, W_i allows us to write:

$$\int_V \mathbf{W}^{\mathrm{T}} \left[k_c \frac{\partial^2 \mathbf{N}}{\partial x_i^2} \mathrm{T} - \rho C_p v_i \frac{\partial \mathbf{N}}{\partial x_i} \mathrm{T} + \dot{Q} \right] dV = 0 \tag{4.29}$$

After application of the Green-Gauss theorem, the above equation becomes:

$$\int_V \left[k_c \frac{\partial \mathbf{W}^{\mathrm{T}}}{\partial x_i} \frac{\partial \mathbf{N}}{\partial x_i} \mathrm{T} + \rho C_p v_i \mathbf{W}^{\mathrm{T}} \frac{\partial \mathbf{N}}{\partial x_i} \mathrm{T} + \mathbf{W}^{\mathrm{T}} \dot{Q} \right] dV + \int_S \mathbf{W}^{\mathrm{T}} k_c \frac{\partial \mathbf{T}}{\partial n} = 0 \tag{4.30}$$

The last term of Eq. (4.30) is written by the boundary condition as follows:

$$\int_S \mathbf{W}^{\mathrm{T}} k_c \frac{\partial \mathbf{T}}{\partial n} = -\int_{S_q} \mathbf{W}^{\mathrm{T}} q \, dS - \int_{S_h} h \mathbf{W}^{\mathrm{T}} \mathbf{NT} \, dS + \int_{S_h} \mathbf{W}^{\mathrm{T}} h T_0 \, dS \tag{4.31}$$

Then, Eq. (4.30) can be represented by the following equation:

$$\mathbf{HT} = \mathbf{F} \tag{4.32}$$

where
$$\mathrm{H} = \int_V \left[k_c \frac{\partial \mathbf{W}^{\mathrm{T}}}{\partial x_i} \frac{\partial \mathbf{N}}{\partial x_i} + \rho C_p v_i \mathbf{W}^{\mathrm{T}} \frac{\partial \mathbf{N}}{\partial x_i} \right] dV + \int_{S_h} h \mathbf{W}^{\mathrm{T}} \mathbf{N} \; dS \tag{4.33}$$

$$\mathbf{F} = \int_V \mathbf{W}^{\mathrm{T}} \dot{Q} \; dV + \int_{S_q} \mathbf{W}^{\mathrm{T}} q \; dS - \int_{S_h} \mathbf{W}^{\mathrm{T}} h T_0 \; dS \tag{4.34}$$

Special Numerical Features

Treatment of Rigid Region

In the machining process, situations do arise in which rigid zones exist; the plastic deformations are concentrated on the primary and secondary deformation regions. The rigid zones are characterized by a very small value of effective strain rate in comparison with that in the deforming body. Because the effective strain rate appears in the denominator of the stiffness matrix, the system of Eq. (4.24) becomes very ill-conditioned when the rigid regions are included in the analysis. This numerical problem is overcome by assuming that the stress-strain rate relationship in Eq. (4.9) is approximated by:

$$\dot{\varepsilon}_{ij} = \frac{3}{2} \frac{\dot{\bar{\varepsilon}}_o}{\bar{\sigma}_o} s_{ij}, \qquad \text{with} \qquad \bar{\sigma}_o = \bar{\sigma}(\varepsilon, \dot{\bar{\varepsilon}}_o, T), \qquad \text{for} \qquad \dot{\bar{\varepsilon}} \le \dot{\bar{\varepsilon}}_o \tag{4.35}$$

where $\dot{\bar{\varepsilon}}_o$ is the limiting strain rate and takes an assigned value. Too large a value of the limiting strain rate results in a solution in which the strain rate of the rigid zone becomes unacceptably large. On the other hand, if we choose too small a value of limiting strain rate, then the convergence of the iteration solution deteriorates considerably. For some applications (Gangjee, 1987; Todt, 1990), a value of $\dot{\bar{\varepsilon}}_o$ two orders of magnitude less than the average effective strain rate works well.

Treatment of Neutral Point

In the machining process, a stagnation point of the flow exits on the non-perfectly sharp tool cutting edge with material above the stagnation point flowing into the chip and material below the stagnation point flowing back into the workpiece. The velocity, and hence the frictional stress, changes direction at the neutral point. Because the location of the neutral point is not known *a priori*, special numerical techniques must be used when applying the friction boundary conditions. This problem is frequently observed in metal forming, such as ring compression, rolling, and forging. Among several methods, we use a modified form of the friction stress term proposed by Chen and Kobayashi (1978) as:

$$T_f = \tau_f \left[\frac{2}{\pi} \tan^{-1} \left(\frac{|v_s|}{v_o} \right) \right] \tag{4.36}$$

where T_f is the modified friction stress, τ_f is the friction stress, v_s is the sliding velocity of a material at the tool-workpiece interface, and v_o is a small positive number compared to v_s.

Incompressibility Condition

Because the penalty constant, which is used to remove the incompressibility, is very large, the second term of the global stiffness matrix Eq. (4.25) dominates in the velocity equilibrium equation. If the second term is non-singular, only the trivial solution $\mathbf{v} = 0$ may be possible. One of the methods making the second term singular is to use reduced and selective integration. This method was successfully applied to the incompressible visco-flow problems by Malkus and Hughes (1978). A second-order Gaussian quadrature integration is used for the first term and a first-order point formula for the second term.

The proper choice of penalty constant is important in successful simulation. Too large a value of C_K can cause difficulties in convergence, while too small a value results in unacceptably large volumetric strain. Numerical tests show that an appropriate value can be estimated by restricting volumetric strain rate to 0.0001 to 0.001 times the average effective strain rate.

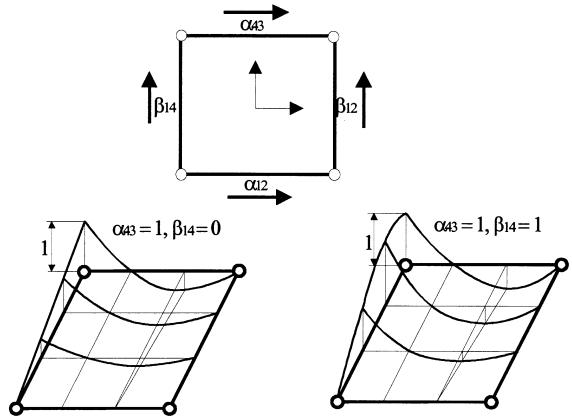


FIGURE 4.2 Weighing functions. Reprinted from Kim, H.W. and Sin, H-C., Development of a thermo-viscoplastic cutting model using finite element method, *Int. J. Mach. Tools Manuf.*, 36(3), 1996, 379–397, with kind permission from Elsevier Science Ltd., The Boulevard, Lanford Lane, Kidlington, OX5 1GB, UK.

Upwind Scheme

It is well known that the standard discretization of energy equations involving a convection term, when the velocity is high, may give rise to spurious oscillations that completely blur out the solution of the problem. Those oscillations were first observed when finite-difference methods were used while studying heat and mass transfer problems.

We choose the weighting functions as shown in Fig. 4.2 (Heinrich et al., 1977). We can write, for example, the shape function N_i for node i in a quadrilateral element as:

$$N_i = N_i(\xi) \cdot N_i(\eta) \tag{4.37}$$

Similarly, the weighting functions can be written as:

$$W_i = W_i(\xi) \cdot W_i(\eta) \tag{4.38a}$$

$$W_i(\xi) = N_i(\xi) + \alpha_{ij} \frac{3}{4}(1 - \xi)(1 + \xi) \tag{4.38b}$$

$$W_i(\eta) = N_i(\eta) + \beta_{ik} \frac{3}{4}(1 - \eta)(1 + \eta) \tag{4.38c}$$

In the above, the subscript j (or k) is that of the adjacent node lying along the same element side and

$$|\alpha_{ij}| = |\alpha_{ji}|, \quad |\beta_{ij}| = |\beta_{ji}| \tag{4.39}$$

In this chapter, as a full upwind scheme is used, $\alpha_{ij}(\beta_{ji})$ is either 1 or -1 and is determined by the direction of the average velocity vector along the element side ij.

Boundary Condition Problems

Velocity Field: Free Surface and Chip-Tool Contact

The geometry and boundary conditions shown in Fig. 4.3 are used to model orthogonal cutting. From rake angle, clearance angle, tool edge radius, and depth of cut, the geometries of the tool and workpiece are determined. The boundary conditions include a cutting velocity (v) and no flow normal to the base of the control volume or tool surface (D-E). A-B, C-D, E-F are free surfaces of the chip, and the boundaries of the chip free surfaces must be determined iteratively during simulation as they are initially unknown.

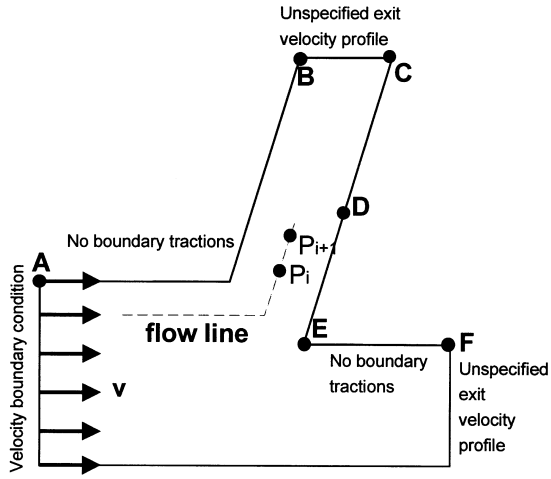


FIGURE 4.3 Velocity boundary condition used for the cutting model. Reprinted from Kim, H.W. and Sin, H-C., Development of a thermo-viscoplastic cutting model using finite element method, *Int. J. Mach. Tools Manuf.*, 36(3), 1996, 379–397, with kind permission from Elsevier Science Ltd., The Boulevard, Lanford Lane, Kidlington, OX5 1GB, UK.

In this chapter, the free surface of the chip can be calculated by requiring that the normal component of the surface velocity be zero. This method was originally proposed by Zienkiewicz et al. (1978) and applied to an extrusion problem. If a fixed point of known coordinate values on a free surface exists, from that point the coordinates of free surface can be calculated by integrating as follows:

$$\frac{dy'}{dx'} \equiv \alpha_s \qquad \text{or} \qquad y' = \int_0^{x'} \alpha_s \, dx' \tag{4.40}$$

where x' and y' are the coordinates of the surface and α_S represents the direction of velocity vector.

The contact length is a critical parameter in coupling the thermal and material flow aspects of the cutting model because it defines the thermal conduction path between chip and tool. Using the condition that the normal stress at every node along the chip-tool interface is negative, the contact length can be calculated. The calculation procedures are as follows:

- Assume the initial chip configuration and chip-tool contact length.
- Calculate the velocities of all nodes and stresses of the nodes along the chip-tool interface by the finite element method.
- Calculate the coordinates of free surfaces from Eq. (4.40). If some nodes penetrate the tool face, they are assumed to be in contact with the tool and the free surfaces are recalculated. A positive stress at any node along the interface means that the chip has separated from the rake face of the tool. Where the chip separates from the tool is then treated as a free surface, and its position is recalculated.
- Continue until the normal velocity component on the free surfaces of the chip is found to be zero and the normal stress at every node along the chip-tool interface is negative.

Temperature Field

Temperature boundary conditions are shown in Fig. 4.4. Most external surfaces that contact the air are taken to be adiabatic, that is, heat losses to the surroundings by convection and radiation are assumed to be zero. On the right-hand side and the lower boundary of the workpiece, temperature gradients normal to the boundary are very small and hence are taken to be zero. On the left-hand boundary of the workpiece, room temperature is assumed.

Infinite elements (Peter, 1980) are introduced to represent the areas of tool that are much larger in comparison with the deformation areas of the workpiece. The elements on the boundaries of AB in Fig. 4.4 are infinite elements in the longitudinal direction. The shape function of the infinite element

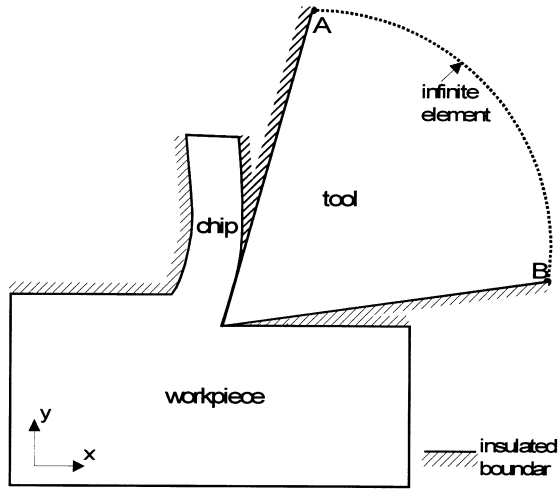


FIGURE 4.4 Temperature boundary condition used for the cutting model.

is given by the following equation to represent the infinity of the element in η-direction:

$$M_i(\xi,\eta) \;=\; N_i(\xi,\eta)\cdot g_i(\xi,\eta) \tag{4.41}$$

where i is the nodal number and N_i is the shape function of the normal element; g_i is called the decay function and is given as follows:

$$g_i(\xi,\eta) \;=\; \left(\frac{\eta_i-\eta_o}{\eta-\eta_o}\right)^m \tag{4.42}$$

$\eta_o < -1$ represents some origin point in η-direction and m is set to 1.2 in this study.

Effective Strain Calculation

The solution is velocity or strain rate in the steady state. Thus, the effective strain can be calculated by the following procedure.

All values of effective strain rate at the center points of finite elements can be interpolated by those on nodal points. Take the two neighboring points P_i and P_{i+1} on a flow line in Fig. 4.3 as examples. After checking to see which element the point P_i belongs to, velocity components V_i on the point P_i can be determined by linear interpolation. This check is made numerically using the natural coordinate system. The next point P_{i+1} on the flow line is calculated from:

$$P_{i+1} \;=\; P_i + v_i \times \Delta t \tag{4.43}$$

where Δt denotes the time increment and can be adjusted appropriately. After checking to see which element the point P_{i+1} belongs to, the effective strain rate on the point can be calculated by linear interpolation. The interpolated strain rate is added incrementally to the value of effective strain at the previous location as:

$$\bar{\varepsilon}_{i+1} \;=\; \bar{\varepsilon}_i + \dot{\bar{\varepsilon}}_{i+1} \times \Delta t \tag{4.44}$$

Using the above procedure, a new grid system is constructed by the points on the selected flow lines. Finally, the effective strain at the center of a finite element can be obtained through linear interpolation of the values on the new grid system.

Computational Procedures

Four iterative cycles are performed during a cutting process simulation. The first solves the viscoplastic equations for the velocity and strain rate distributions in the chip and workpiece. At ambient temperature conditions and assumed strain distribution in the chip and workpiece, the nodal velocity and strain rate distributions are calculated by the direct iteration method using the FEM. After each iteration, the strain rate is compared to the initial value. The iterations are continued until the initial and calculated strain rates coincide. Once the velocities have been determined, the temperature is calculated. It is considered that the heat generation in the chip and workpiece is due to plastic deformation and frictional heating and that the chip-tool contact length defines the thermal conduction path between chip and tool. Because the elevated temperatures will significantly alter the material and thermal properties, an iterative solution is again required until the temperatures converge. Next, chip geometry is determined. After the solution of the viscoplastic and temperature equations, the computed velocities on the surface of the chip are checked to ensure that they are parallel to the free surface. If this condition is not satisfied, the coordinates of the chip free surface are updated, the grid is remeshed and the velocity and temperature distributions are computed again. The final step is to calculate the strain distributions. Flow lines and strain distributions are determined by linear interpolation. The iterations are continued until the initial and calculated strains converge. In calculating the temperature and strain distributions, it was found that the procedure converged fairly rapidly, usually within four or five iterations. Figure 4.5 shows the flowchart of the cutting process simulation program.

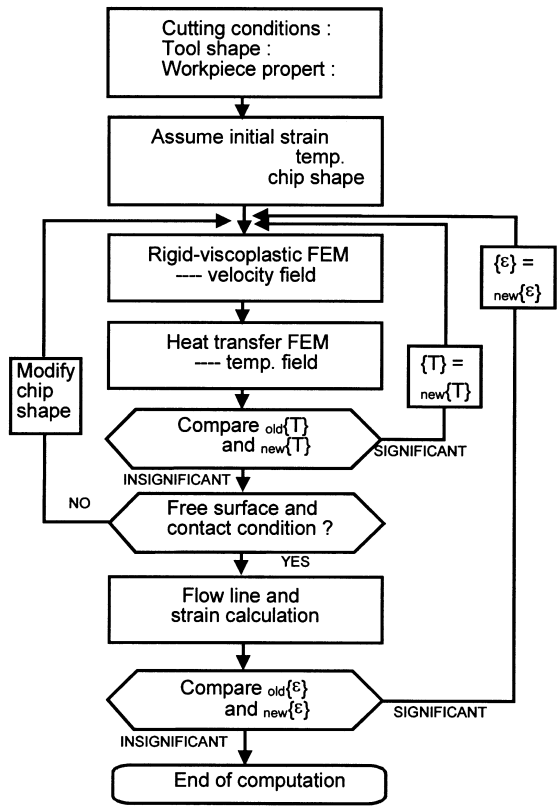


FIGURE 4.5 Flowchart of cutting analysis. Reprinted from Kim, H.W. and Sin, H-C., Development of a thermo-viscoplastic cutting model using finite element method, *Int. J. Mach. Tools Manuf.*, 36(3), 1996, 379–397, with kind permission from Elsevier Science Ltd., The Boulevard, Lanford Lane, Kidlington, OX5 1GB, UK.

4.4 Application

Material Properties and Friction Forces Determination

We used the method proposed by Oxley and Hasting (1976) to determine the flow stress σ. It is as follows.

The flow stress of the workpiece can be represented by the well-known stress-strain relation:

$$\sigma = \sigma_1 \cdot \varepsilon^n \tag{4.45}$$

where ε is the uniaxial flow strain, σ_1 is the stress, and n is the strain hardening index. For each of the plain carbon steels considered, the results of Oyane et al. (1967) show that a velocity-modified temperature parameter T_{mod} gives a good fit with the experimental results for a given temperature and strain rate, and that the values of σ_1 and n thus obtained can be plotted against T_{mod} to give curves representing the flow stress properties of each of the steels. The velocity-modified temperature parameter is defined by:

$$T_{\text{mod}} = T\left[1 - \nu \log\left(\frac{\dot{\varepsilon}}{\dot{\varepsilon}_0}\right)\right] \tag{4.46}$$

where T is the temperature, $\dot{\varepsilon}$ is the uniaxial strain rate, and ν and $\dot{\varepsilon}_0$ are material constants. By representing these curves mathematically and using rescaling functions, continuous changes in σ_1 and n over the ranges of velocity-modified temperature and carbon content considered have been represented by a relatively simple set of functions. The σ_1 and n curves obtained from these functions are given in Fig. 4.6. The material constants, ν and $\dot{\varepsilon}_0$ are 0.09 and 1/s, respectively.

In calculating temperatures, the appropriate temperature-dependent thermal properties were determined in the following way. The influence of carbon content on specific heat is found to be small and the following equation (Oxley 1989)

$$C_p/(\text{Jkg}^{-1}\text{K}^{-1}) = 420 + 0.504\ T/°\text{C} \tag{4.47}$$

can be used for all of the steels. However, there is a marked influence of carbon content on thermal conductivity, and for the 0.2% carbon steel it is given as (Oxley 1989):

$$k_p/(\text{Wm}^{-1}\text{K}^{-1}) = 54.17 - 0.0298\ T/°\text{C} \tag{4.48}$$

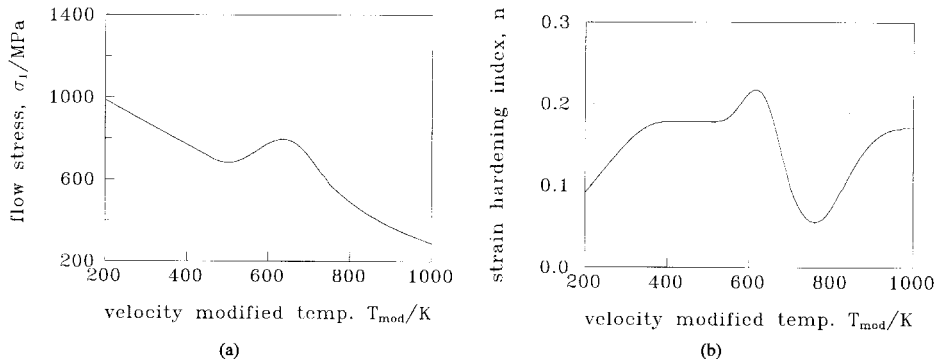


FIGURE 4.6 Flow stress results plotted against velocity-modified temperature: (a) flow stress, σ_1; (b) strain hardening index, n. Reprinted from Kim, H.W. and Sin, H-C., Development of a thermo-viscoplastic cutting model using finite element method, *Int. J. Mach. Tools Manuf.*, 36(3), 1996, 379–397, with kind permission from Elsevier Science Ltd., The Boulevard, Lanford Lane, Kidlington, OX5 1GB, UK.

For the friction force, the following equation proposed by Usui and Shirakashi (1982) is used:

$$\tau_f = k\left[1 - \exp\left(-\lambda\frac{\sigma_n}{k}\right)\right] \tag{4.49}$$

where λ is the experimental constant determined by tool and workpiece, σ_n is the normal stress, and k is the shear stress. Equation (4.49) satisfies the following two boundary conditions:

- As normal stress increases, friction force approaches shear stress.
- When normal stress is decreased to zero, Coulomb's law is realized.

$$\mu = \frac{d\tau_f}{d\sigma_n} = \lambda\exp\left(-\lambda\frac{\sigma_n}{k}\right) \cong \lambda\left(1 - \lambda\frac{\sigma_n}{k}\right) \tag{4.50}$$

Numerical Simulation Example

The thermo-viscoplastic cutting model was used to simulate the cutting of 0.2 % carbon steel. Cutting conditions and tool configuration are shown in Table 4.1. Figure 4.7 represents the initially assumed chip geometry. Chip geometry is adjusted iteratively during the simulation using the free surface condition

TABLE 4.1 Cutting Conditions and Tool Configuration for the Simulation

Cutting speed (m/s)	2.16
Depth of cut (mm)	0.2
Rake angle (degree)	12
Tool edge radius (mm)	0.1
Clearance angle (degree)	5
Width of cut (mm)	1

Reprinted from Kim, H.W. and Sin, H-C., Development of a thermo-viscoplastic cutting model using finite element method, *Int. J. Mach. Tools Manuf.*, 36(3), 1996, 379–397, with kind permission from Elsevier Science Ltd., The Boulevard, Lanford Lane, Kidlington, OX5 1GB, UK.

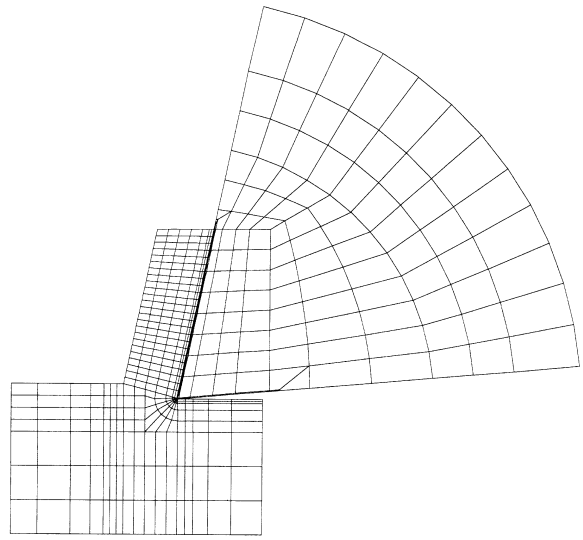


FIGURE 4.7 Finite element model with initial chip geometry for 12° rake angle, 5° clearance angle, 0.01 mm tool edge radius, and 0.02 mm depth of cut. Reprinted from Kim, H.W. and Sin, H-C., Development of a thermo-viscoplastic cutting model using finite element method, *Int. J. Mach. Tools Manuf.*, 36(3), 1996, 379–397, with kind permission from Elsevier Science Ltd., The Boulevard, Lanford Lane, Kidlington, OX5 1GB, UK.

TABLE 4.2 Simulation Results

Principal force, Fc (N)	375.18
Thrust force, Ft (N)	120.82
Chip thickness, t2 (mm)	0.50
Chip-tool contact length, *ln* (mm)	0.38
Temp. in chip-tool contact, Tint (°C)	651.44
Maximum temperature, Tmax (°C)	680.66

Reprinted from Kim, H.W. and Sin, H-C., Development of a thermo-viscoplastic cutting model using finite element method, *Int. J. Mach. Tools Manuf.*, 36(3), 1996, 379–397, with kind permission from Elsevier Science Ltd., The Boulevard, Lanford Lane, Kidlington, OX5 1GB, UK.

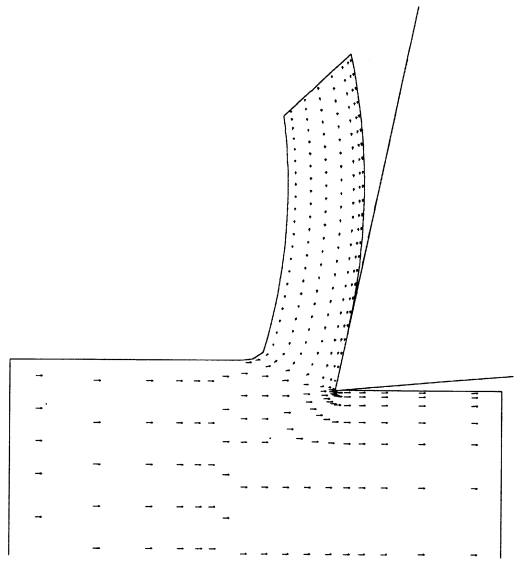


FIGURE 4.8 Predicted chip geometry and velocity vectors. Reprinted from Kim, H.W. and Sin, H-C., Development of a thermo-viscoplastic cutting model using finite element method, *Int. J. Mach. Tools Manuf.*, 36(3), 1996, 379–397, with kind permission from Elsevier Science Ltd., The Boulevard, Lanford Lane, Kidlington, OX5 1GB, UK.

and contact condition. This cutting model can predict velocity, stress, strain, and strain rate distributions in the chip and workpiece, and temperature distribution in the chip and tool. The outputs from the cutting model are shown in Table 4.2.

Figure 4.8 shows the velocity vectors in the chip and workpiece, and the chip geometry. The velocity vectors are tangential to the free surface of the chip, thus confirming the validity of the predicted chip geometry. Contours of effective strain rate are shown in Fig. 4.9. The maximum effective strain rate reaches its maximum value of 60,751/s ahead of the tool edge. It exhibits a large gradient in the finite shear zone ahead of the tool edge and increases toward the tool edge. Figure 4.10 shows the contours of maximum shear stress. They have a maximum of 612 MPa, and also exhibit a finite region in the shear zone ahead of the tool. Figure 4.11 shows the contours of temperature in the chip and tool. Temperatures in the chip-tool contact region are higher than those in any other. The effective strain contours in Fig. 4.12 have a maximum of 3.13. It can be seen that the deformation in the chip is not constant and is getting larger toward the tool edge.

Figure 4.13 shows the measured and simulated principal forces and thrust forces for the speed of 2.16 m/s and the depth of cut from 0.07 mm to 0.2 mm. As expected, the principal forces and the thrust forces increase with increasing depth of cut. Good correlation is found for the principal forces and the thrust forces over the entire range of depth of the cut tested. In Fig. 4.14, the contours of effective strain rate for the depth of cut of 0.1 mm are compared with those for 0.2 mm. As the depth of cut increases,

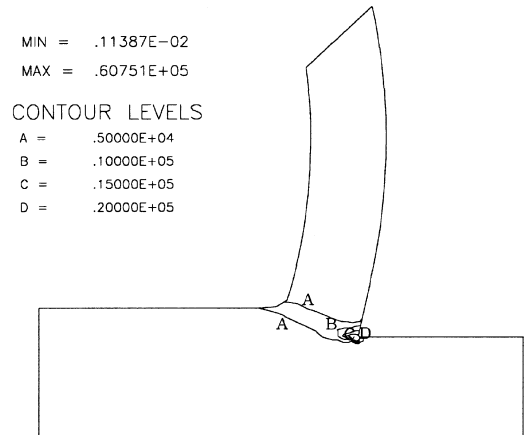


FIGURE 4.9 Contours of predicted effective strain rate. Reprinted from Kim, H.W. and Sin, H-C., Development of a thermo-viscoplastic cutting model using finite element method, *Int. J. Mach. Tools Manuf.*, 36(3), 1996, 379–397, with kind permission from Elsevier Science Ltd., The Boulevard, Lanford Lane, Kidlington, OX5 1GB, UK.

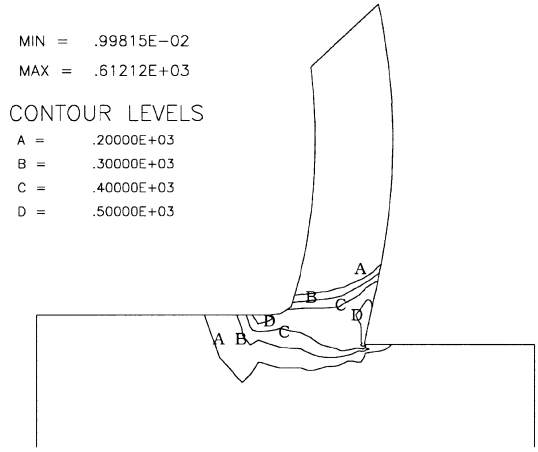


FIGURE 4.10 Contours of predicted maximum shear stress. Reprinted from Kim, H.W. and Sin, H-C., Development of a thermo-viscoplastic cutting model using finite element method, *Int. J. Mach. Tools Manuf.*, 36(3), 1996, 379–397, with kind permission from Elsevier Science Ltd., The Boulevard, Lanford Lane, Kidlington, OX5 1GB, UK.

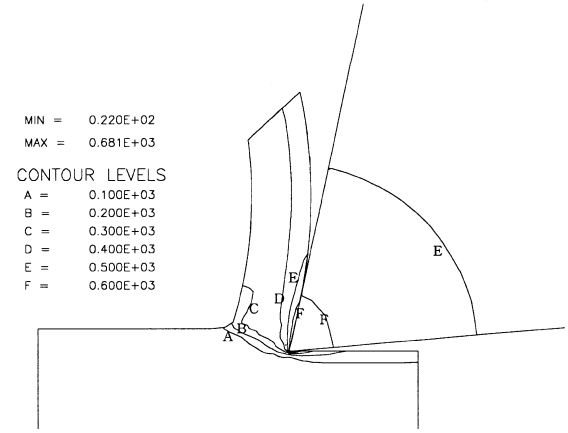


FIGURE 4.11 Contours of predicted temperature. Reprinted from Kim, H.W. and Sin, H-C., Development of a thermo-viscoplastic cutting model using finite element method, *Int. J. Mach. Tools Manuf.*, 36(3), 1996, 379–397, with kind permission from Elsevier Science Ltd., The Boulevard, Lanford Lane, Kidlington, OX5 1GB, UK.

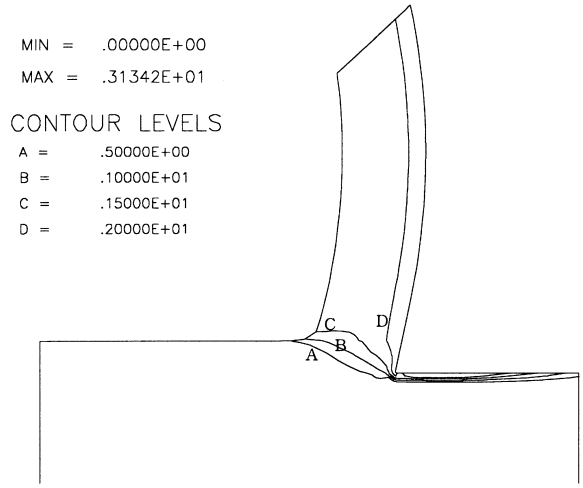


FIGURE 4.12 Contours of predicted effective strain. Reprinted from Kim, H.W. and Sin, H-C., Development of a thermo-viscoplastic cutting model using finite element method, *Int. J. Mach. Tools Manuf.*, 36(3), 1996, 379–397, with kind permission from Elsevier Science Ltd., The Boulevard, Lanford Lane, Kidlington, OX5 1GB, UK.

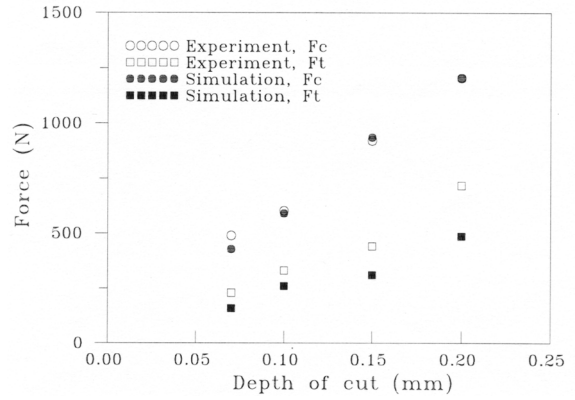


FIGURE 4.13 Comparison of predicted and measured cutting forces with depth of cut for a cutting speed of 2.16 m/s. Reprinted from Kim, H.W. and Sin, H-C., Development of a thermo-viscoplastic cutting model using finite element method, *Int. J. Mach. Tools Manuf.*, 36(3), 1996, 379–397, with kind permission from Elsevier Science Ltd., The Boulevard, Lanford Lane, Kidlington, OX5 1GB, UK.

the maximum effective strain rate decreases but the deformation region does not change much. It is consistent with Stevenson and Oxley's experimental observation (1969; 1970). Figure 4.15 shows the measured and simulated principal forces and thrust forces for the depth of cut of 0.1 mm. The two cutting speeds tested are 2.16 m/s and 3.02 m/s. It can be seen that the principal forces and the thrust forces decrease as the cutting speed increases. The simulated values and the experimental values are in good agreement in the range of the cutting speed tested. Figure 4.16 shows the contours of the effective strain rate for the cutting speed of 2.16 m/s compared with those of 3.02 m/s. As the cutting speed increases, the deformation region extends larger in the workpiece and the maximum effective strain rate increases. It is also consistent with Stevenson and Oxley's experimental observation (1969; 1970). The thickness and the curl radius of the chip decrease with increasing cutting speed. Similar agreement was found for the depth of cut of 0.2 mm and cutting speeds of 2.16 m/s and 3.02 m/s.

For the cutting temperature, it is very interesting to compare the simulation results with Tay et al.'s results (1974; 1976). They first described the use of the FEM for calculating machining temperature distributions in 1974. This procedure depended on a strain rate field being available for each set of

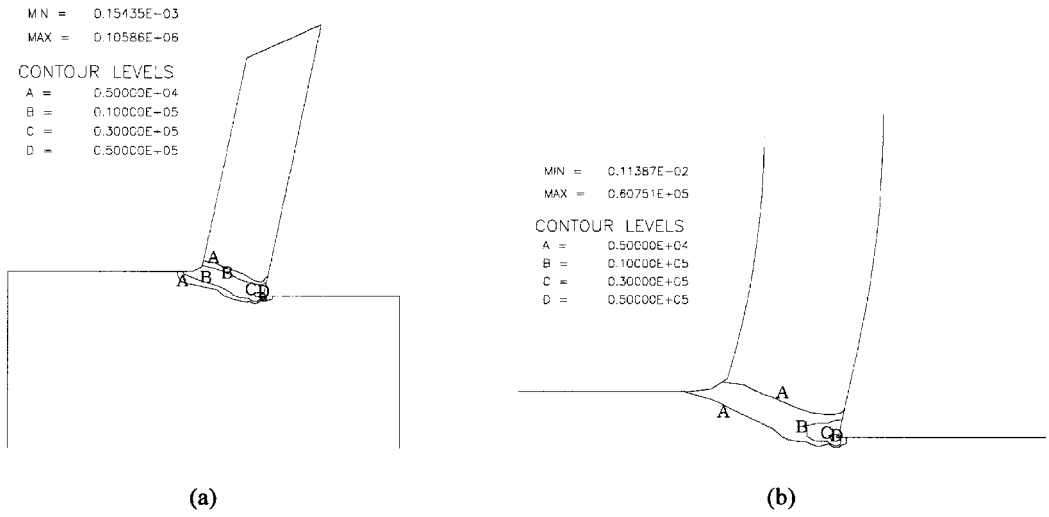


FIGURE 4.14 Effect of depth of cut on the contours of effective strain rate: (a) t1 = 0.1 mm; (b) t1 = 0.2 mm. Reprinted from Kim, H.W. and Sin, H-C., Development of a thermo-viscoplastic cutting model using finite element method, *Int. J. Mach. Tools Manuf.*, 36(3), 1996, 379–397, with kind permission from Elsevier Science Ltd., The Boulevard, Lanford Lane, Kidlington, OX5 1GB, UK.

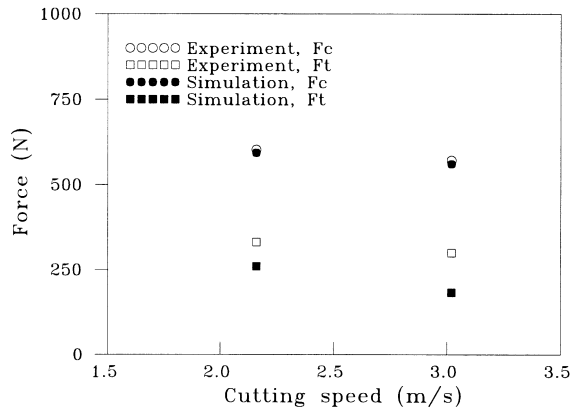


FIGURE 4.15 Comparison of predicted and measured cutting forces with cutting speed for depth of cut of 0.1 mm. Reprinted from Kim, H.W. and Sin, H-C., Development of a thermo-viscoplastic cutting model using finite element method, *Int. J. Mach. Tools Manuf.*, 36(3), 1996, 379–397, with kind permission from Elsevier Science Ltd., The Boulevard, Lanford Lane, Kidlington, OX5 1GB, UK.

conditions; quick-stop tests on specimens printed with fine grids were conducted for this purpose. A simplified version was later described (1976). However, their method is not predictive because it requires inputs that cannot be arrived at from material properties but must be obtained from metal-cutting tests for the specific combination of tool and workpieces, tool geometry, and cutting conditions used.

Figure 4.17 shows the simulated temperature distributions compared with Tay et al.'s results (1974). Very good agreement is found, especially for the maximum temperature and the location of maximum temperature as shown in Table 4.3. In Fig. 4.18, the temperature distributions for relatively high cutting speed from Tay et al.'s results (1976) are given, along with the corresponding simulated results. Good agreement is also found for the temperature distributions, the maximum temperature, and the location of maximum temperature.

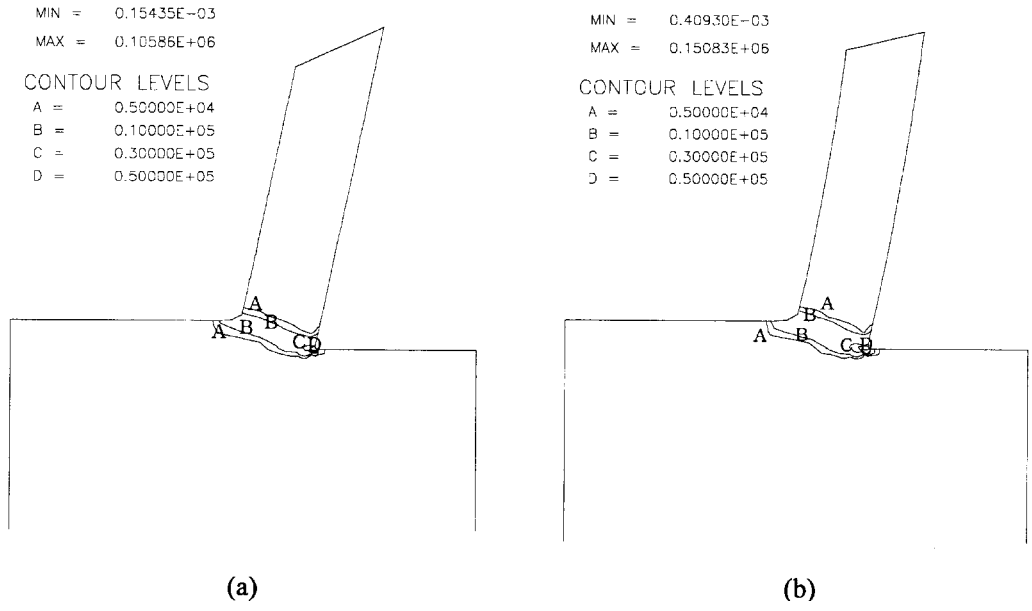


FIGURE 4.16 Effect of cutting speed on the contours of effective strain rate: (a) v = 2.16 m/s; (b) v = 3.02 m/s. Reprinted from Kim, H.W. and Sin, H-C., Development of a thermo-viscoplastic cutting model using finite element method, *Int. J. Mach. Tools Manuf.*, 36(3), 1996, 379–397, with kind permission from Elsevier Science Ltd., The Boulevard, Lanford Lane, Kidlington, OX5 1GB, UK.

TABLE 4.3 Comparison with Tay et al.'s Results (1974)

FEM simulation	Tmax (°C)	607.2
	Location of Tmax (mm)	0.525
Tay et al.'s results	Tmax (°C)	615
	Location of Tmax (mm)	0.560

Reprinted from Kim, H.W. and Sin, H-C., Development of a thermo-viscoplastic cutting model using finite element method, *Int. J. Mach. Tools Manuf.*, 36(3), 1996, 379–397, with kind permission from Elsevier Science Ltd., The Boulevard, Lanford Lane, Kidlington, OX5 1GB, UK.

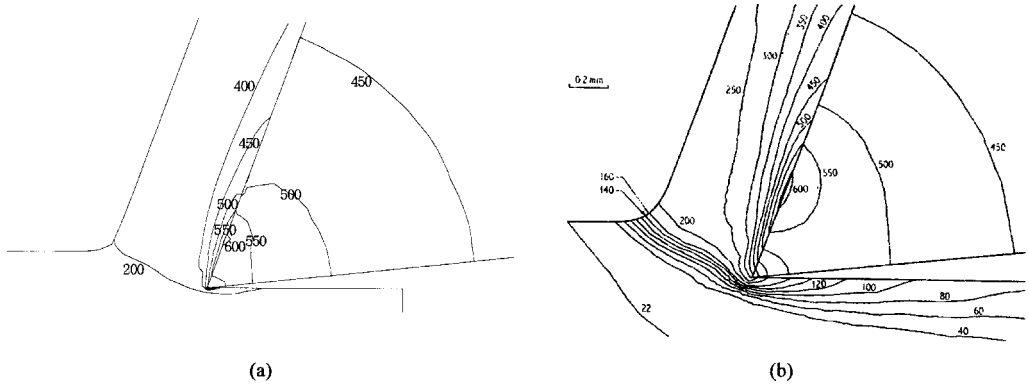


FIGURE 4.17 Temperature distribution for the comparison with Tay et al.'s results (1974): (a) predicted temperature distribution; (b) Tay et al.'s results. Reprinted from Kim, H.W. and Sin, H-C., Development of a thermo-viscoplastic cutting model using finite element method, *Int. J. Mach. Tools Manuf.*, 36(3), 1996, 379–397, with kind permission from Elsevier Science Ltd., The Boulevard, Lanford Lane, Kidlington, OX5 1GB, UK.

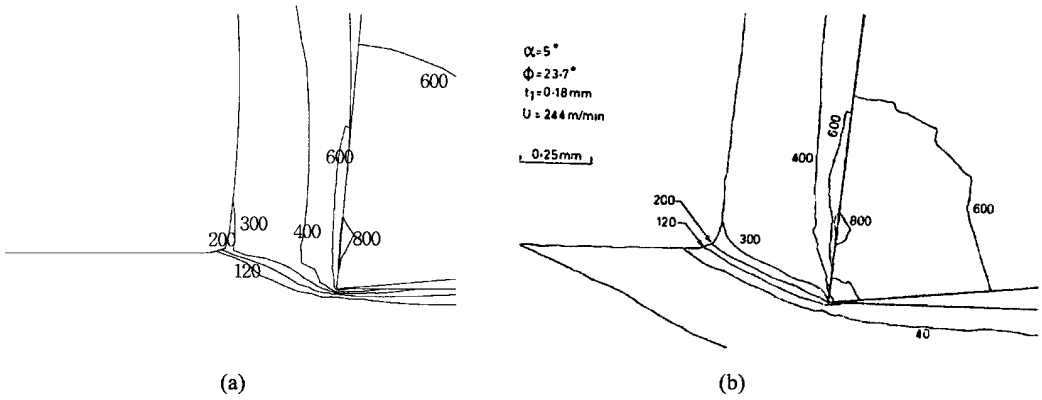


FIGURE 4.18 Temperature distribution for the comparison with Tay et al.'s results (1976): (a) predicted temperature distribution; (b) Tay et al.'s results. Reprinted from Kim, H.W. and Sin, H-C., Development of a thermo-viscoplastic cutting model using finite element method, *Int. J. Mach. Tools Manuf.*, 36(3), 1996, 379–397, with kind permission from Elsevier Science Ltd., The Boulevard, Lanford Lane, Kidlington, OX5 1GB, UK.

4.5 Conclusions

The basic concepts for a thermo-viscoplastic cutting model have been described in some detail for the practical application to the simulation of machining processes. After a review of the literature, it is shown how the finite element method (FEM) can be implemented to allow development of the cutting model. A number of special problems arise during development of numerical methods to realize such a capability. A successful application is illustrated through orthogonal cutting simulation for 0.2% carbon steel. For an orthogonal steady-state cutting process, we believe the thermo-viscoplastic cutting model allows one to understand the cutting mechanism and to accurately predict cutting forces, cutting temperatures, chip thickness, chip curling, etc.

Although there is active research on modeling the machining process, there are numerous research topics that should be incorporated into the ideal cutting model. In the future, one can expect developments in the following fields:

- For Lagrangian approaches, modeling the plane strain to plane stress transition during chip formation, establishing the chip separation criterion, and reducing the computational time
- For Eulerian coordinate frames, considering the elastic effect, that is, computing the residual stresses of the machined surface
- Developing the cutting model with oblique cutting processes
- Analyzing the machining process with negative rake angles, variable rake face configuration, built-up edge, and cutting fluid effects considered
- Consideration of tool behavior and wear
- Analyzing and modeling the machining process for "difficult-to-machine" materials such as nickel-based alloys
- Obtaining accurate data on flow stress at high stress, strain rate, and temperature for most engineering materials
- Determining the friction characteristics between chip and tool that is suitable for machining

Acknowledgments

The majority of the work reported herein was reprinted from *Int. J. Mach. Tool. Manufact.*, Vol. 36, Kug Weon Kim and Hyo-Chol Sin, Development of a Thermo-Viscoplastic Cutting Model Using Finite Element Method, pp. 379–397, Copyright 1995, with kind permission from Elsevier Science Ltd., The

Boulevard, Langford Lane, Kidlington 0X5 1 GB, U.K. The authors gratefully acknowledge the permission given by Elsevier Science Ltd.

References

Bagchi A and Wright PK (1987). Stress analysis in machining with the use of sapphire tools. *Proc R Soc Lond* A 409:99–113.

Carrol III JT and Strenkowski (1988). Finite element models of orthogonal cutting with application to single diamond turning. *Int J Mech Sci* 30(12):899–920.

Chen CC and Kobayashi S (1980). Rigid plastic finite element of ring compression. In *Applications of Numerical Methods to Forming Processes*. ASME Applied Mechanics Division, 28:163–174.

Gangjee T (1987). Friction and lubrication in finite element analysis of forging processes. Ph.D. dissertation, Clemson University, Clemson, SC.

Chandrasekar H and Kapoor DV (1965). Photoelastic analysis of tool-chip interface stresses. *ASME J Eng Ind*, 87:495–502.

Doyle ED, Horne JG and Tabor D (1979). Frictional interactions between chip and rake face in continuous chip formation. *Proc R Soc Lond* A 366:173–183.

Hashemi J, Tseng AA and Chou PC (1994). Finite element modeling of segmental chip formation in high-speed orthogonal cutting. *J Mat Eng and Performance* 3(5):712–721.

Heinrich JC, Huyakorn PS and Zienkiewicz OC (1977). An 'upwind' finite element scheme for two-dimensional convective transport equation. *Int J Num Meth Engng* 11:131–143.

Ikawa N, Shimada S, Tanaka H and Ohmori G (1991). An atomistic analysis of nanometric chip removal as affected by too-work interaction in diamond turning. *Ann CIRP* 40:551–554.

Iwata K, Osakada K and Terasaka Y (1984). Process modeling of orthogonal cuting by the rigid-plastic finite element method. *ASME J Engng Matl and Tech* 106:132–138.

Joshi VS, Dixit PM and Jain VK (1994). Viscoplastic analysis of metal cutting by finite element method. *Int J Mach Tools Manufact* 34(4):553–571.

Kato S, Yamaguchi K and Yamada M (1972). Stress distribution at the interface between tool and chip in machining. *ASME J Engng Ind* 94:683–688.

Kececioglu D (1958). Shear strain rate in metal cutting its effects on shear-flow stress. *ASME J Engng Ind*, pp 158–168.

Kim KW and Sin HC (1996). Development of a thermo-viscoplastic cutting model using finite element method. *Int J Mach Tools Manufact* 36(3):379–397.

Komvopoulos K and Erpenbeck S (1991). Finite element modeling of orthogonal metal cutting. *ASME J Engng Ind* 113:253–267.

Lajczok MR (1980). A study of some aspects of metal machining using the finite element method. PhD dissertation, North Carolina State University.

Lin ZC and Lin SY (1992). A coupled finite element model of thermal-elastic-plastic large deformation for orthogonal cutting. *ASME J Engng Ind* 114:218–226.

Lin ZC and Pan WC (1993). A thermoelastic-plastic large deformation model for orthogonal cutting with tool frank wear. Part I: Computational procedures. *Int J Mech Sci* 35(10):829–840.

Lin ZC and Pan WC (1993). A thermoelastic-plastic large deformation model for orthogonal cutting with tool frank wear. Part II: Machining application. *Int J Mech Sci* 35(10):841–850.

Lin ZC and Liu CC (1996). Analysis of orthogonal finish machining using tungsten carbide and diamond tools of different heat transfer coefficients. *Int J Mach Tools Manufact* 36(1):73–88.

Malkus DS and Hughes TJR (1978). Mixed finite element methods - reduced and selective integration techniques : a unification of concepts. *Comp Meth Appl Mech and Engng* 15:63–81.

Malvern LE (1969). *Introduction to the Mechanics of a Continuous Medium*, Prentice Hall, Englewood Cliffs, New Jersey.

Merchant ME (1944). Basic mechanics of the metal cutting process. *ASME J App Mech* 11:168–175.

Morcos WA (1980). A slip line field solution of the free oblique continuous cutting problem in conditions of light friction at chip-tool interface. *ASME J Engng Ind* 102:310–314.

Moriwaki T, Sugimura N and Luan S (1993). Combined stress, material flow and heat analysis of orthogonal micromachining of copper. *Ann CIRP* 42:75–78.

Nakayama K and Arai M (1976). On the storage of data on metal cutting process. *Ann CIRP* 25:13–18.

Oxley PLB (1989). *Mechanics of Machining: an Analytical Approach to Assessing Machinability*, Ellis Horwood, Chichester.

Oxley PLB and Hastings WF (1976). Minimum work as a possible criterion for determining the frictional conditions at the tool/chip interface in machining. *Phil Trans R Soc Lond* 282:565–584.

Oyane M, Takashima FO, Sakada K and Tanaka H (1967). The behaviour of some steels under dynamic compression. *10th Japan Congress on Testing Materials*, pp 72–76.

Peter B (1980). More on infinite elements. *Int J Num Meth Engng* 15:1613–1626.

Shaw MC, Cook NH and Finne I (1953). The shear-angle relationship in metal cutting. *ASME* 75:273–288.

Shih AJ (1995). Finite element simulation of orthogonal metal cutting. *ASME J Engng Ind* 117:84–93.

Shih AJ (1996). Finite element analysis of orthogonal metal cutting mechanics. *Int J Mach Tools Manufact* 36(2):255–273.

Shih AJ and Yang HT (1993). Experimental and finite element predictions of residual stresses due to orthogonal metal cutting. *Int J Num Meth Eng*, 36:1487–1507.

Shimada S, Ikawa N, Tanaka H, Ohmori G and Uchikoshi J (1993). Feasibility study on ultimate accuracy in microcutting using molecular dynamics simulation. *Ann CIRP,* 42(1):91–94.

Shirakashi T and Usui E (1973). Friction characteristics on tool face in metal machining. *J Japan Soc Prec Eng,* 39(9):966.

Stevenson MG (1975). Torsional Hopkinson-bar tests to measure stress-strain properties relevant to machining and high speed forming. In *Proceedings of 3rd North American Metalworking Research Conference*, Pittsburgh, Carnegie Press, 291–304.

Stevenson MG and Oxley PLB (1969/1970). An experimental investigation of the influence of speed and scale on the strain-rate in a zone of intense plastic deformation. *Proc Instn Mech Engrs*, 184:561–576.

Strenkowski JS and Carroll III JT (1985). A finite element method of orthogonal metal cutting. *ASME J Engng Ind* 107:346–354.

Strenkowski JS and Carroll III JT (1986). An orthogonal metal cutting model based on an eulerian finite element method. *Proc 13th NSF Conf on Prod Res and Tech*, pp 261–264.

Strenkowski JS and Mitchum GL (1987). An improved finite element model of orthogonal metal cutting. *Proc North American Manufacturing Res Conf*, Bethlehem, Pa, pp 506–509.

Strenkowski JS and Moon KJ (1990). Finite element prediction of chip geometry and tool/workpiece temperature distributions in orthogonal metal cutting. *ASME J Engng Ind* 112:313–318.

Tay AO, Stevenson MG and Davis G (1974). Using the finite element method to determine temperature distributions in orthogonal machining. *Proc Instn Mech Engrs* 188:627–638.

Tay AO, Stevenson MG, Davis G and Oxley PLB (1976). A numerical method for calculating temperature distributions in machining from force and shear angle measurements. *Int J Mach Tool Des Res* 16:335–349.

Todt M (1990). Simulating die-workpiece heat transfer during metal forming process. MS thesis, Clemson University, Clemson, SC.

Ueda and Manabe (1993). Rigid-plastic FEM analysis of three-dimensional deformation field in chip formation process. *Ann CIRP,* 42(1):35–38.

Usui E and Takeyama H (1960). A photoelastic analysis of machining stresses. *ASME J Engng Ind* 82:303–308.

Usui E and Shirakashi T (1982). Mechanics of machining-From descriptive to predictive theory. In *The Art of Cutting Metals-75 Years Later*, pp 13–35.

Wu JS, Dillon Jr W and Lu WY (1996). Thermo-viscoplastic modeling of machining process using a mixed finite element method. *ASME J Engng Ind* 118:470–482.

Zhang B and Bagchi A (1994). Finite element simulation of chip formation and comparison with machining experiment. *ASME J Engng Ind* 116:289–297.

Zienkiewicz OC, Jain, PC and Onate E (1978). Flow of solids during forming and extrusion : some aspects of numerical solutions. *Int J Solids Struct* 14:15–38.

5

Diagnostics for Monitoring Maintenance and Quality Manufacturing

5.1 Introduction

5.2 Monitoring Maintenance
Performance Prognosis and Health Diagnostics • The Situation/Trend Monitoring of Processes • The Relational Frames: Heuristics vs. Causality • Maintenance Policies: Preventive, Predictive, Proactive • Monitoring Architectures: Diagnoses Organization

5.3 Quality Manufacturing
Quality Measurement: Cognitive Scales • Quality Metrology and Abstract Standardisation Rules • Quality Approval Trials: An Example of Development • Quality Indices and Design-for-Comfort • Product Life-Cycle Checks and Automatic Testing

5.4 Condition Monitoring and Facilities Upkeeping
Product Inspection and Properties/Functional Checks • Measuring Equipment for Automatic Dimensional Inspection • Shop-floor Cells for On-process Automatic Testing • Dimensional Monitoring for Compliant Pieces Approval Tests

5.5 Trend Monitoring and Fit-with-Norms Restoring
The Experimental Setup for Vibro-acoustic Signatures Diagnostics • The Online Detection of Non-persistent Vibro-acoustic Images • The Evolutionary Power Density Signatures • The Wavelet Transform of Timbric Signatures • Sample Outputs and Case Discussion

5.6 Maintenance of the Monitoring System
Intelligent Measurements and Instrumental Reliability • Quality Maintenance of the Measuring Instrumentation • Measurement Process Control: Sample Results and Assessments

5.7 Conclusions

Rinaldo C. Michelini
University of Genova-Italy

Francesco Crenna
University of Genova-Italy

G.B. Rossi
University of Genova-Italy

5.1 Introduction

Economy of scope aims at prearranging productive means tailored to artifacts with customer-driven quality. The statement leads to several prescriptions for optimising the return on investments, such as just-in-time schedules, quick-response planning, lean engineering, quality functions deployment, flexible specialization, proactive maintenance, and other clever tricks to expand offers, improve quality, and reduce time-to-market with maximum exploitation of the available resources. Setups correspond to quite a different philosophy as compared to the largely experimented economy-of-scale paradigms, whose primary goal is production preservation with the highest output delivery, therefore accepting that hidden capacity (functions and resources redundancy) could be set aside as a spare option.

The innovation, leading to "intelligent" manufacturing, is supported by information technology; it cannot, however, be established without thorough changes of mind for the setting of material processes and the fitting of governing logic. These changes are outcomes to real-time management of production plans and instrumental resources and are enabled through process monitoring, to provide transparent access to every relevant variable, joined to the online availability of pertinent diagnosis frames. Symptom assessment and trend detection are essential wedges of knowledge-intensive setup started by computer integration, to deal with instrumented rigs, endowed by consistent data fusion and signature restitution techniques leading to data properly established according to objective (metrology) standards.

Costs of maintenance are a source of increasing concern for many companies, annually reaching levels up to 3–10% of the equipment cost. Demands will result in changes of work organisation, with upkeeping becoming not less critical than manufacturing itself. The existing organisation now already looks for preventive and predictive maintenance jointly enabled; in fact, the inefficiency of foredoing arises from resort to conservative schedules (as opposed to the equipment-measured needs); prediction based on failure symptoms, on the other hand, hinders reliable continuity, as maintenance is undertaken when misfits are established. Thereafter, the availability of instrumented devices and efficient measurement setups leads to revising the monitoring precepts, aiming at preserving "normal" running conditions conservativeness while prosecuting the life periods as long as conformance to specification is assessed. The expression "monitoring maintenance," indeed, means ability of observing the set of critical process quantities; in fact, monitoring supplies real-time visibility of the ongoing production and provides full records of product quality. Summing up, in view of the process-keeping policies, the following attitudes distinguish:

- *postaudit:* maintenance operates once breakdowns already occurred;
- *preventive:* maintenance makes use of process external variables (overall elapsed time, actual operation intervals, etc.) and to *a priori* reliability data;
- *predictive:* maintenance surveys the process variables, to detect failure symptoms and *then* to enable restoring actions;
- *proactive:* maintenance oversees the behaviour regularity by controlling actions.

With proactive maintenance, it is expected that corrective actions are commissioned for preserving safe running, avoiding the waste of (off-process) preventive schedules and the pitfalls of already established degradation.

Comparison between the maintenance policies is not discussed. Attention is, instead, focused on the innovations brought forth by "intelligent" manufacturing and the related issues of "intelligent" measurements. These lead to economical returns to enable "quality" manufacturing, aiming at producing (with zero-defects) exactly these artifacts that buyers look for, with the highest added value (with respect to competitors). Flexibility, as the first option, supports variable product mixes; as the second one, it has to grant process adjustment to trim products to customer satisfaction; the latter also means that flexibility supplies adaptivity for regulation and restore, aiming at failures risk removal and defective health compensation; maintenance is viewed on a completely new basis, to replace a 'failure reactive,' by a 'soundness

proactive' philosophy, namely, to reinstate the process and preserve 'normal running conditions' by means of control actions which avoid situations leading to machinery faults or degradation.

The present chapter deals with the evolution brought forth by the interlacing of manufacturing and measuring processes, supported by computer intelligence options. "Intelligent" manufacturing or "intelligent" measurements are outcomes, appearing as technology-driven issues. Return on investment is a comparatively more elaborate issue, largely depending on integrating the knowledge frames into standard relational contexts which only provide reliable foundation to engineering activities. The chapter sections develop as follows. An overview on monitoring maintenance is given first, emphasising the innovation aimed at the proactive mode opportunities, with focus on the interlacing of measuring and manufacturing for automatic diagnostics setting. The next section introduces the standard assessment of artifact properties, including the "quality-at-the-large" concept to acknowledge the "fitness-to-purpose" of delivered items, along their expected operation life. The subsequent section deals with traditional tasks in dimensional testing, to explore knowledge integration issues that jointly enable the transparency of a product's quality data and the requirements of process upkeeping. A section is then devoted to developing a total-quality diagnostic frame, with focus on a case application requiring innovative measurement setups in terms of signature detection, feature restitution, scale setting and mapping processes. A final section is concerned with the maintenance of the monitoring setup itself, as soon as the options of "intelligent" instrumentation are available.

5.2 Monitoring Maintenance

A manufacturing facility needs trim, support, restore, and repair actions to maintain the involved resources, in order to preserve the running conditions aiming at total quality of the processed throughput. Earlier mass production was consistent with the preemptive rules of preventive maintenance, based on the *a priori* estimation of the failure rate figures and the pre-setting of operation periods within the MTBF bounds; stops, as well, were established with convenient safety margins, to grant production delivering by resources redundancy. Buyer-driven production is presently based on flexible specialisation and quick response planning; quality and due-date are achieved by adaptive production schedules, generated by integrating control and management, with the options of "intelligent" manufacturing, while effectiveness is enabled through leanness.

The evolution in maintenance organisation, for the above reasons, is becoming a relevant opportunity and is usefully characterised, Fig. 5.1, by distinguishing:

- the off-process setups: by *restoration* rules, when, at breakdown, the resources are turned off from duty-state; or by *preemptive* rules, at fixed timing or given amount of life consumption, provided by (estimated) reliability data
- the on-process setups: by *predictive* rules, as resources are monitored to detect the onset of failure symptoms; or by *proactive* rules, to keep normal conditions by controlling ongoing functional prerequisites

Latter options for monitoring operations require instrumented processes interfaced with knowledge databases aimed at specialised diagnosis setups, respectively, related to:

- Ill-running (situations or trends) monitoring, to prevent breakdown by early detection of the symptoms of the anomalies
- Continuous process monitoring and control of the safe-running situations or trends, to preserve the conformance to specifications of every engaged resource

Intelligent instruments make feasible monitoring maintenance plans, with detection of functional performance signatures, consistently related to operation thresholds and not simply to detection of symptoms related to already established misfits. Upkeep actions are, then, said to aim at proactive mode, as situation and/or trend monitoring provides information to preserve fit-with-norm conditions.

Mode	Assessment	Strategies	Example
POST-AUDIT at breakdown, as case arises	To acknowledge the non-working condition, as the case arises	# Opportunistic re-establishment after failures	The driver is on foot; car needs maintenance
PREVENTIVE according to estimated reliability	To pre-establishe safe running periods based on previous experience	# Periodic recover and/or component replacements	The careful driver maintains the car at regular 10000-km intervals
PREDICTIVE prediction of failure symptoms	To monitor the current behaviour for system failures detection	# Discontinuous restoring at misfits detection	The driver has the car repaired when he finds out downgrades (lower speed, power loss, cooling fluid overheating, etc.)
PROACTIVE controlling 'normal conditions' operations	To check safe-tunning characteristics online and to preserve normal condition	# Upkeep at operation thresholds, by resolving failure root cause	Experience characterises normal condition: small deviations lead the driver to control the car operativity (slightly different noise, higher fuel consumption, harder steering, etc.)

FIGURE 5.1 Maintenance organization modes.

The onset of anomalies is avoided with reliable leads, since trimming or restoring actions are performed as a case arises, possibly during idle or hidden times, by resolving at the root the failure causes. Proactive maintenance concerns include product, process, and measurement quality:

- product quality ('fitness for purposes' or 'conformance to specification') is the basic feature to assure market competitiveness of the artifacts
- process quality means 'zero-defects' production and is the principal requirement to reach manufacturing effectiveness
- measurement quality means transparency, within known uncertainty patterns, of the process variables and the generating influence effects

The extent of view has accordingly to cover the interconnected diagnostic frames and give uniform descriptions of the relational context and unifying methodologies for the quantitative assessment of the observed attributes. The ISO 9001 standards provide the references for tests and controls of products, processes, and measurement devices. The procedural instructions are made available at each range (Fig. 5.2), aiming at:

- Product diagnosis: tolerance and geometric inspections (by dimensional trial and mapping, etc.); functional approvals (by patterns recognition, etc.); structural tryings (by nondestructive tests, etc.); etc.

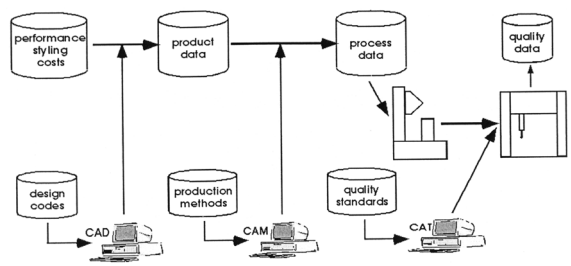


FIGURE 5.2 Diagnostic frames interconnection: product, process, measurement.

- Process diagnosis: exploring resource fitness (tool monitoring, etc.); identifying the causal influences (feedback from product to process, etc.); managing the failure prediction (inference by experimental evidence, etc.); etc.
- Measurement diagnosis, to obtain, for instance: standard uncertainty restitution (enabling automatic gauging procedures, etc.); validation of the instrumental setup (by, e.g., ISO 10012-Part II-driven traceability, etc.); etc.

The build-up of monitoring maintenance by predictive/proactive mode is discussed more in connection with methodological issues than with instrumental aids. Computer technology already offers smart hardware and software; the big hindrance is efficient exploitation of data, by transforming "supervision" into (reactive or proactive) "retro-fit." The promotion of conservative running conditions requires full understanding of the past, present, and future behaviours of the manufacturing facilities and execution of redress, compensation, and repair operations as soon as the execution need is recognised, before degradation symptoms appear. These two steps, to understand for diagnosis and to upkeep for quality, are logically joined, or their separation means such awkward issues as:

- Maintenance without monitoring is like taking drugs with no concern for health data.
- Monitoring without maintenance is like recording illness and looking for miracles.

Performance Prognosis and Health Diagnostics

The basic idea behind monitoring is to measure one or more indicators and assess the actual "health" of the system. Mechanical facilities and machinery have performance that can be classified into different ranges when compared to ideal fit-with-norms. Gradual departures from normal settings can be detected by condition monitoring; then proactive maintenance applies at given thresholds, avoiding the risk of misfits (namely, termination of the equipment's ability to perform its required functions). Lack of appropriate knowledge of fit-with-norms behaviour would require one to measure symptoms of function degradation and to forecast the course of the deterioration; the approach turns out to be reactive, as prognosis is put into practice by misfit occurrences.

Keeping plans, scheduled with online measurements, makes it possible to adapt deeds to actual needs and to ensure that beforehand specified-performance is preserved. In this way, proactive programming is enabled, with benefits such as:

- Transparent equipment life extension, within "fit-with-norms" conditions
- Product quality with highest return in customer satisfaction
- Improved up-time and reduced unplanned outages (maximal plant productivity)
- Less expensive upkeep, with continuous control and minimal waiting time
- Longest "safe-life" horizons with standard health and steady output quality

Proactive maintenance of machinery and plants over their lifecycles is recognised by industry as a subject of growing concern. The option, as properly assessed by the safe-life figure G(t) can deliver large economies to companies, reduce natural resource depletion, and prevent unnecessary wasted efforts in replacing facilities that can, via proper upkeep, continue to perform their required functions for considerably longer periods of time. The switching, from fault detection to health diagnostics (and disease avoidance), requires field-oriented expertise, new measuring equipment, and sophisticated data-processing abilities.

The reliable and accurate assessment of current, fully operative (fit-with-norms) life in machinery is, in fact, made possible by aids such as:

- Intelligent measurement setups (yielding adaptive setting, self-calibration, parameter estimation by genetic algorithms, automatic restitution of data uncertainty, etc.)

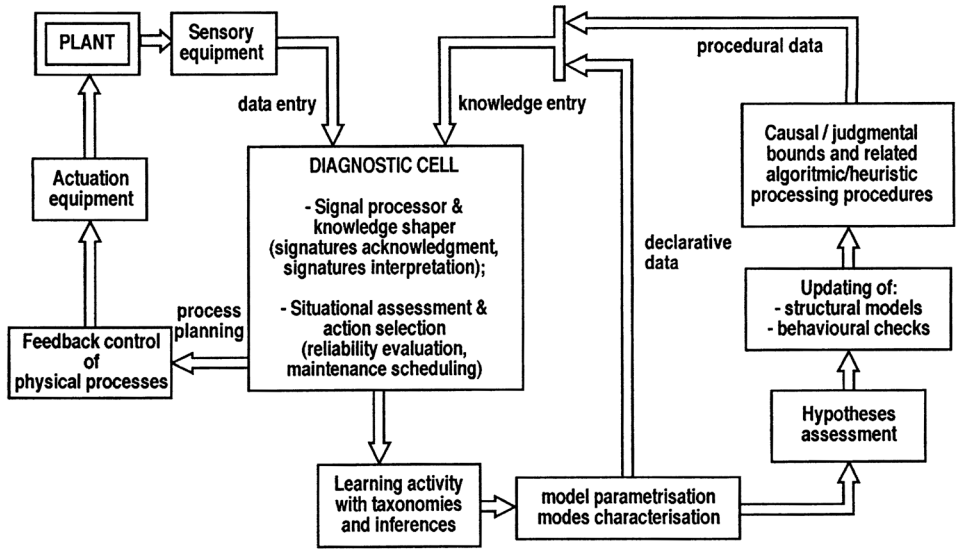


FIGURE 5.3 Automatic diagnosis setups.

- Computer-aided testing modules: ARMA, ARMAX, NARMAX, etc. models; automated feature extraction; data fusion and process identification; etc.
- Advanced signal processing techniques such as time-frequency analysis, evolutionary spectra signatures, wavelet mapping images, etc.
- Adaptive pattern matching schemes, e.g., fuzzy controllers, neural networks, etc.

on condition that the knowledge of the relational contexts characterising machinery and plant behaviour is available and properly exploited due to the user's expertise.

Performance prognosis over the current operation horizons, is, in fact, still the job of trained experts. The listed aids can open the way to automatic assessments provided that standardisation is progressively expanded on the reference knowledge used to build objective diagnoses. Basically, this leads (see Fig. 5.3) to the organisation of the data track (along the measuring chains) and to the explicit setting of a conditional track (along the cognitive mapping restitution chains), with due concern for the selected mode for performing the monitoring operations.

A diagnosis frame benefits of the recent advances in intelligent instrumentation, based on interconnected information flows, crossing the data line of the measurements to establish conditioned prognoses; actually, we distinguish:

- *A priori data*: system hypotheses about the observed quantities, characteristics of the instrumental set-up, estimates on surroundings influences, etc.
- *Observed data*: detected signals and restituted signatures, according to the conditioning measurement chain and/or learning procedure
- *A posteriori data*: situations and/or trends acknowledgement, symptoms classification, etc., with due regard of the *a priori* relational schemes

Industrial diagnostics have been the object of many technical papers; among others, we quote the following: (AGU.97), (BaK.97), (BeB.97), (BiK.97), (Bro.94), (CHA.92), (Ell.97), (IwM.77), (Juu.97), (KhS.97), (KoK.97), (LMR.93), (MCR.96), (MiR.87), (MiR.93), (MMC.96), (MRC.97), (OND.97), (PoT.86), (Rao.97), (Shi.88), (Shi.89b), (ShN.97), (SMN.73), (VoS.97), (YWS.92). The conditioning contexts typically explore continuous or discrete probabilistic models (Gauss-Markov processes or chains);

parametric statistical models (adaptive AR or ARMA weighing filters, with steady failure rate); and opportunistic self-fitting models (neural nets, fuzzy logic identifiers, etc.). Monitoring diagnostic mode distinguishes, as previously pointed out, predictive rules (based on decay patterns) from proactive rules (based on fitness-for-norms control). Symptoms, finally, are detected as spot properties (by situations acknowledgment), or as time drifts (trends by respect to given thresholds), to provide performance measurements and health (disease) estimates. Resorting to standards for the *conditioning* contexts (further to the *processing* contexts) is, possibly, the most intriguing innovation brought forth by *intelligent* measurements.

The Situation/Trend Monitoring of Processes

Now consider the data obtained by process monitoring, distinguishing outputs of processing contexts and issues of conditioning contexts. The soundness or, reciprocally, decay condition of the equipment operativity is analysed with respect to time observations. Prognosis is derived by situation checks (point or source assessments) or by behaviour responses (stimulus or transform tests), grounded on explanatory models, inferred by previous experiments and system hypotheses. Results dependability has systematic validation when data restitution refers to standards.

Failure state and damage progression only occasionally fit in with fully self-consistent causal frames; often, data are collected with redundancy; symptoms are cross-verified as the most plausible issue of the system hypotheses. Industrial diagnostics take advantage of automatic restitution procedures, incorporating learning, inference, judgmental, and validation loops. The knowledge build-up distinguishes a few kinds of activity modes:

- monitoring: signal acquisition, signatures extraction, features detection, etc.
- diagnosis: recognition/prediction of current characterising situations/trends
- keeping: symptoms acknowledgment, regulation, and reintegration planning, etc.
- learning: revision of databases, upgrading of conditioning frames, etc.

For diagnostic purposes, a system is a set of entities and relations, characterised by pertinent knowledge bases and common behavioural modes: the entities correspond to state variables, process degrees-of-freedom, etc.; the relations are contexts, labelled by temporality, likelihood, reasonableness, causality, etc.; the knowledge base is expressed by system hypotheses, empirical know-how, etc.; the current behaviour is characterised by means of actual measurements. The diagnosis paradigm bears trustfulness when identification and learning loops systematically provide objective issues.

A diagnostic frame, Fig. 5.4, covers the following main stages: process monitoring and signatures detection; damage prognosis and keeping plans; maintenance actions and pathology checks (knowledge build-up). Each stage instrumentally profits from information technology. Aspects are reviewed by referring to examples.

Signatures Detection

The monitoring range is directly related to the experimental setup and has to face opposing requests: to collect every relevant information so that the characterisation of the machinery cannot elude proper assessments; to reject useless data so that only pertinent signatures are detected for optimising the monitoring duty.

As an example reference, Fig. 5.5, the diagnostical frame for turbine rotors of power plants is considered [MiR.89]. Bearing monitoring, for example, avails itself of three acquisition chains: synchronous-fast; asynchronous-fast; conditional-slow. Time hierarchy uses “slow” trends (typically, temperature data) as the triggering background and “fast” signatures (typically, vibration data) as evolutionary foreground. The subject has already been tackled by several studies. For example, the modular programme SARM [MiR.90b] makes use of both source and transform signatures. Diagnosis concerns—for proactive maintenance, the control of running set-points to preserve normal conditions; for predictive maintenance, the detection of incipient anomalies as a trigger to start restoring actions.

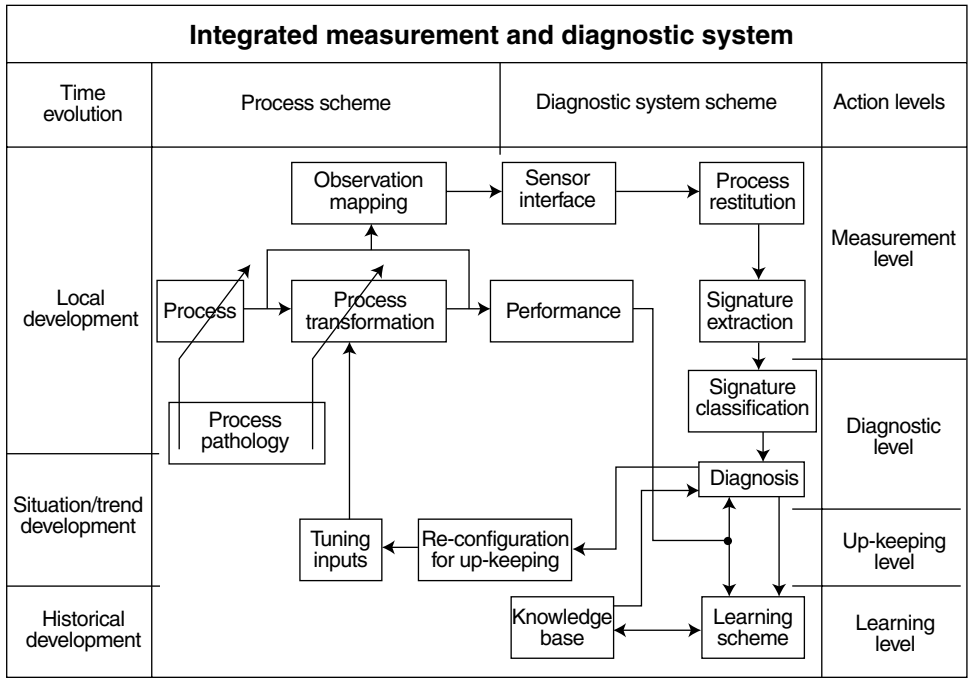


FIGURE 5.4 Block schema of a general diagnosis frame.

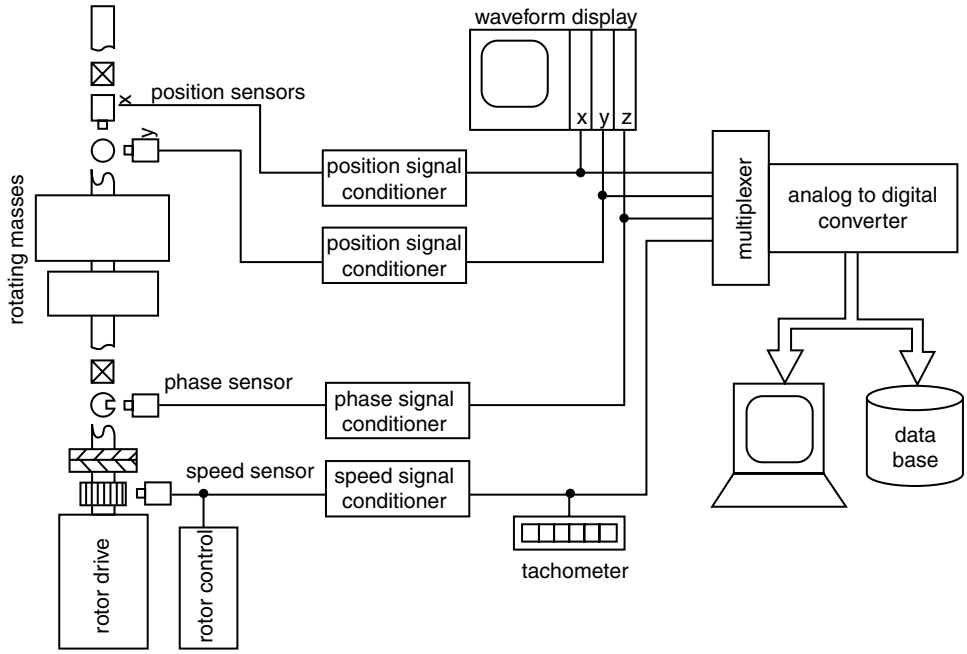


FIGURE 5.5 Block schema of a rotor diagnosis frame.

Upkeeping Planning

Instrumental supervision aims at avoiding the conservative setups of the programmed restoration at previously fixed time intervals, by enabling, instead, upkeeping duties to be accomplished (by proactive mode) to preserve normal running conditions. Situation and trend monitoring are both considered to regulate and reinstate. Risk analysis, with the outcomes for testing, classification, ranging, etc., aims at condition setting to start revisions and/or regenerations with safe margins before any possible damage; to improve the reliability of the setting, trend monitoring helps in estimating, along with error signals, their progression rate so that reparation can be programmed in advance and, possibly, delayed, according to the condition monitoring rules, with due account of the failure risk increase and of the damage onset probability.

Intelligent manufacturing refers to programmed maintenance, as a beforehand guess, for the setting of the tactical horizons, to run optimal schedules on steady production plans; for effectiveness, it switches: to proactive regeneration, while establishing the strategic horizons, with zero-defect schedules; and to predictive restoration, when facing execution horizons and, to any evenience, occurrence of incipient anomalies cannot be removed. All in all, monitoring maintenance, at higher sophistication, is built with the support of knowledge intensive environments; it exploits system hypotheses for feed-forward plans and uses process data for closing proactive control to preserve safe running conditions and for enabling reactive actions at (programmed either unexpected) discontinuities for resetting purposes.

Check and Learning Schemes

Industrial diagnoses, most of the time, make reference to mainly empirical knowledge that needs be tested, widened, specialised, improved, etc. along the life cycle of the investigated machinery; these goals are generally achieved by a learning loop, according to two approaches (Fig. 5.6):

- *Conditions elucidation*: a consistent set of system hypotheses is anticipated and checks are planned to acknowledge when critical issues develop
- *Features assessment*: normal vs. anomalous behaviours are inferred by experimenting and recognising regularity, as far as standard issues are preserved

The first approach can follow causal (structured conditions) or heuristic (judgmental frames) tracks. The second is typically based on self-learning rules, with uncertainty to be compressed concerning both modelling contexts and measurement restitution; this means that need of repair or retrieval actions resort to the detection of changes in the behaviour when these out-span some given threshold. With the former approach, the restoring actions are related to absolute scales, which might be changed only by modifying the mapping schemes (selected according to a representational paradigm).

The expansion of low-cost instrumentation suggests increased monitoring and referencing more and more, to updated information and to discriminating checks. The design of new equipment, accordingly, will modify, aiming at mechatronics solutions, with a relevant role played by the exploitation of data continuously gathered on-process and of prognoses established from experimental evidence. This turns

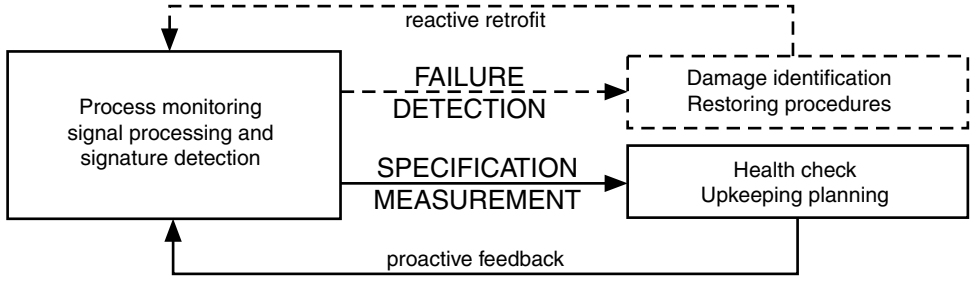


FIGURE 5.6 Knowledge loops and prognoses drawings.

into a changeover of the testing setup, with instrumental checks and learning performed automatically and with results given as standard issues of the process situation/trend monitoring system.

The Relational Frames: Heuristics vs. Causality

By mechatronics, technology opens up adaptive resources, with the ability of adding information as soon as recognised to affect the underlain process. The relational context can be separated into deep-knowledge or shallow-knowledge frames. The former uses structured models and algorithmic procedures; the latter, heuristic models and plausibility manifolds; mixed-mode options are also used, which combine statistical inference and judgmental guess or learning loops and empirical validations. Access to relational frames offers alternatives to draw prognoses: current diagnostics could expand, hoarding data or picking meaningful signatures by acknowledgment cycles, moving with:

- Correspondence analysis: by creating taxonomies and ordering the features
- Consistence analysis: by selecting pertinent bounds and activating procedural methods

The synthesis is performed by clustering, contingency arraying Z proximity mapping, pattern recognition, etc., according to heuristic or causal decision rules.

Diagnoses are stated automatically (with intelligent instrumentation) by comparing detected signatures and (already assessed) symptom patterns; prognoses are issued, via statistical (regression analysis, etc.) or heuristic (expert modules, etc.) inference, through computer loops that condition actual measurements and stored results by means of the hypothesised relational models. Monitoring, by itself, implies oversee actions to detect situations or trends. Recognition of nature or cause of a given observed behaviour and forecasting the processes course with or without some planned controls are further actions requiring selective reasoning and decision supports. The field has fast-moving frontiers, in connection with the evolution of AI tools.

Earlier diagnosis has been related to the ability of assessing origins and reasons of (occurred) failures; causal frames, instanciated by fault trees, have been the main elucidation technique, based on data provided by previous experimentation. Thus, the use of sensor systems appears as a complement to improve reliability assessments and failure occurrences forecast, based on larger amount of in-process measurements. Data redundancy and fuzziness are better tackled by heuristic frames, supported by knowledge based modules, which avoid conflicting issues by proposing plausible choice with regard to some selected relational context. The innovation is brought forth by the ability of processing knowledge (data with conditioning frames) and not merely data (coded information).

Knowledge architectures distinguish (Fig. 5.7):

- A data level, with the declarative knowledge initially procured and continuously updated and widened during the machinery's lifecycle
- A schemata level, with the procedural knowledge provided by field experts and coded, as reference methods, for processing the data
- A govern level, with the decisional aids (inference motor) to select consistent methods once the updated descriptions are acknowledged

Coding at the data level is simplified by object programming. Objects are elemental knowledge, with {*field*, *method*, *(belief)*} for the declarative and procedural information (and the related confidence bound). The instructions remain hidden, unless activated by the govern level; methods, by that way, supply algorithmic or logic sequences, or give rise to analog reasoning, taxonomies, procedural rules, thus, in general, to heuristic correspondences. Computer aids in industrial diagnostics primarily use expert modules, with included inference motors to address execution progress, according to a selected decision architecture.

The domain develops as an application of AI and, for software implements, reference is done to the existing technical literature. Special emphasis should focus on the ability to provide visibility on the uncertainty or,

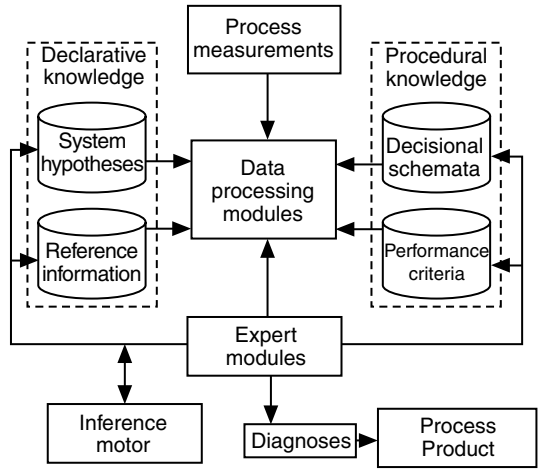


FIGURE 5.7 Knowledge bases (declarative, procedural) and decision aids.

reciprocally, on the belief that currently acquired knowledge bears. Automatic instruments perform explicit restitution of data (nominal measurements) and of data-band (confidence bounds); the fact requires deeper attention because: "cognitive" scales (according to the representational theory of measurements) are defined for standard assessments of quality; and: "intelligent" measurements are enabled for concurrently adapting processing chains and steering decision logic. The technical papers in the field are quickly increasing and examples include: (AlN.91), (BDI.95), (Bar.91), (BrG.97), (Bro.94), (CMR.96), (CMR.97a), (CJM.90), (JLF.97), (LMR.93), (MiR.89), (MiR.90b), (MiR.93), (MiR.94), (Mor.84), (MRC.97), (OND.97), (PhO.95), (PoT.86), (Pul.91), (Rao.97), (RKP.95), (Shi.88), (ShN.97), (SMG.97), (TeB.94), (UOS.92), (ZNF.96).

Maintenance Policies: Preventive, Predictive, Proactive

Upkeeping plans bear practical validation by the economic return of avoiding costs due to facilities failure. Preventive plans are established during the design stages, when reliability data are selected for granting the fitness-for-purposes of each device. The maintenance policy built on preemptive rules aims at the availability, $D(t)$:

$$D(t) = \text{MTBF}/(\text{MTFF} + \text{MTTR}) \tag{5.1}$$

where MTBF is mean time between (forecast) failure; MTFF is mean time to forecast failure; and MTTR is the mean time to repair.

Obviously, a safe estimate of MTFF considers the distribution of empirical results and reduces the operation intervals, MTBF, depending on the acceptable risk level.

On-process monitoring provides updated details; maintenance based on predictive rules can delay the restoring operations by obtaining larger availability spans, $D(t)$:

$$D(t) = \text{MTBF}/(\text{MTTF} + \text{MTTR} + \text{MTCM}) \tag{5.2}$$

where MTBF is the mean time between failure (symptom); MTTF is the mean time to failure; MTTR is the mean time to repair; and MTCM is the mean time to condition maintenance.

Condition monitoring maintenance does not uniquely depend on the *a priori* information (used by totally preemptive rules) and computes MTBF and MTTF figures with updated information obtained all along the device work-life; (preventive) maintenance is thus delayed and performed only when useful,

namely, when the failure rate starts having an upward trend. By restoring, the failure rate figure is made to recover the original steady value. With life-cycle design, knowledge-intensive setups are devised from the artifact development stage and monitoring maintenance concepts are becoming standard option in engineers' practice. This leads to modification of the instrumental setup, aimed at detecting the deviations from the normal running conditions and at using the data for online reintegration each time some given threshold is overrun and the facility output fails to match the conformance-to-specification requests.

The approach refers to the quantity MTTT, mean time to threshold. If the artifact is consistently designed, with the proper knowledge frame, monitoring provides the link toward a proactive (based on fit-for-norms condition), from predictive (based on reactive mending) setting. The quantity MTBT, mean time between threshold, is acknowledged to plan upkeeping operations, which will preserve the machine's functional capabilities (and the expected output quality); the resulting safe life-span figure, $G(t)$, depending on diagnostical on-process data, is reckoned as:

$$G(t) = \text{MTBT}/(\text{MTTT} + \text{MTTU} + \text{MTAM}) \tag{5.3}$$

where MTBT is the mean time between threshold; MTTT is the mean time to threshold; MTTU is the mean time to upkeeping; and MTAM is the mean time to pro-active maintenance.

The quantity MTTT is used to state the artifact intrinsic or bestowed properties; MTTU depends on restoring capabilities; MTAM expresses the process-driven modulation of the maintaining actions. The estimation of MTTT is performed by selecting prudent thresholds with respect to the current MTTF figures. The safe life span, $G(t)$, is evaluated, referring to the data provided by properly designed diagnostic tools, as a function of a cautious estimation of MTBT, instead of the previously considered MTBF. The change from the availability $D(t)$ to the safe life span $G(t)$ concept is, possibly, an obvious suggestion. It cannot be obtained unless the artifact's life-cycle properties are actually described by a well-acknowledged model and are kept under control.

Monitoring Architectures: Diagnoses Organization

Industries expect effective support by monitoring maintenance resources setups. To that aim, a functional model, combining manufacturing and measuring actions, should be detailed (Fig. 5.8) with specification of the involved data flows. On the manufacturing side, the main input concerns the product/process characterisation and the main output deals with the product features and process set-points; auxiliary data flows are required, at the input, to specify tolerances, supplies' data, procedures, methods, etc.; and at the output, to obtain productivity, performance, returns, etc. On the measuring side, the main input is represented by the measurements (situation and trend signatures) of the monitoring equipment, and the

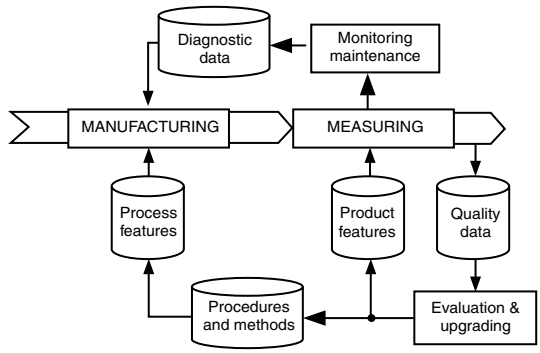


FIGURE 5.8 Concept schema of monitoring maintenance setups.

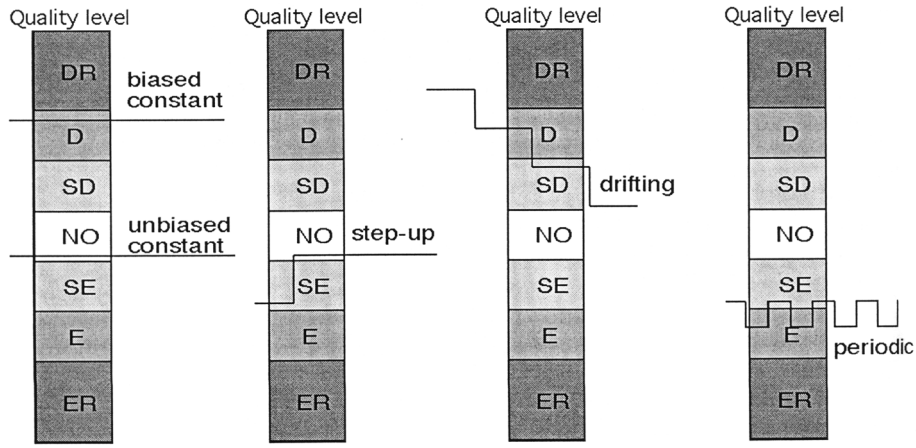


FIGURE 5.9 Reference courses of process monitoring time behaviour.

main output provides the diagnostic information (with elucidation of the originating causes); auxiliary inputs are the system hypotheses, the setting of the process-to-products relations, etc.; and auxiliary outputs cover the suggestions for improvement, the records to prove the delivered quality, etc.

The processes are conveniently referred to as "jobs"; data are acquired at the end of each job. For our goal, quality data are the representation of product features in a (standard) format, capable of being compared to product specification (and to fitness for use). Conventional quality data are given in Boolean (go/no go) or in (finite-length) numerical formats. Diagnostic frames need to perform cross-linked product-and-process measurements; proper information is generally provided by coarse quantised data, with resolution (over-all uncertainty) corresponding to the (process) value band. Thus:

- The product-approved tolerated interval is divided into m bands, e.g., $m = 5$; for example:
  - A (unbiased) nominal reference band (NO)
  - Approved slightly defect (SD), defect (D), etc. bands
  - Approved slightly excess (SE), excess (E), etc. bands
- The product rejected features could similarly be ranked by bands; for example:
  - Small (large) defect rejected (DR) and excess rejected (ER) bands
  - Overflow (defect or excess) values, out of the instrument spans

Monitoring yields sequences of discrete values. The time evolution considers results over fixed observation windows and the following typical courses can be distinguished (Fig. 5.9):

- Unbiased constant values: steady course within the nominal reference band
- Biased constant values: course with data (partially or totally) in the off-set bands
- Step-up (step-down) values: constant courses with a single abrupt up- (down-) shift
- Repeated step-up (step-down) values: drifting courses with upward (downward) trend
- Periodic step values: cyclic courses with periodic up- and downshifts
- (Steady average) scattered values: random manufacturing quality

The technical literature involving diagnostics development, as already pointed out, is increasing very fast. Typical references include: (AlP.79), (BDI.95), (BDL.97), (BLS.97), (Cem.97), (ChC.97), (FoS.91), (IMR.93), (JiD.90), (JVR.90), (KJE.91), (LaD.84), (MCB.90), (MiC.85), (MiR.90b), (MiR.94), (Mor.84), (PhO.95), (Pul.91), (RuT.82), (Shi.89a), (SeM.90), (SMG.97), (SoB.97), (YuH.88).

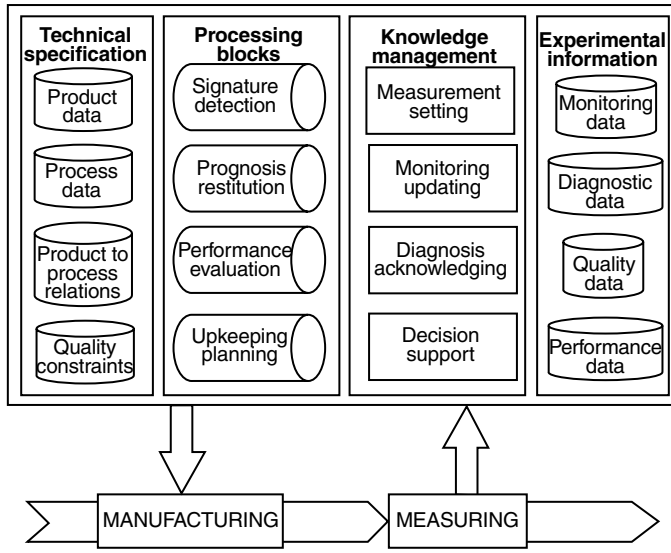


FIGURE 5.10 Block schema of monitoring maintenance equipment.

Learning and knowledge accumulation are working means of monitoring setups for artifacts quality assurance and process upkeeping control. As for diagnoses, it is useful to define (Fig. 5.10) a set of specialised databases, a set of knowledge-based modules, and a set of data processing blocks; namely:

- Databases: these are permanent or upgraded storages comprising system hypotheses, measurements, and updated prognoses; for example:
  - Technical specification, describing product, process, and (currently updated) bounds between the two
  - Experimental information, including (raw) monitoring data, (product) quality data, and (process) diagnosis data
- Expert modules: these supply decision aids for setting/resetting the measuring and manufacturing processes; for example:
  - Monitoring rig shaping, to select measuring chains, signature detection, and quality data restitution schemes, pertinent quality indices, etc.
  - Diagnostic rig shaping, to work out quality problem cause-finding, process ability decision, fit-with-norms upkeeping suggestions, etc.
- Processing blocks: these perform acquisition, analysis, abstraction, selection, etc. tasks on the current memory data; for example:
  - To determine product features, for quality approval checks as compared to technical specification and for preparing tendency forecasts
  - To recognise product/process bonds, to explain issued quality problems, and to draw conclusions on the originating causes
  - To assess process restore/upkeep actions, for selecting upgrade/control functions to preserve fit-with-norm conditions, or to alleviate misfits

The control rig operativity follows from acknowledging current quality requirements (experimental data vs. product/process specification) and by enabling feedback actions consistent with the desired maintenance policy. The proactive mode refers to (situation or trend) thresholds, employed for quality management during manufacturing, in view of finding influences and suggesting remedies from the information loop:

- Understanding the problem: to select the signatures, to look at data courses, to stratify and compare data, to classify issues and forecasts leading to fit-to-norm standards
- Analysing the causes: to determine the involved factors, to choose the conditioning relational frames, to verify the consistency of the chosen diagnostics
- Enabling the improvement actions: to evaluate upkeeping policy effectiveness, to examine the upgrading issues of process parameters remodulation

The information loop requires fairly sophisticated knowledge frames to associate product features, process parameters, and improvement actions. Typical opportunities include investigating multiple pattern issues by cross-examination of redundant quality data to draw interpretations and verify their soundness; singling out cause/effect structures, by iterating the information loops with alternative monitoring/diagnostic rigs; performing instrumentation up-keeping, by including measurement uncertainty assessments for upgraded monitoring quality, etc. The list reveals the inter-related nature of manufacturing and measuring each time monitoring maintenance is accounted for. It also raises questions of the possibility of collecting sufficient data and of building consistent knowledge to grant proactive keeping (or predictive restoring) feedbacks. Answers largely depend on the availability of intelligent instruments, as soon as the area experts are in the position to offer the knowledge frame of a sufficient diagnostics.

5.3 Quality Manufacturing

The durables industry faces the typical situations of affluent societies, wherein several potential buyers exist in a market that is approaching saturation by the offers of equivalent competitors. An enterprise can preserve or improve its marginal share ratio of worldwide trade, provided the "quality" figure of the product mixes is suitably distributed to win satisfaction of increasing sets of purchasers. Goods become increasingly sophisticated, grouping many sub-items that combine several technologies. Quality manufacturing means "customer satisfaction" (technically assessed *fitness for purposes*) and, for efficiency, "zero-defects" (*conformance to specifications* of all artifacts) needs be preserved. This requires recurrent collation with consumers and continual updating and upgrading of offers. Integrated design is thus becoming the extra knack, to customise goods with properties enhancing desirability with respect to competitors. The issues are properly recognised and a large literature exists in the field; for example: (AlP.79), (BDI.95), (CIM.97), (CIS.97), (DrC.95), (KuM.93), (Mic.92a), (Mic.92b), (RIL.92), (RuT.82), (SuM.85), (SwM.95), (TDR.97), (VoW.97), (WSH.92), (ZNF.96).

Buyer education and fair-trade rules impose crisp assessments of artifact properties; quality engineering must be a standard accomplishment, based on approval tests, referring to objective scales. Manufacturers must join flexible specialisation and productive break-up; effectiveness is given by the ability of sharing competencies and combining sub-supplies to reach diversified quality by integrated design and incorporation of special components and technologies, most of the time purchased by external suppliers. An enterprise can expect to succeed in expanding its trade position provided that the *quality figures* of the product mixes are distributed to meet customer demands. "Quality" is a complex attribute, which should be present throughout the life cycle, up to artifact replacement and environmental protection rules. Its measurement often exploits *engineer* scales and *conventionally* standardised restitution procedures. The nondestructive tests are, performed by *certified* personnel to assess the structural properties of mechanical components. The concept can usefully be extended to measure the "quality-at-the-large" figures, having explicit interest in granting objective protection to suppliers and consumers according to fair-trade rules.

The "quality-at-the-large" goal is leading to design accomplishments, with falls off on artifact life-long properties, such as safety, reliability, performance, energy saving, anti-pollution, or other technicalities liable to improve the quality of a given offer, beside potential buyers. Some of the properties may be fixed by registered authorities with compulsory rules or prior registration precepts. Others are established to improve ergonomic operativity or the friendliness feeling of the running conditions, and, more generally,

	checking for quality	vouching for quality	
functional approval	testability	assurance	conform to specs.
life-cycle operativity	reliability	certification	fitness for purposes
	metrology references	legal standards	

FIGURE 5.11 The patterns for the artifact quality acknowledgment.

to ameliorate the comfort as users benefit from the option. Design-for-comfort belongs to the field of adding user satisfaction outfits, which can be attached to a device, to distinguish one offer from others. The betterments are sometimes minor gadgets, sometimes useful contrivances, or real improvements. As a result, the appeal of an artifact depends on several details, for example: proper styling as visual pattern or sound emission; functional performance with reliable display of the actual running state; usage friendliness with transmission of a safety feeling; etc. The extra features have a value for buyers and a cost for manufacturers; both obtain a benefit when the price is equally spaced between value and cost. The statement, on one side, bears engineering relevance, whether specified by technology-driven assessments. On the other hand, the level of friendliness feeling or of aesthetic character to be transmitted to users is quite subjective; design-for-comfort is, for instance, a critical business, too frequently tackled with carelessness and leading to hardly predictable issues. As long as the characterising features remain subjective, fair trade rules are not established at least according to quality measurement precepts.

Quality Measurements: Cognitive Scales

The above considerations show the evolution from the earlier business-driven ideas, such as CWQC (companywide quality control), to the new goals in quality engineering, which grant objective tests, for buyers and sellers protection, ruled by legal metrology. The quality will aim at fitness-for-purpose (according to user's needs) with reference to the product expected life cycle; its assessment (Fig. 5.11) has to be established by objective standards, possibly, with technical certification issued by proper accreditation schemes. The consciousness of the drawbacks of unsuitably stated quality scales is still lagging; several manufacturers are mainly concerned with managing quality as a business-internal option to improve competitiveness and do not realise that the constraints for a compatible development will increase in number and cannot evade the objectivity of technology-driven specifications.

The equivalence "quality = value" is theorised by (free-market) econometric models, presuming highly qualified manufacturing enterprises, highly aggressive consumer organisations, and highly effective accreditation bodies. The resulting environment could become highly conflicting, with unfairly poised opportunities among partners, unless the quality of *details* bears reference standards. The metrologically consistent setup is, possibly, established by the representational theory of measurements (Mic.92a), (Nar.85), (Rob.79) to define proper scales and measuring methods and by extending the use of such indices as a legal bound of any commercial dispute.

The selection and the standardisation of quality indices are far from being generally accepted. Perhaps this development of legal metrology is still facing the hindrances first met by Napoleon's committees, when they were trying to fix, all over Europe, the scales of the basic physical quantities. Therefore, we will move with sample presentations that do not presume strict standard traceability of the measured entities, rather than the objective repeatability of the tests. The approach progresses along a simple track:

- The "detail," reporting the quality-at-large concept to be evaluated, is recognised
- The related "signature" is defined by a mapping law and proper instruments/methods
- The restitution scale is verified to systematically repeat with reliable uncertainty
- The representation soundness is verified and the conditioning context specified

The representational theory of measurement (Fig. 5.12) (MCR.96) supplies simple criteria and rules for defining metrologically consistent scales. Quality, or more precisely, any specific detail, is made to correspond to the set of functional and aesthetic features, with technical (e.g., reliability) or emotional (e.g., ergonomy) relevance, which characterise the offered item in connection to customer's satisfaction.

According to the representation paradigm, the measurement is "*the objective assignment of order-symbols (numbers) to attribute objects or events, in such a way as to describe relations between them (by means of the metrological scale, provided together with the selected mapping)*".
By representation, totally ordered relational structures, $\langle X,.,R_1,R_2,\ldots\rangle$, are associated with appropriate types of scales (ordinal, interval, ratio, absolute, etc.) by means of well-formed formulas.
For engineering applications, the coding formalisms deserve consistency checks, to specify dimensional (reference units) and statistical (influence uncertainty) restitution.
The extensive entities have positive concatenation structures, with partition-based additivity; the basic homomorphism maps $\langle X,.,R_0\rangle$ into $\langle X,.,Re^+,\leq,+\rangle$; the set of the additive representations is a ratio scale and the results are expressed in cardinal numbers. These are the coding schemata for quantity and are acknowledged by the accounting operation, used for direct measurements (partition into replication of unit elements). Negative numbers require context interpretation (with, e. g., the meaning of 'not-possessed' vs. 'in-house' quantities).
The intensive entities have straightforward representation; the relational context is monotonic, with sequence scales corresponding to ordinal numbers. These, thereafter, are the coding schemata for quality and introduce the abstract scales, for the extended standards of the representational theory.

FIGURE 5.12 Prerequisites of measurement representational scales.

It may become necessary to rank these features by means of synthesised *cognitive* references, properly selected to simplify the issuing of standard rating by evaluation teams (through patterned assessment tests). In the future, in fact, evolution of legal metrology, from the unified scales of physical quantities (to weigh, e.g., mass) to the replicated measuring rigs and methods (to assess, e.g., material strength), will lead, by that way, to standardised sets of measurands with *cognitive* scales used as reference (to rank, e.g., comfort feeling). Representational scales will be recognised by means of empirical assessments (e.g., by patterned interview methods) and not, necessarily, by means of system hypotheses providing fully structured causal contexts. The *details* of artifacts measured according to (properly acknowledged or only hypothesised) non-destructive tests, for instance, are examples of technically sound, "quality-at-the-large" indices. The sample metrological frame built on vibro-acoustical images, as well, belongs to such a domain; it provides explanatory hints of offspring in quality manufacturing, when established into objective frames.

Representation paradigms meet increasing interest for technical fields, particularly in quality engineering, to develop accreditation schemes based on conventional scales that could not be stated with relational frames inferred from existing physical laws. By the representational approach, in fact, the measurement of ordered conjoint structures (entity X and predicate P), is performed by the abstract factorisation of the ordering rule into the constituents' orderings. The decomposition follows from conditions on the relational independence, which allow the scales to operate separately on the empirical structure X and on the (single or replicated) predicate structure P. The connectivity (or similar assumption) of partial orderings is not strictly necessary. The generality of the results granted by this approach makes the coding formalism quite powerful. The extension to cover uncertainty has been investigated more recently. It explores the ordering by weak transitivity with value-band restitution: the coding abstraction remains unaltered, allowing the reference aid to deal with experimental data. In any case, the approach brings forth the idea that every entity (physical or cognitive) can be mapped into a numerical scale (Fig. 5.12): sizes (extensive entities) have partition rules for generating magnitude symbols (cardinal numbers); ranks (intensive entities) have ordering rules for generating range symbols (ordinal numbers).

Quality Metrology and Abstract Standardisation Rules

The representational theory further supports unifying views to deal with (ordered) conjoint structures, possibly having predicate sub-structures related by connective and instanciation operators whose elucidation is afforded by semi-groups (weak transitivity formulations, qualitative reasonings, etc.). This helps singling out numerical structures and uncertainty assessments by a constructive approach (Fig. 5.13). Uncertainty assessments are similarly stated by well-formed formulas (even if, at the moment, little investigation is available): qualitative (ordinal) scales and quantitative (cardinal) scales, indeed, are

Definition: the representation scales are order mapping of conjoint structures: $\langle \mathbf{X} \times \mathbf{P} \geq \rangle$:

- Obtained by an abstract factorisation rule (from $\geq$, into: entity $\geq_X$ and predicate $\geq_P$)
- Leading to numerical structures with strong transitivity

For practical purposes: the metrological scale is a conventional structure formed by:

(a) An empirical relational system (namely, the entity under observation), joint to
(b) An associated numerical relational system (namely, the metrics), and together forming
(c) An homomorphism (namely, the measurement, within the selected representational paradigm).

Operatively, a measurement paradigm is thus based on the assumption of being able to prove:
The mapping meaningfulness: the given conditions on the empirical systems are sufficient for the existence of an objective assignment of symbols (numerical systems);
The system testability: the above said conditions shall be checked experimentally, in order to assure the trustfulness of the measurement methods;
The scale construability: the representation should have a constructive proof, which provides a method for the measurements, by means of direct theoretical background (physical metrology) or by means of a universal testing instrument (technical metrology).
The constructive approach provides well-formed formulas and also describes uncertainty.
Uncertainty is assigned by a nominal value with a joint set (probability or membership function, etc.):
– The 'empirical into the numerical structure' homomorphism is given by weak transitivity formulations;
– The measurement is expressed by a meaningful realisation and its associated uncertainty (value band).

FIGURE 5.13 Constructive approach of representational scales.

The **learning**, by relational schemata based on first-level mathematical logic patterns, uses only:
• Elemental symbols: constants, variables, predicates, identity ($=$), order ($<$, $>$), semi-order $\leq$, $\geq$), separation ($\bullet$ & $\bullet$), connectives ($\neg$ not; $\wedge$ and), instanciators ($\wedge$ *for all*; $\exists$ there exist);
• Derived logical connectives: $\vee$ *or*: $[(A\ B) = \neg(\neg A \wedge \neg B)]$; $\rightarrow$ if ... *then*: $[\neg A \vee B]$; $\leftrightarrow$ if and only if: $[(A \rightarrow B) \wedge (B \rightarrow A)]$.
The build-up of logic images by first-order patterns grants syntactic consistency; the logic images are formal structures for which any coded expression is directly interpretable. First-order representations are akin to automata style: coding of a variety of structured modes leads to ordinal or to concatenation patterns, with joint and branching connections stated without ambiguity.
To preserve the representation with well-formed formulas (that can be directly translated into actions), the structural complexity of first-level relational frameworks, unhappily, grows to unrealistic levels. The interpretation efficiency requires different formalisms, based on conceptual models, whose validation is mainly based on semantic and on pragmatic verifications.
The **learning**, by schemata based on higher-level logic, uses (causal) procedures:
The empirical structures are derived from conceptual models (algorithmic or heuristic relational frames);
The standards are derived by patterning the decisional frames.

FIGURE 5.14 Patterned learning by first- or higher-level logics.

actually given by finite format (quantised) figures, with resolution-repeatability indices expressed by value bands; interval computation and set algebra provide basic aids to propagate the detected uncertainty; and the use of multiple-coding variables merges crisp and fuzzy data into unified forms, making sense to scale with semi-order representation. In this way, the analysis of uncertainty can start, aiming at modelling the relational binds that properly fit the empirical (observed) occurrences. Abstractly, the confrontation of diagnostics against structural uncertainty yields specialised plans aiming at: monitoring, surveillance, prognoses, upkeep, govern, manage, etc., with steps to assess consistent descriptions and steps to perform progressive learning.

Intelligent measurements have already been characterised by their ability to deal with standard conditioning contexts and not only with standard processing contexts. Knowledge propagation by purely syntactic patterns (according to mathematical logic) is, however, feasible for elemental goals only (Fig. 5.14);

the problems of engineers' interest, up to now, have profited by "hard" framing and solutions have been obtained by exploiting conceptual models. This "hard" framing characterises lots of engineers' scales, based on the standardisation of the reference instrumental setup. Well-known examples include hardness scales (Vickers, Brinell, etc.), noise pollution (according to shaping wave-band filters, etc.), etc. The evolution towards "intelligent" instrumentation makes it possible to standardise both the data processing blocks and the decision judgmental modes. It follows that learning can progress by patterned rules, also with high-level logics, provided the "hard" framing of conceptual model selection is transparently fixed. Sample comments are discussed in connection with the standardisation of acoustic signatures.

Signatures extracted from noise or sound emission have already been exploited for diagnosis (RKP.95), (JiD.90), (IwM.77), (detection of ill-operating conditions, etc.) or for specifying safe running levels (SMG.97), (YWS.92) (noise pollution, inside given 'limits'). The acoustic emission of individual artifacts (engines, doors, dash panel triggers, etc.) can, moreover, be turned into a distinguishing feature and also used as an advertising message (KJE.91). This more subtle goal presumes to model sound generation phenomena, in order to oversee the release of quality devices. The quantitative description might further be referred to by integrated design, aimed at the development of new (market-driven) devices, endowed by given sound images, objectively assessed by ranking the comfort feeling levels (or other quality index).

The appraisal of acoustic images associated with artifacts, to resorts, etc. bestows large relevance to psychological factors. Judgments can be pursued by the patterned interview method, with purport depending on the ability of preserving constant environmental conditions during the evaluation trials. For such goals, the jury databases need to be built completely with preregistered signals and ordered using sound synthesisers based on standard methods to generate the reference set of synthetic sound images with know accuracy. Standardization by patterning the decision logic (Fig. 5.14) indeed becomes a nondisposable request; the lack of objectivity jeopardises fair competition of manufacturers and proper protection of consumers.

The storing of sets of measured signals is a useful means to classify, analyse, and back-generate families of acoustic images with standard emission features. The shaping of the sound patterns of actual devices (e.g., door closure, safety locking, etc.), to yield reference sets of acoustic images, is a considerably more complex task. The procedure is actually undertaken by a series of constructive steps, such as:

- Phenomena classification to define rank orders and rating scales, possibly based on psychological motivations
- Behaviour modelling, with causal and heuristic frames to condition both device dynamics and actuation patterns
- Parameter setting of the relational representations to provide reference means for developing devices with the proper sound image

In the first step, the sound images are assessed by objective scales, provided that tests are performed with metrological consistency. The second step leads to proper functional models for the equipment's transform/behaviour, making it easy to predict, by computer simulation, actual command sequences and time evolution. These models typically include causal blocks (to duplicate the structural dynamics) and logic blocks (to emulate the decision patterns). Then, the third step exploits the fitted-out models by modifying the parametric setting to investigate differences in acoustic emission obtained by reshaping the equipment.

The measurement of synthetic indices to assess artifact quality will enter into practice as soon as metrology standards are issued and accepted with proper traceability of both the processing (transforms applied along the data flows) and the conditioning contexts (decision supports provided through the control flows) when *cognitive* scales are used. The ideas are expounded upon in the following, by means of an explanatory example, for the case application of the acoustic signatures, considering the sound image of a minor component for car fitting. At the moment, most of the time, sets of conventional measurements are simultaneously carried on to give a description of the artifacts' properties with the clear assessment of the individual details, even if the approach makes it difficult to graduate the quality as "fitness-for-purpose" or satisfaction of a given customer. The next section is given over to reviewing aspects of the process integration of the measurement equipment for the effective exploitation of quality data, issued by standard scales, as arises with the

exploitation of the modern CMM. Subsequent paragraphs in this chapter section are concerned with introducing concepts on the use of synthetic indices to measure comfort properties by means of (mainly) conventional scales. These concepts are starting ideas for the developments suggested in Section 5.5 where the use of non-standard signal-mapping procedures is examined.

Quality Approval Trials: An Example of Development

Quality inspection by synthetic index is an increasingly important opportunity. An example situation is considered for explanatory purposes. Panel instruments of every cockpits or other work—or even, leisure—front seats are triggered by knobs or levers, sometimes with multiple key positions to enable several actions. For most cars, for instance, the push-buttons and swing pullers are distributed on the dash-board and should be accessed without looking at them so as not to divert the driver's attention from road. Sound is a relevant means to provide details on the action accomplishment, in order to:

- Distinguish the addressed device from the originating signature
- Understand the final situation (activated/disabled) of the command
- Recognise the safety and trustfulness ability of the dashboard
- Acknowledge the "class" shared by vehicle, cabin, panel, etc.
- Extract practical data by inborn/inurement reflex

Self-confidence and reliability are indeed important. Self-discriminating abilities and high-standing marks are similarly essential preconditions to weigh outfit soundness. From the propagated sound, the driver obtains the feeling not only of the effectiveness of a chosen device, but also the influentiality and friendfulness of the purchased artifacts as a whole. These kind of items, moreover, are usually bought from (several) purveyors; technical specifications thus need be turned over to the suppliers, with a clear indication of properties and quality ranges that will be possessed, on duty, during running conditions. The developments and findings concerning engineering tasks need to enter the way of quality metrics; the final setting of a new car, accordingly, will go through an expanding number of measurements, with attention on the quality-at-the-large options because users wish, further to failures removal, the feeling that every device has fully healthy assets, provided by readily understandable messages.

Refer to the acoustic image related to the sliding actuator of a ceiling-light fixture (CGM.97), (Fig. 5.15), placed within reach of the driver's hand. It typically corresponds to a three-position switch: *on*; *off*; *on*, *with open doors*. Its characterisation presumes a set of technical specifications, related to:

- Driving force: a standard lever actuation law is assumed (progressive effort by a single finger, centrally applied to the lever)

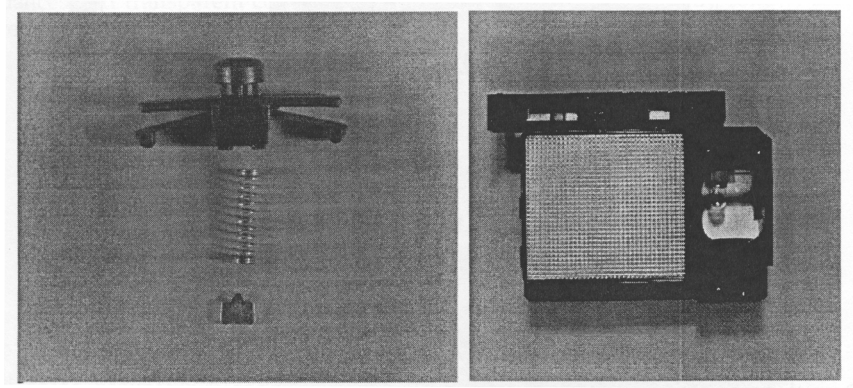


FIGURE 5.15 A sliding actuator of car ceiling-light devices.

- Interconnection between device and ceiling: a direct packing or a gasket linking modify the fixture behaviour
- Interaction with the surrounding framing: the nonlinear cross-couplings are deferred as second-order effect analysis

In keeping with the existing practice, the driving acoustic patterns are explored by means of spectral measurements joined to analysis of the noise sources. Foreground specifications concern time signatures and harmonic spectra, with background data on time-frequency distributions to correlate the frequency components during the actuation cycle and to avoid irksome peaks. The analysis has a threefold purposs:

- To measure the acoustic emission by means of proper representational mapping
- To obtain the reference features for synthesising equivalent sound images
- To certify the quality of the restituted attribute by consistent metrology patterns

Once the testing conditions are defined, the acoustic image is measured according to proper time-frequency standards. These sound images are pulsed, nonpersistent waves. Usual spectral analysis, based, on time-varying windows, leads to "tonal" features only. But agreeableness also depends on "timbric" features (i.e., on emission dithering trends) and time-frequency analysis should be based on generalised spectra (e.g., obtained by Gabor transforms) or on wavelet decompositions (with proper functional bases, for duly accurate analysis and synthesis). Alternatively, the standardised sets of measurands, used as a reference for patterned assessment tests, can be expanded to cover the practical range of all traded devices (to make consistent comparisons). This second option is, perhaps, cumbersome, but still preferred today because generalised spectral analysis does not currently possess mapping standards and unified scales and is, therefore, avoided unless it is necessary to compress the relevant data into very synthetic signatures, as in the case developed in Section 5 of this chapter.

This sample study deals with usual frequency spectra, measured according to standard phonometry (Fig. 5.16) by means of the conventional B-scale of noise metrology. The collection of the combined information—*sound emission waveforms + related frequency signatures*—is performed to provide an extended collection of data to be proposed as jury reference. A more effective representation scheme (e.g., wavelet-based) could be considered in parallel, to assess "best" data compression into "synthetic" sound images. Once the physical meaningfulness of generalised spectra is accepted (and used), these more

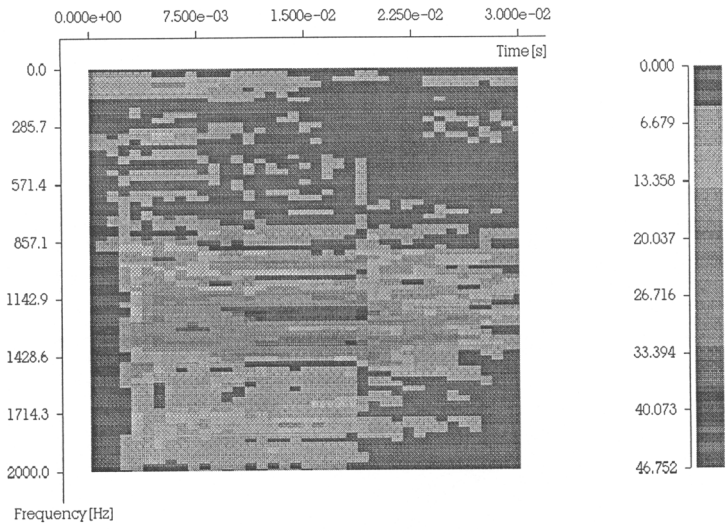


FIGURE 5.16 Acoustic images of the ceiling-light device.

effective mappings can be used as process standard replications, providing a means to establish a consistent scale for the collective ranking of the traded devices (according to details further discussed in the following).

The example discussion has introduced a situation that enterprises will in the future face more and more, namely: the capability of improving the 'quality' of a given artifact in connections with very subtle prerequisites, that, nether-the-less, are critical to keep the competitiveness of the business. The "quality-at-the-large" indices need be reached at the design stage and the analysis requires objective assessments, combining detailed modelling of the operation behaviour of the devices, with measurement of the relevant influence variables by means of objective observation schemes. By hazy data, muddled ideas follow and confused developments are addressed. Conventionally, today noise metrology is already well stated. Instruments, however, involve reference restrictions for detection, processing and restitution. Sensing devices are, typically, followed by limited resolution A/D converters and data buffering abilities. For signals processing, programmable units are good opportunity, aiming at adaptive filtering while storing, as long as useful, observed data, intermediate results and final signatures; signals are provided as ordered (quantised) data-blocks; standard algorithmic or logic modules can be addressed. This means that the digital instruments makes easy to compute features, obtained as non-linear outputs of multiple-inputs systems, by acknowledged processing sequences and transparent conditioning logics; next steps will certainly follow, based on more effective data-processing techniques and standardised decision patterns.

Quality Indices and Design-for-Comfort

The design-for-comfort process is conditioned by the actual performance of the considered device (e.g., the ceiling-light fixture) in operation (as car outfit). Requirements should be turned over to the suppliers of the components, with proper indication of the quality ranges to be certified. On the other hand, habitableness and friendliness feeling are quite subjective attributes. To be turned by the manufacturer into an advertising message transmitted to users, the acoustic image of any devices (from engine, to key-board push buttons) has to become distinctive feature of a particular car and any new artifact should be designed to share the distinguishing patterns. This fact suggests that design-for-comfort paradigms could evolve to higher sophistication, with concept and approval stages intimately bounded to assess and to verify each phase of the design cycle. The development and testing should fulfill a twofold duty:

- To obtain quality patterns according to buyer requirements, with efficient control of all production properties
- To improve quality patterns, in terms of averaged figures and variances, to win upper-class consumers

The case study, at the first step, is based on standard spectral measurements, in conjunction with detailed analysis of the noise sources, to provide objective assessments of the given equipment technical appropriateness: foreground specifications concern the total emission with back conditions on the spectral distribution to avoid irksome peaks. A considerably more complex goal is represented in the second step by the shaping of the acoustic emission of the particular equipment (push-buttons, door closures, safety lockings, etc.), to be used for providing hints on how to redesign it and to reach the wanted betterments. The spectral analysis must then be generalised or replaced by more specialised data restitution techniques; the design-for-comfort approach, accordingly, needs evolve, exploiting parametrised models and investigating how structural or operational changes could be addressed to modulate the details of the device to reach the requests.

Reconsider the case of car ceiling-light fixtures. The analysis is performed with the parametric CAD Pro/Engineering code, to model (Fig. 5.17) the sliding pin and related guides. Pro/Mechanics aids in performing kinematical animation and structural analysis, to obtain, for instance (Fig. 5.18), the vibration modes up to fourth order. Solid modelling of the sliding pin is fundamental to fix the modes; surface shaping and finishing at finger coupling is almost as relevant to fix the effort gradient and to address the driving actuation law according to proper patterns. The situation is further modified by the profile given

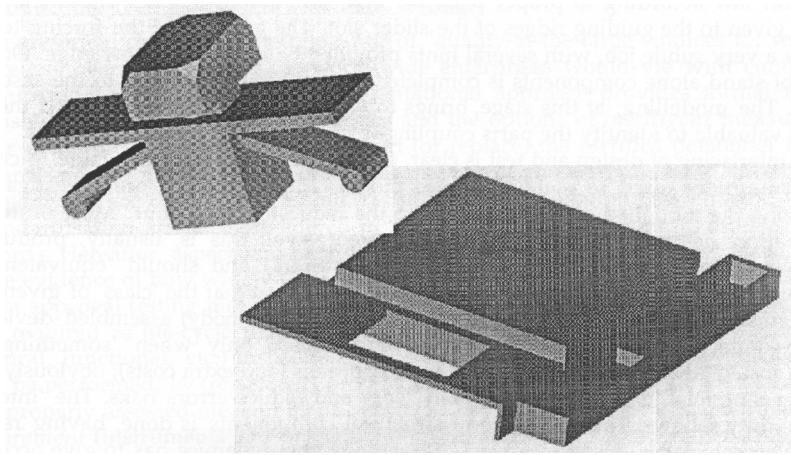


FIGURE 5.17 A CAD solid model of the sliding pin.

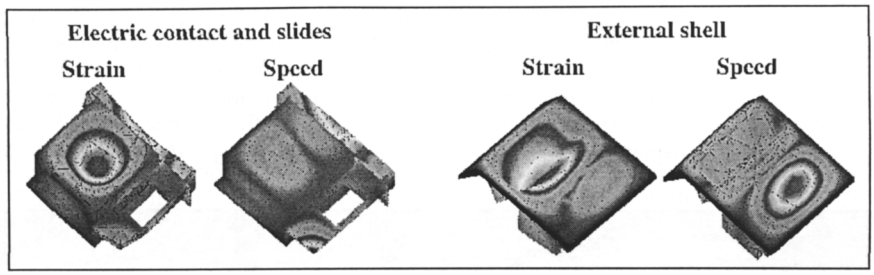


FIGURE 5.18 Typical behavioural patterns obtained by the CAD solid model.

to the guiding ridges of the slider slot. The analysis of the forcing terms is usually a very subtle job, with several hints provided by empirical knowledge. Once the study of stand-alone components is complete, the investigation turns to the assembled device. The modelling, at this stage, affords well-defined causal frames and the CAD help is valuable in identifying the parts coupling (compliance and damping) effects.

The interlacing of design and test is clear, as a specimen needs to be available to confirm the assumptions used to state the basic model. Improvements are investigated by modifying the modelled components, up to the requested behaviour. Most of the basic information must be gathered at the equipment level; this is usually produced by external suppliers (as compared to car manufacturers) and should equivalently be used for several series of vehicles (after customisation) so that the "class" of given offers is recognised. The subsequent characterisation of the (car body) assembled device (the ceiling-connected light-fixture) is then investigated only when something goes wrong and quality has to be recovered with additions (and extra costs). Obviously, direct packing is preferable to simplify assembly work and reduce errors risks. The integrated design duty follows. The modelling of structural components is done, having resort to finite elements code, based on parametric settings; the dynamics must account for different restraining links and multiple driving forces. Joined or combined parts have cross-coupling influences and the nonlinear damping effects are rarely predictable unless proper experimental data are available. The sets of sound signatures and/or extrapolated families of acoustic images become an effective means of identifying the parameters, by trimming the inertial elements and the compliance or damping terms.

The availability of new devices with the properly selected acoustic image deserves design abilities, shared by the car manufacturer and the device supplier through cooperative efforts based on transparent

access to the conditioning technicalities to obtain the predicted properties by minimal cost and time-to-market. Integrated design cannot start without guesses to predict the sound emission of highly complex structural assemblies. At this stage, the reduction to equivalent lumped parameters models is more properly undertaken; but the equivalence is consistently reached on condition that devices are already typified (actual dynamics is limited to a given inputs class), so that the vibro-acoustic behaviour is assessed for the assembled vehicle, within defined duty ranges only. Once the equivalent model is stated, however, consistent variations to the structure can be studied, to obtain adapted sound images according to some desired patterns. Based on these premises, the procedure is a validation tool, as for the concept design, leading to innovative solutions; for example, to reach a given comfort rank in terms of timbric features. The development closes at this stage, thanks to the incorporation of quality data, directly assessed by product/process trials and objectively shared by part supplier, assembly integrator, and artifact user.

Product Life-Cycle Checks and Automatic Testing

The presented case shows the relevance of quality-at-large indices for redesign (and reengineering) purposes, as worldwide enterprises would vie with competitors into fair-trade surroundings. This representational approach provides formal rules to establish standards also for *cognitive* scales and the options of *intelligent* measurement appear as important contrivance to support (or replace) the *certified* personnel duty, by standardising the monitoring and the diagnostics of the measurement process itself with *automatic* acquisition, processing, restitution, and validation instruments. Options in standardisation are further considered in Section 6, cross-referencing to established ISO standards. Hereafter, aspects are recalled, in view of quality-at-large ideas, simply to give evidence of how widely these are exploited in engineering practice. Indeed, the artifact approval testing moves more and more out of the manufacturer's sphere of action, being required as a life-cycle request. The execution of properties checks by (geometric, structural, functional, etc.) inspections is thus becoming an expanding business, which needs be performed by (registered) professionals, following metrologic standards or, at least, properly assessed measurement schemes. Returns obtained by standardising the measurement (instruments and procedures) setup could affect many fields. This subject, once again, is quite broad and even a survey outspans the limits of the present chapter. As an example case, nondestructive testing (NT) is mentioned, because it is traditionally related to industrial diagnostics.

Conventional NTs are aimed at assessing presence and seriousness of structural misfits or defects, by means of (non-invasive) measurements of related properties. The set of verifications covers cracks, cavities, inclusions, porosities, dislocation pile-ups, gradients (of composition, of grains' stretching, of welding penetration, etc.), galvanic coating detachments, surface slivers, etc. As a general rule, the results in NT belong to the field of assessing weak indicators. Measurement quality depends on the metrological consistency of the instrumental setups and of the restitution schemes. The large relevance played in the past by personnel qualification is more and more switched toward instruments, due to computer integrated solutions. Still, the selection of the sensor interface and display features is a critical job, to be faced by specialists (certified 3rd-level professionals).

The classification of NT techniques is conventionally performed by means of transducing methods at the detecting front (Fig. 5.19), giving rise to a series of measurement devices for visual tests, electromagnetic radiation tests acoustic radiation tests, etc. With the expansion of intelligent instrumentation, a better classification should include the restitution procedures to give explicit account of the aids for self-calibration, for results consistency checks, for uncertainty computation, etc. Presently, large efforts are being put forth by instrumentation manufacturers, mainly in view of the friendliness of displayed outputs, by the introduction of the computerised restitution. This option is useful, as it avoids interpretation fuzziness, but should be normalised so that all devices (automatically) yield consistent results.

Calibration remains a critical operation to be executed by referring to known sample defects. Traceability is an open question, unless for results affected by sufficiently large value band, or for diagnostic purposes of a given artifact, where the process trimming is generally sufficient for the time propagation of

Inspection Method	Result Documentation	Restitution Aids
Visual and optical inspection	Written reports (photographs)	Machine vision,
		video technology
Leak testing	Written reports (photographs)	Mass spectrography,
		bubble emission
Liquid penetrant testing	Written reports (photographs)	Image processing
Magnetic particles testing	Written reports (photographs)	Image processing,
		pattern recognition
Radiography-radiation testing	Film and paper radiography	3-D reconstruction,
		computer tomography
Electromagnetic testing	Written reports/records	Signatures detection,
		pattern recognition
Ultrasonic testing	Written reports/records	Signatures detection,
		pattern recognition
Acoustic emission testing	Written reports/records	Signatures detection and
		acknowledgment

FIGURE 5.19 Classification of NT techniques in terms of front-end transducers.

self-consistency. The idea of formalising an operation on-the-field logic which standardises the accomplishments of the quality systems is proposed as a management tactic to improve effectiveness (see the QFD method); it should be completed by the specification of the reference metrics and by the records of the conditioning procedures used for the quantitative restitution of the (nominal) results and of the associated (instrumental) uncertainty. The subject is further considered in Section 6.

5.4 Condition Monitoring and Facilities Upkeeping

The goal—quality manufacturing—is conditioned by product approval checks, based on objective experimental assessments and, therefore, requires one to extend the metrology concern with reference standards for all attributes entailing industrial interest; the option ⟨monitoring maintenance⟩ is conditioned by the re-use of quality data, to establish online diagnoses and to accomplish on-process all expedient actions, which preserve the "normal" running condition of the involved facilities. The economy of scope looks for efficiency by optimally managing the available functions and resources with the queasy compliance of sticking to the core business: unexploited opportunities are nuisances, as well as the addition of not actually useful devices or accomplishments. Concepts are clear; the practice faces difficulties at the level of principles (e.g., for fixing new metrics and selecting innovative measuring chains); and at the level of ordinary courses (e.g., for the integration of diagnostics at the shop-floor for recovery effectiveness).

The development of the subject in this section addresses the latter level topics; the instrumental setting is basically established on conventional devices and methods; in fact: the measurements concern usual quantities, such as the ones of length metrology; the diagnoses are obtained according to the conditions elucidation approach, when the specification of quality patterns and the related tolerated ranges are available as absolute data. The underlying manufacturing processes typically present tolerance patterns that readily provide a standard frame for classifying artifact quality by absolute ranks. The relevant innovation depends on data integration capabilities; therefore, it is useful to keep the attention on the equipment and its interfacing endowments, moving from collections of data commonly in use for product inspections.

Dimensional tests have traditionally been used to perform approval checks, in order to accept (or reject) artifacts according to the previously established tolerances. Results are example "quality data," verified as "spot" properties. Today, coordinate measurement machines are offered as integrated testing stations that make possible combined series of inspections, with duty adapted cycles, in order to:

- Stratify and compare data, aiming at the cross-examination of anomalies
- Investigate multiple patterns for singling out plausible causes

- Propagate the uncertainty figures with assessment of measurement confidence
- Perform, in general, the typical duties of intelligent instruments

The computer-integrated environment is such a clever option that it will not be limited to artifact diagnoses; rather, it should be turned to equip manufacturing and measuring processes for enabling full online diagnoses. Fields with high economic impact, such as in the automotive market, have already seen large investments in intelligent automation and are increasingly concerned with quality control. In such a case, the metrologic assessment of all parts that compose a car body has high relevance for the performance of the automated process, to reach aesthetic and functional goals by transparent methods and certified issues. Effective dimensional metrology fits-out are therefore required to provide on-process monitoring with integrability ranges as any other functions of the factory automation. This is, perhaps, a not fully developed issue. Investments in inspection means, indeed, have been performed, in the past, mainly in secluded equipment and dedicated gauges (as reviewed in next sub-section); with intelligent automation, the exploitation of the shop-floor information system must be fully enabled because efficiency improvement and cost reduction are, as usual, incumbent challenges.

Product Inspection and Properties/Functional Checks

Artifact approval tests have been in use as quality checks of industrial businesses, to grant the standard delivery of mass production. With steadily reset manufacturing processes, statistical control brings sufficient assurance in terms of tolerated bands; the circumstances suggest how to organise inspections on the products in such a way as to provide feedback information for the optimal setting of the processes. The outcome is referred as product monitoring and is the central scope of quality engineering. It is equivalently (but more efficiently) achieved through process monitoring, which measures the process-to-product properties and performs process keeping and retro-fitting to preserve to zero-defect manufacturing. With automation, a further step namely measurement diagnosis, is added to grant data quality of the overseeing rig and to accomplish calibration and restoring tasks.

The monitoring of industrial artifacts involves several quantities: geometric attributes, mechanical strength, functional performance, pollution level, or any property assessed by standard scales. Aimed at product diagnosis, paradigmatic issues are offered by:

- Dimensional properties and related angular/linear displacement measuring rigs
- Mechanical strength reliability and related structural-trying instrumental settings
- Operation running characteristics and related functional testing equipment

The state-of-the-art and the equipment for tolerance and geometric tests allow effective arbitration between data acquisition rate and processing versatility. For explanatory purposes, devices used for dimensional inspection are collected (Fig. 5.20) in terms of increasing flexibility, by distinguishing levels:

- Single-engagement, single-purpose devices
- Parallel-engagement, multiple-sensing devices
- Single-engagement, programme-adapted instruments
- Task-driven, programmable measurement devices
- Multiple-axes, task-adaptive measurement centres

The domain of dimensional metrology bears paradigmatic relevance when dealing with product inspection duties, and sophisticated sample developments are recalled in the following paragraphs that enter into some technical details in view of monitoring functions enabled by automatic measuring devices; properties checks and functional inspection of artifacts are common practice, that can be extended more and more by intelligent manufacturing.

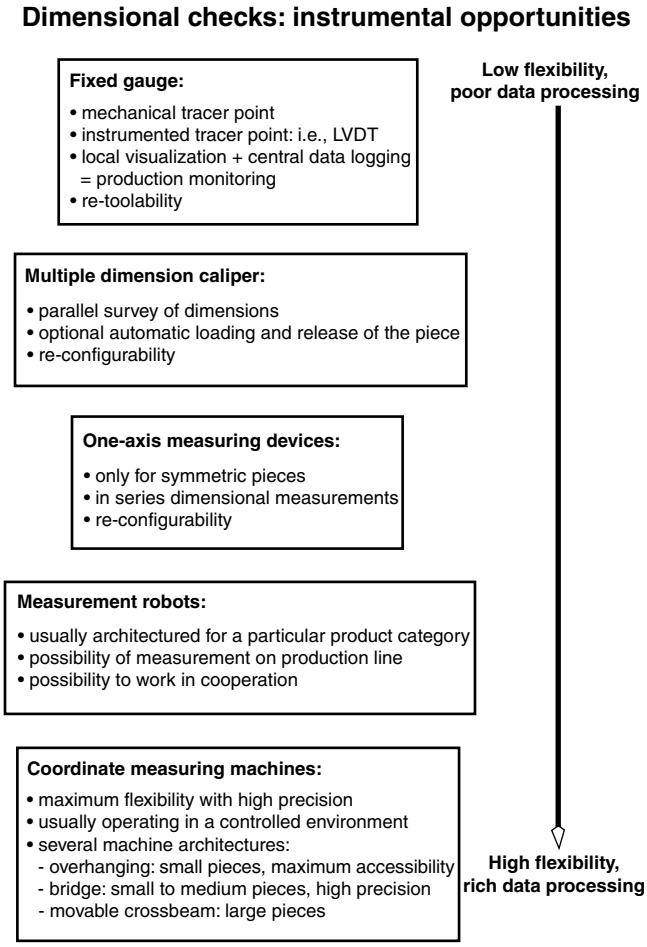


FIGURE 5.20 Classification of dimensional inspection equipment.

The artifacts' mechanical properties are sometimes checked at the end of manufacturing. More generally, measurements are performed all along the device life cycle in order to assess the persistency of suitable reliability levels. Special relevance is reserved for non-invasive techniques, based on indirect observation, with transducing elements, of the attribute to be measured. The field is expanding, due to the innovative pace of sensors and signal processors. The traditional area of nondestructive testing is fast changing, and a synthesis of principal methods and basic diagnostic abilities has been summarised.

The functional inspection also needs to monitor the artifacts in operating condition. Measurement techniques depend on the driving set-points, the performance demands, the surrounding conditions, etc., and are strictly related to the type of product's to be inspected. Testing quite often profits from the availability of intelligent instruments and computer integrated procedures, which make it possible to carry out individual checks of low-cost industrial artifacts (washing machines, household electrical appliances, etc.) or of generic supplied parts (combustion engines, electrical motors, etc.) to grant their operation performance. With, again, only explanatory purposes, the functional testing rigs are compared (Fig. 5.21) in terms of increasing versatility, by distinguishing the set of separate results simultaneously obtained: power consumption for standard duty cycles; current peaks at critical occurrences; energy saving attitude; averaged yields; performance, efficiency, safety, etc. figures; pollution, acoustic, etc. emission ranges; and others. Properties checking evolves with a hierarchic knowledge setting that distinguishes between:

Functional Test Mode	Quality Assessment	Feedback Chanches	Information Coverage	Example: Engine Test
Simplest: GO/NO GO test	The product is in the specified limits; no quantitative information on the process	No chanche	Product OK	Power, torque and other checks
GO/NO GO with data logging	The product performance level is determined; quantitative information jon the process available	Production quality trend	Product trend	Power, torque and other quantities are recorded as input/output trends
As above + measurement of non-directly-related quality indices	The product performance and reliability are determined; redundant quantitative information on the process available	Proactive production quality monitoring	Product trend and diagnostics	As above + measurement of: - vibration and noise level; - temperature of the cooling; - exahust gas analysis; - etc.

FIGURE 5.21 Classification of Functional Inspection Equipment.

- Monitored quantities: pressure, temperature, torque, velocity, vibration, noise emission, etc.
- Reference features: overpressures, overheatings, overspeed, etc.; acoustic signatures, lubricant consumption, etc.; wear, backlash, etc.

Product monitoring, leading to standard functional checks, is a comparatively well-established field. Indeed, many technical requirements are fixed by compulsory rules (electromagnetic compatibility, safety limits, pollution ranges, recycling prescriptions, etc.); some bear relevant economical issues (operation and maintenance costs, disposal and dismantling norms, etc.); while others can simply make one offer more interesting in a worldwide market. In this chapter, attention is focused on quality inspection by means of synthetic indicators, defined to measure "quality-at-the-large" figures; the topic was introduced using the sample development of the car light box, to show a way to establish engineering scales for measuring the "comfort" that buyers may obtain by a particular artifact; it is further developed in next Section 5.5.

Measuring Equipment for Automatic Dimensional Inspection

Industrial dimensional metrology and geometric testing have traditionally used a series of calipers or other gages, previously prepared for the inspection of the parts, the sub-assemblies, and the individual artifacts to verify their agreement with tolerances. These implements are produced with reference to *master* pieces, manufactured to provide the "standards"; both calipers and masters need to be checked periodically for the persistency of their metrological fitness. The situation becomes critical with large and rapidly varying product mixes. Each component to be inspected requires related masters and gages, before even starting production plans, making unlikely any quick-response programme. The trend toward efficiency (Fig. 5.20) profits from information technology, once again with precision mechanics, to propose the flexible gaging units or cells, unique measuring equipment for families of varying pieces.

Flexible gaging units are the evolution of known coordinate measurement machines, with the inclusion of computer control and the adaptation for shop-floor operativity. Typical measuring equipment (Fig. 5.22) is characterised by high-accuracy (cartesian) arm and wrist, for the proper mobility of the

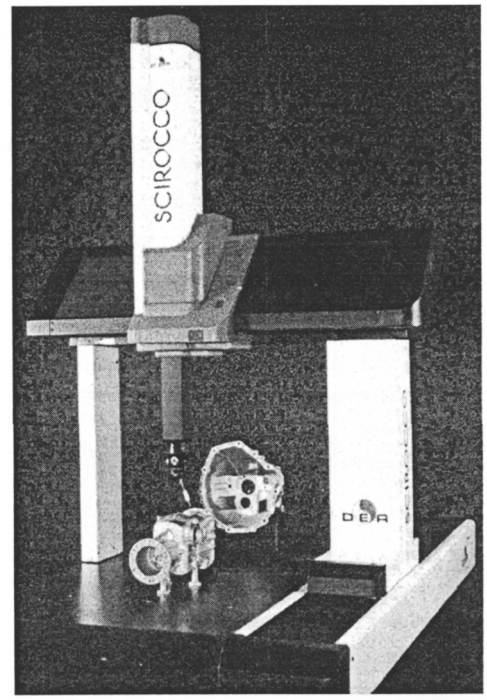


FIGURE 5.22 Flexible gauging unit, FGU. (Courtesy of Brown & Sharpe.)

Calipers	Flexible Gaging Units
Dedicated rigs, providing assessments with respect to master pieces	Calibrated devices, providing asolute solid geometric assessments
Imply certification and recertification charges, to keep inspection soundness	Require metrological confirmation with traceability of the gaging charts
Deserve storing, dispatching, and handling operations for enabling the service	Are standing instruments and need to be included by the shop logistic service
Issued as final development phase (once the proper masters are available)	Immediately ready, once 3-D models are generated by a CAD package

FIGURE 5.23 Sample comparative properties of the flexible gauging units.

sensorised tip. Pieces of any shape, comprised by the workspace, can be inspected, giving rise to absolute geometric restitutions. The gauging programme is ready to be accomplished as soon as the design of the component is complete, no matter the prototypes or master pieces. The comparison of calipers and flexible gauging units (FGUs) (Fig. 5.23) shows that drastic change, with respect to established habits, appears in view of factory automation; the approval checks are done in an absolute reference frame, whose origin and orientation are located, by proper rules, integral to each (solid) piece. The results are immediately available for further processing (statistical treatments, diagnoses, retrofitting, etc.) and for storing to keep records of the current situations and trends. Today, gauging units are equipped with usual PCs and the software outfit easily covers all practical needs, including:

- Automatic recognition and qualification of the sensor, inserted at the tip
- Quick selection procedures for generating basic metrologic elements
- Syntactic editor for off-line programming, with internal functional checks
- Extended database of form features, reference shapes, 3-D frames, etc.
- Extended database for dimension, attitude, and geometric tolerances
- Library of algorithmic procedures for reckoning 3-D solid frames

- Options for addressing and recognising different pieces of a pallet
- Options for the fast link of part-programmes to inspect sequences of pieces
- Preparation of measurement protocols, with personalised formats and messages
- Diagnostic-oriented self-calibration checks (see, e.g., the Fig. 5.45)
- Restitution of tolerances according to standards (ISO 1101, ANSI Y 14.5, etc.)
- Options for multi-point correlation, for data fusion, for threshold analysis, etc.
- Reckoning of statistical indices, as cumulative and as incremental figures
- Real-time supervision of the production process, by data feeding back
- Other similar action, compatible with the data processing module

All in all, machine uncertainty and measurement precision are also related to the test goals and to the part programme. Repeatability depends on the probe and the speed, and might be preserved to remain below 1 μm. Once the measuring cycle is established, the reference positions and directions need be acknowledged accordingly, using a frame solid to the piece; results are then independent of the fastening methods. The measurement restitution will undergo proper trimming and compensation updatings. In fact, each flexible gauging unit requires a previous accurate calibration assessment to compute the error distribution for the different workspace locations; then, departure from axis linearity, amount of bending and twisting deflections, offsetting of the frame directions, etc., are recorded and the software compensations are superposed on the actual outputs. Similar trimming is performed to account for drifts or wavering of the temperature along the structural elements; several sensors are distributed to reckon timely updatings. In this way, equipment uncertainty is primarily avoided by software compensation and calibration operations duly performed according to standard procedures (that can be run automatically on properly sensorised and fixtured units).

The online restitution capabilities and the on-process control chances deserve special attention. The charts of the statistical quality control are computed and displayed with the desired numeric and graphic formats (with concern for any particular standardisation; e.g., FORD Q 101, etc.). Measurements can be shared by several gauging units operating in parallel. Then, a centralised data evaluation system might oversee the shop production, to generate in real-time: control charts, time records, statistical indicators, uncertainty checks, oriented analyses (process capability, quality trend, etc.), etc.; and to supply feed-back data for resetting the manufacturing facilities. Consider, for example, the checks on the grinding issues of a blade fastener (Fig. 5.24); the part programme requires the localisation of the different form features by measuring over 30 points; for feedback corrections, both situation and trend evaluations are considered:

- The situation checks organise by value bands: no corrections are done for central band data; actions by the grinding machine controller are suggested to remove the bias of approved bands data; outside actions by the shop controller are suggested, aimed at verifying tool wear-out, parts provision, fixtures condition, etc.
- The trend checks look after the monitored courses (Fig. 5.9) comparing results and pre-set time histories, in view of proactive maintenance; the time response of the closed-loop regulated process is, as well, obtained by trend monitoring, with due account for the scheduled product flow.

The setting of the central band is asked to omit corrections inside the process intrinsic randomness; the identification of the process time delay is required to avoid iterating the corrections, before assessing their effects. The example retrofit can be easily adapted to different cases of NC machines (lathes, millers, etc.), with proper concern for peculiarities (serial corrections deserve absolute reference to not propagate errors; slanting surfaces need be weighed by the slope coefficient; etc.). This is important for factory automation and becomes an effectiveness factor, with return on investment, as the proper integration rank is achieved.

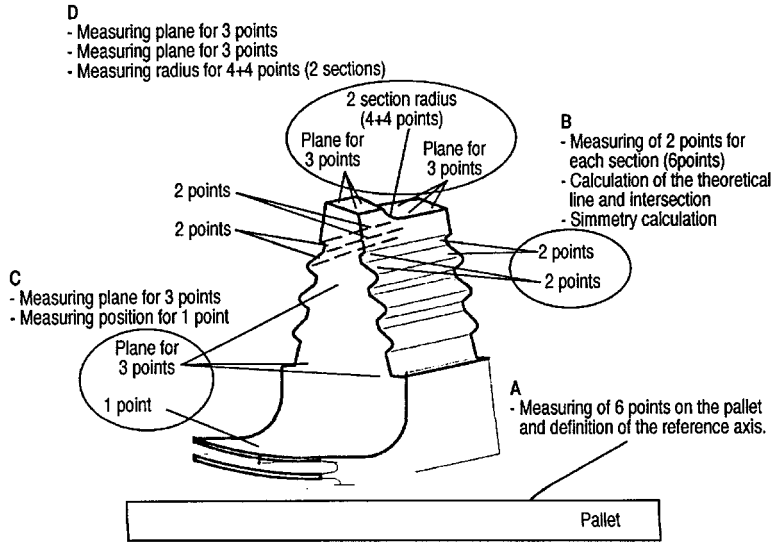


FIGURE 5.24 Blade fastener: inspection details of the part programme.

Shop-floor Cells for On-process Automatic Testing

The discussion in the present section is further delineated by referring to an example development, namely, the dimensional monitoring of (compliant) thin-wall pieces in the automotive industry. Excepting engine and transmission, a vehicle is built assembling components having sculptured geometry; for technical (structural, aerodynamical, etc.) and styling (aesthetics, habitableness, etc.) reasons, tolerances apply on 3-D shapes, with relevant fall-off on production costs (once the quality data are fixed). The inspection bodies, assemblies, parts, etc. deals with a wide variety of materials (metal sheets, glasses, plastic pieces, etc.), form features (affected by handling and positioning deflections, etc.), geometric sizes (with prescriptions on out-of-all spans, on local internal ranges, etc.), and further constraints on the amount, the rate, and the accuracy of the data acquisition. On these premises, the multi-axes coordinate-measurement machines was developed as a typical mechatronics issue, based on merging four (main) competencies: high-accuracy mechanics, multi-axes control, software programming, and sensor technology.

Today, instruments and methods are largely shared by field experts, yet continuous improvements are scheduled to adapt measurement units and stations according to the requirements of intelligent manufacturing. Basically, one should distinguish the measurement robots (Fig. 5.25), corresponding to the philosophy of modular operation units, positioned along the manufacturing lines with flow-shop organisation, from the measurement centres, (Fig. 5.26) more related to the high flexibility arrangements of job-shop setups, based on the technological versatility of the FMS. Both solutions refer to cartesian architectures for the navigation path and exploit three degrees-of-freedom wrists to settle the probe attitude; they mainly distinguish with respect to the workspace size and accessibility.

With the first setting, the horizontal beam runs along the vertical slide of a pillar, which is free to move along the longitudinal track. The strokes, practically unlimited in length and over a few meters in height, allow the transversal approach of sculptured surfaces on the top, inside and beneath the car body. Vertical rams offer lower internal features accessibility and do not permit under-body inspection. Typically, two pillars are coupled; the exploration with the symmetric arm ensures balanced longitudinal travel, supplies multi-axes inspection options, and, by paralleling the tasks, a dramatic reduction of duty-cycles is achieved. To operate on the shop-floor, the setup has to be ruggedised, aimed at behaviour independence from the surroundings, by containing the measurement station to a pressurised cell with

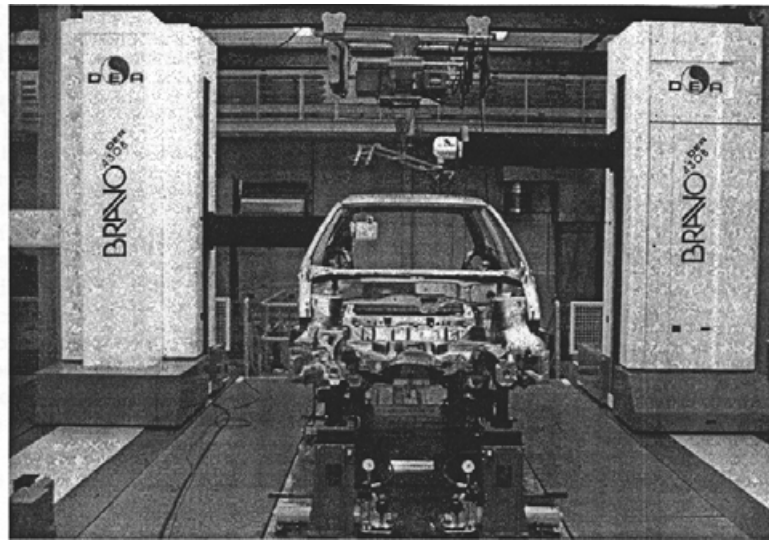


FIGURE 5.25 Measurement robots inspection cell. (Courtesy of Brown & Sharpe.)

FIGURE 5.26 Automatic inspection station. (Courtesy Brown & Sharpe.)

loading and unloading stands. The (continuous or indexable) wrist transforms each arm into a five-axes robot. At both beam tips, different probes are used.

The second setting is directly derived from coordinate-measurement machines technology, by minimising price and optimising performance. The bridge architecture is preserved, with vertical ram giving full accessibility to $2\frac{1}{2}$ components (sheet metal pieces, glass elements, internal plastic panels, bumpers, etc.). Useful upgrading is given by the structural innovation of slant guides to support the crossbars, which ensure wider bearing separation (better roll trimming), joined to less weight and lower centroid, for equal stiffness ratio. This leads to high speed and accuracy figures, while offering low-cost machines, affordable by even small firms of the supplying market.

With intelligent manufacturing, the shop-floor organisation aims at the economy of scope paradigms and the measurement machines targets have to move ahead according to new goals, such as:

- Quick-response planning: with inspection programmes and fixtures ready when new components are at the prototype stage; with automatic measuring stations, installed in the shop, capable of operating in severe surroundings; with online data restitution, to make possible proactive upkeeping; etc.
- Just-in-time scheduling: with real-time dimension checks, to preserve zero-defect delivery; with the accuracy of the inspection rooms transferred to on-process tests; with distributed monitoring actions to avoid the addition of final approvals; etc.
- Total-quality diagnostics: with oversee functions covering the overall production; with sampling frequency tuned to the manufacturing rate and increasing as deviation from normal conditions appears; etc.

The automated measurement machines are obliged, accordingly, to evolve and to include the new requirements in terms, namely, of:

- Accuracy: the maximum measuring uncertainty shall remain, for the environmental registered conditions, within certified standards (otherwise, results are meaningless).
- Flexibility: the switching in real-time between every pieces of the production mix is required (otherwise, the investment would out-grow the enterprise's budget).
- Efficiency: the speed should grant statistical consistency of the collected data, to make possible the certification of the delivered artifacts.
- Versatility: the re-setting, in front of pieces with totally different shape or size, must be done by refixturing, with resort to series of modular units.
- Integrability: the equipment will share (physical and logic) resources with the main manufacturing flow, by supporting uniform interactivity modes.

The sample architectures, Figs. 5.23, 5.25, and 5.26, reveal the change in dimensional inspection equipment, aimed at their inclusion online in order to keep the process under control. In fact, it is absolutely required that data be evaluated in real-time, with statistical consistency of the results. Under these conditions, monitoring allows one to prevent the production of out-of-tolerance parts and to establish proactive prognoses to avoid possible anomalies of the process. The overseeing function, based on predefined criteria, automatically generates alarms when given thresholds show the drifting out of stick-to-norms requisites; with factory automation, the upkeeping actions can directly follow as soon as the (unmanned) inspection cells generate the warning messages.

Dimensional Monitoring for Compliant Pieces Approval Tests

The monitoring stand outfit is completed by the selection of sensing and fixturing rigs and by the activation of the processing and govern functions. Being concerned with compliant pieces, it is mandatory to check that the nominal geometry is assumed prior to inspection; thus, the individual parts need be handled and supported by fixtures with dedicated re-shaping and holding capabilities, while ensuring proper accessibility to the measurement points. This duty is not without difficulties; in fact, the production process can often generate pieces with large position uncertainty and the measuring routines must recognise the intrinsic deviations or the deflection swerves with respect to the strain spread. Therefore, the measurements need be collected through structured probe paths (to exploit local textures of 3-D bodies) and the results should be assessed in view of their statistical consistency. The probe attitude (and not only the path followed by the inspection sequence) is critical for that duty; continuous wrists are thus required (as indexable wrists do not always grant smooth orientation).

Contact probing remains the more widely used technique because it joins reliability and accuracy, as required by the existing standards (Fig. 5.27.) The inspection sequences are established to assess the

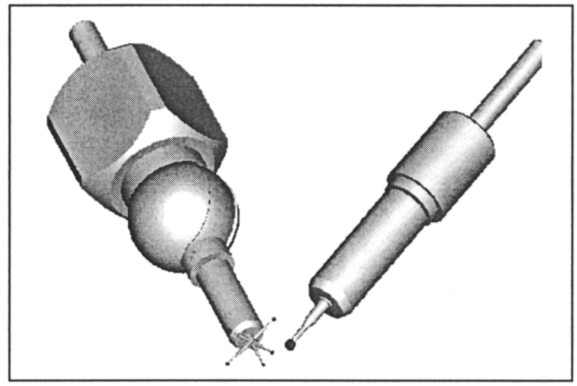


FIGURE 5.27 The typical ruby-sphere contact probe.

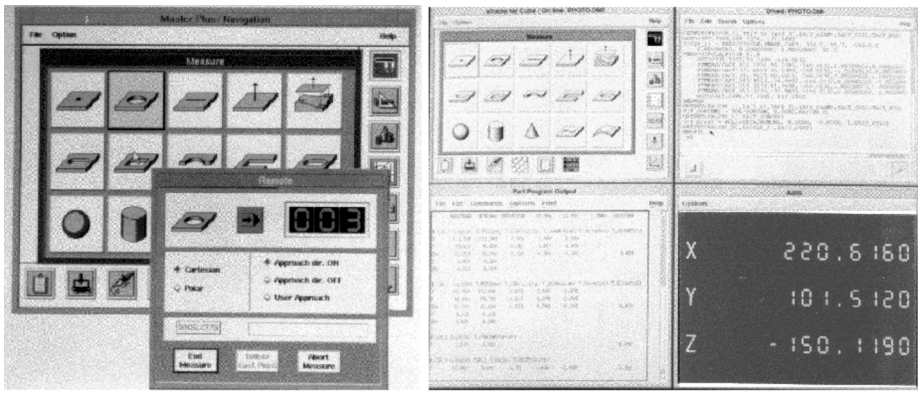


FIGURE 5.28 Programming options of coordinate-measurement machines.

position of the selected elements of each sculptured surface, both as absolute and relative locations. The non-contact probing, based on laser triangulation, is usefully combined in high-performance cells; vision technology provides benefits to supervising functions, to address the probe path and to speed-up the duty cycle. Sensor innovation progressed rapidly in recent years and some changes are likely to appear in the future, particularly as new technologies reach comparable reliability and accuracy in the restitution of dimensional data.

Fixtures, capable of granting for each compliant piece its nominal geometry, are very complex objects, deserving time-consuming design and rising to high final costs. It could result that most of the time this auxiliary equipment is not ready for part prototypes and might only be partially available at the start of production. Any change in the pieces, moreover, requires demanding fixture updatings, making it difficult to assess the actual benefits (or drawbacks) brought forth by the modification. A better approach looks for adaptive modular fixtures. These are built by means of a kit of modular devices. Once located in the workspace, the measuring robot can be used as an accurate manipulation device, to modify the modular support until the desired shapes are obtained. The concept of adaptive fixturing devices is valuable during development and at pre-series stages, when the product has to be modified and trimmed to particular goals, with due attention to the effectiveness of the results.

The flexibility of the overall hardware resources is exploited on condition that one has proper software aids (Fig. 5.28). The inspection plans will be generated and assessed in close connection with the

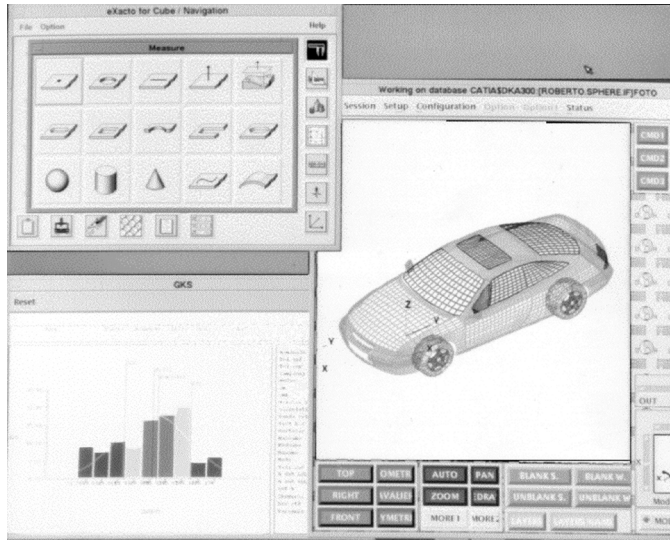


FIGURE 5.29 Example application: approval checks for car body shaping.

manufacturing process. The sculptured surfaces are specified since the design phases, but are actually brought out with manufacturing. The inspection-oriented CIME packages, thereafter, must be capable of supplying the measuring part programme (while generating the production part programme) on the basis of the 3-D models and of simulating the typical inspection cycles to detect (possible) criticalities, without requiring the availability of the parts (nor even of the related manufacturing facilities). Then, as soon as the real inspection duties are performed, the self-diagnostics of the programmes is enabled for the consistency checks of the procedures. The self-diagnostics, as further discussed in Section 5.6, are aimed at providing control, in real-time, of the reliability of the main process and of the measurement process. For online duty, the metrologic consistency of data cannot be fully stated through sufficient *a posteriori* statistics (to grant real-time operativity and monitoring maintenance actions); it is achieved by exploiting the *a priori* specification of the part geometry to perform on-process data smoothing interpolation and correlation analysis.

The dimensional monitoring stations are, last but not least, specific resources of the intelligent manufacturing environment (Fig. 5.29). They should be integrated within the enterprise information system at all levels. At the organisational level, the resources are managed according to the CIME requirements criteria. At the coordination level, the schedules are integrated to optimise the throughput efficiency. At the execution level, the shop-floor control will ensure the recovery and reintegration actions to exploit flexibility for the proactive maintenance goals. The integrated control-and-management module of the automatic measurement stations must be capable of a number of tasks and overseeing functions to give transparency of duty progression and to support the interaction of the station with external occurrences (routine actions [e.g., pieces loading and unloading, adaptive fixtures resetting] or unexpected contingencies [e.g., dispatching delay, misfit alarm]). Local controllers are presently engaged in ruling the different active safety plans, quite necessary in view of the very high operation speed of the equipment.

The basic philosophy underlying on-process corrective actions, enabled by real-time dimensional parts inspections, was summarized above, with reference to "stiff" objects, coming from computer-controlled machining centres. For the compliant pieces of the automotive industry, dimensional checks affect more extended situations. Shapes and tolerances of the (plastic strain) processing rigs have great relevance to the quality of the final metal sheet components. Focus on parts monitoring, in this case, has centrality; the collected data on the die geometry has, in fact, direct fall-off on the quality of the (drawing) process or on the tolerances of the (forged) products. Once the manufacturing resources are made ready, with their exacting requirements, the monitoring of production provides the warning for the proactive mode

maintenance of the processing rigs to preserve the desired tolerances. The details on how (in practice) these accomplishments are met can immediately follow from the sample discussion above. The integration of monitoring data is also useful for modifying the subsequent series of tasks, by joining the pieces into assemblies with the aid of proper clamping fixtures, in such a way that tolerance compensation is reached.

5.5 Trend Monitoring and Fit-with-Norms Restoring

The diagnoses inferred in the domain of dimensional inspections move away from data of the traditional metrology. Increasing industrial interest is likely to develop, aimed at diagnoses exploring drastic changes of the references, with an effective involvement of the representational scales. The topics are introduced in this section in relation to an example application, where the approval checks require the functional grading of the produced artifacts. Diagnoses mainly develop according to the features assessment approach, when self-learning is required in order to buildup sufficient knowledge for assessing fit-with-norms behaviour, leading to approved quality. The underlying manufacturing flows are typically depicted as contexts, evolving as sequences of jobs (information is updated in between each job); the space of the signatures is obtained from large time datablocks, drawing out the process features. Behavioural continuity can properly be used to detect the "wild" occurrences, or to forecast the progressive drifts from the approved "standard" conditions.

On these premises, a total quality environment can empirically be established on the basis of continuous observation of the relevant attributes of the manufacturing processes and by performing situation and/or trend monitoring maintenance aimed at preserving the conditions of zero-defects production. Company-wide quality control has consequences, related to the intelligent task-assessment paradigm, aimed at manufacturing efficiency based on flexible specialisation, with decentralised decisional manifolds (MMC.96). The actual effectiveness of continuous control on the manufacturing processes, enabled by a production-wide diagnostical framework, depends heavily on the capability of extracting reliable data during very short test phases and of feeding back the corrective actions, so that every off-set from fit-with-norms is forced to vanish. Basically, aiming at monitoring maintenance, the quick-response demand is achieved with the fast drawing out of pertinent signatures and the well-timed reintegration cycle; aimed at the release of 'certified' quality, the functional grading for all artifacts should be carried over with transparent criteria. Happily, this second accomplishment does not require added cost to operate after standardisation of the measuring methods.

The Experimental Setup for Vibro-acoustic Signatures Diagnostics

The example case, hereafter used for explanatory purposes, refers to approval tests on the totality of the assembled vehicle engines, performed to detect trends and to enable trimming actions that reinstate the standards of zero-defects manufacturing over the entire processing facility. Vibro-acoustic emission is already widely exploited to assess functional figures (AlP.79) during approval tests or to analyse engine life-cycle health. Quality measurement requires one to acknowledge subsets of signatures that identify each individual anomaly, as contrasted with standard running settings. The checks are complex; links between signatures and physical phenomena are not without question. Once each typical ill-running symptom (e.g., internal rattle, bench knock, wild ping, crankshaft rumble, valve tappet slap, rod knocking, etc.) is detected, the resetting of the manufacturing flow must be done, based on the prescribed reference models. These symptoms are normally discovered by trained experts, listening to engines' acoustic emission over long trials against brake equipment, when the dynamical phenomena reach and preserve their characterising steady-state running conditions (pitch patterns). The tests could cover several minutes; sometimes, they need be repeated for several rotation speeds before reaching reliable separation of the superposed effects. In other terms, the detection performed by human experts requires complex and time-consuming operations and can cover only subsets of mass-produced items.

A total-quality diagnostical frame (TQDF) needs to identify the relations that bind design-and-manufacturing variables and product quality actual figures, to establish upkeeping rules of the productive facilities, for preserving steady quality production. Tests, on all items, should last a few seconds for real-time consistency with flow-shop organisations. Vibro-acoustic analysis is expected to offer a powerful means for setting effective diagnostical frames; it develops with measurement schemes without interfering with the engine behaviour; the setting is a two-step business:

- First, the diagnostical frame must be validated, selecting the signatures to be detected, the thresholds to be maintained, etc., by off-process trials leading to the identification of effective (as for metrological consistency) monitoring schemes, suitable for real-time operation.
- Then only, can the instrumental setup be included on-process, to assess if the normal-conditions are properly kept; the monitoring maintenance is thus enabled to preserve the running operations by control and reintegration loops.

The off-process step must be accomplished first to select signatures that might be recognised as a distinguishing feature of a given engine functional behaviour. One needs to remember that approval tests are used, in this context, not only to accept an engine according to (certified quality) standards, but as well to strictly control the considered manufacturing process. The second step is afforded as soon as the causes of incipient anomalies and the off-sets (with respect to normal conditions) to be removed are understood. The real-time control loops generally require several measurements. Moreover, they need to include the sets of approval tests on the products, mainly for trimming purposes; thereafter, a total-quality diagnostical frame (TQDF) supplies, at the shop-floor level, the fit-up to continuously enable a proactive philosophy. Hereafter, the setting of the TQDF, to be integrated in the monitoring rig of an engine mass-production plant, is discussed insofar as selection of signatures is concerned, with due regard for features responsiveness and timing requirements.

New measurement schemes are required which join high data-compression capabilities (since the totality of the engines need to be tested in fully operative running condition) and oriented consistency (for reliable fit-with-norms assessment and process normal-condition keeping). For real-time control, free acceleration/deceleration checks supply consistent monitoring means, as soon as the symptoms to be detected and the actually measured signatures are reliably associated. Therefore, the reference diagnostics must be established after off-line experimentation, to understand the relational frame in view of proactive maintenance; namely, to assess:

- The overall phenomenic contexts, with sophisticated experimental facilities to grant metrologic consistency and calibration reliability of the monitoring equipment
- The (finally) selected signatures appropriateness, to prove the theoretical foundation of the restitution procedures and to clarify their practical meaningfulness

Let us review the first aspect. A general-purpose experimental facility (Fig. 5.30) performs vibro-acoustic measurements to collect (PoT.86) quality data (IMR.93) of the individual engines. Basically, it uses sampled (with 50-kHz rate) and digitised signals; datablocks are stored for after-processing. Further information is simultaneously gathered, such as the synchronisation pulses (every 720 motor-angle); the thermodynamic status (air, water, and exhaust gas temperatures); the instantaneous angular velocity (observing thresholds on the flywheel); etc. The facility authorises several different checks (RuT.76) at quality diagnostics, it should be noted that each type of abnormal behaviour in terms of performance (max. torques, current specific efficiency, power peaks, etc.) or of operation (rod knocking, internal rattle, wild ping, etc.) also depends on the running condition.

The possibility of performing comparative measurements is important during the build-up step of the TQDF. A typical general-purpose test sequence (Fig. 5.31) would include several steps. Only results of STEP 7 directly interests the total-quality programme to be run online and on-process; the other data, however, are required for setting and assessing the diagnostics frame. Now, the vibro-acoustic images of the free acceleration/deceleration test cannot be extracted by usual data processing. Thus, alternative options have been experimented, built with modulation windows, that may be suitably adapted for the

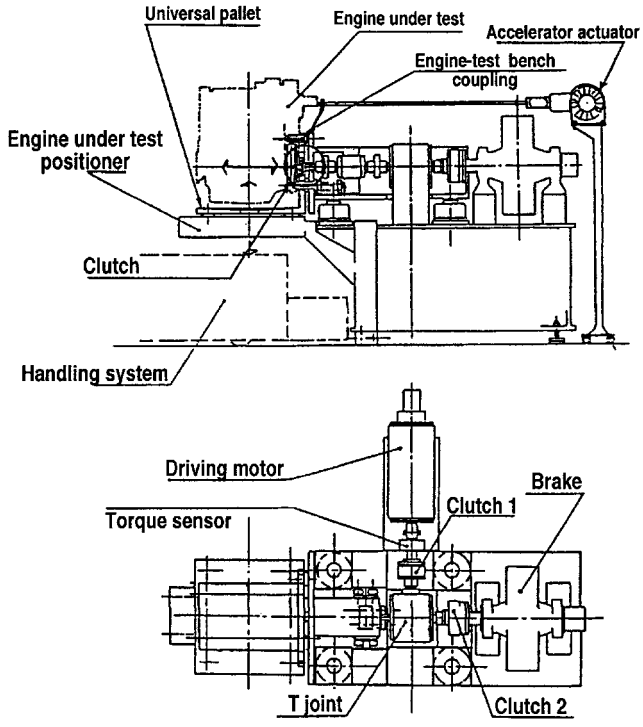


FIGURE 5.30 Experimental stand for engine quality data assessment.

STEP 0: Preparation of the test rig, by loading the pallet carrying the engine to be inspected.
STEP 1: Inspection of the engine; branching the measurement gauges, coupling to the electrical drive, etc.
STEP 2: Training the engine at low speed (around 10 rpm) to detect serious faults (spurious "lost" parts, etc.).
STEP 3: Driver disconnection and functional start-up/coast-down coupled to a brake: bump tests; check of individual cylinders at low speed; point-measurements (pressures, temperatures, etc.).
STEP 4: Control of combustion uniformity; measurement of compression ratios of each cylinder, recording of the power delivered to the brake; measurement of emission outlets.
STEP 5: Multiple steady-state referencing (for diesel engines): combustion and emission checks at requested running conditions.
STEP 6: Setting the combustion control instrumentation and the feedback gains of catalysed injection engines: gauging of the l-probe, trimming the sparking plugs.
STEP 7: Brake disconnection and execution of the free acceleration/deceleration tests (between 700 and 7000 rpm in about 20 sec) for assessing the TQDF.
STEP 8: Repetition of tests 3 (minimum speed) and 4 (standard steady-state), after warm-up (with 'high' temperatures conditions).
STEP 9: Branching-off the engine; unloading the pallet; test closes and a new inspection is ready to start.

FIGURE 5.31 Typical measuring sequence with a general-purpose facility.

application (IMR.93), or based on the exploitation of the wavelet transform (Dan.88), (LeM.86), (Ma.l89), (RiD.91). Measurements according to both approaches require:

- Selection of the reference functional base and specification of the filtering gains
- Development of the representational mapping and of the data restitution code
- Assessment of the characterising signature space and of the related metric

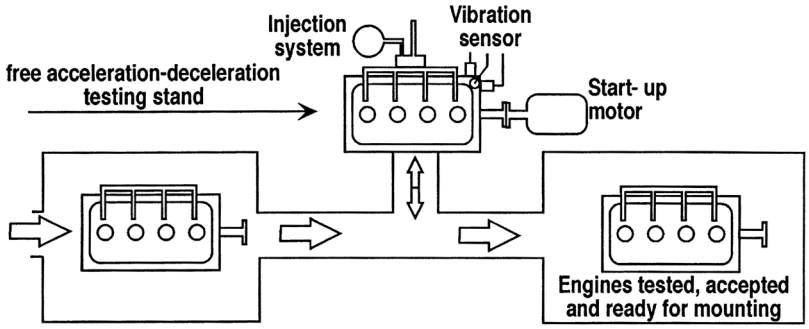


FIGURE 5.32 On-process TQDF based on free acceleration/deceleration tests.

The feasibility of the diagnostical frame moves through (LMR.93) two orders of checks: the processing code is verified to understand each mapped representation soundness; and the signature spaces are generated to record the vibro-acoustic signals of "standard" and of "anomalous" engines and to recognise the differences. The methods are then ready to be used by the TQDF and, when appropriate, the upstream control for the process resetting and the downstream alarm for the product trimming are issued.

The Online Detection of Non-persistent Vibro-acoustic Images

Free acceleration/deceleration tests actually reduce to last no more than a few seconds. They shall be performed directly along the manufacturing lines (Fig. 5.32) after the final assembly stand. The measured images refer to transient periods and could include the (eventually existing) troubles only at level of incipient phenomena; their detectability depends on energy storage build-up laws. Under these conditions, the signal restitution procedures cannot be reduced to frequency monitoring; rather, it requires generalised spectral analysis techniques. In fact, the signature consistency presents theoretical and practical problems:

- Only non-persistent acoustic signals are gathered, to detect timbric modulations.
- The engine defects appear as local modifications of fit-with-norm running images.

Hindrances are removed by data restitution procedures, tuned to separate the features from the current signal non-persistency, using independent mapping of both time and frequency modulations. The underlying system hypotheses supporting computer-based signature extraction procedures can roughly be referred to as timbre-mapping of the vibro-acoustic signals, as contrasted with simple pitch analysis. Trained experts, indeed, refer to complex (with regard to steady-state frequency spectra) signatures that combine modulation effects yield by superposing the troubles (to be detected) on the standard dynamical behaviour of the engine. In general, in fact, the diagnostics heavily depend on the tuning-by-training ability (a self-learning procedure). As soon as the feature is properly characterised, however, the detection of the identifying signature can be specialised, standardising a diagnostics not directly related to human hearing abilities, when this authorises higher reliability and better data compression, while performing the selective timbre analysis. Hints on the restitution procedures are summarised for both evolutionary spectra and wavelet transforms, to introduce the online monitoring maintenance ideas.

A timbre analysis can use weighed harmonic patterns, based on the evolutionary spectra concept, or time-frequency wavelet transforms, based on modulated convolutions. The weighed spectra bear metrologic consistency; patterns keep the physical interpretation of averaged power distribution over the frequency domain. The spectral estimates, moreover, can be performed on a single process realisation provided that the time evolution of the mapped patterns is sufficiently slow with respect to the needed frequency resolution to detect the feature. In other words, acquisition and restitution procedures cannot be validated independently; whereas, weighting spectral parameters and acceleration slopes need to be tuned to each other, to set the diagnostics.

With the wavelet transform, the intrinsic cross-conditional effects of acquisition and restitution are provided by the algorithm itself. The selection of the functional base provides the means for particularising the representations, leading to the most effective mapping of the property to be detected. The extracted signatures are then conventionally ranked with respect to the automatically constructed scale. Results bear metrologic objectivity on condition that one has standardised the (joint) acquisition and restitution method. The next two sections provide introductory comments on both techniques; the literature in the field is quite large and, for details, we defer to the quoted references.

The Evolutionary Power Density Signatures

For stationary processes, the spectral analysis is performed in keeping with the power density PSD estimate, $\tilde{\Psi}(\omega)$; this is the mean power distribution or averaged quadratic signature over the current observation window, since it measures, globally for a given time span, the signal square intensity. For time-varying processes, the PSD expresses the scaled weight that the collected harmonics (while varying with continuity) contribute to the phenomenon, on the normalised time span. The interpretation bears mapping consistency, joint to diagnostics relevance, for the evolutionary power density, EPSD, $\tilde{\Psi}(\omega,t)$, as it expresses the time behaviour of power spectral distribution estimates (Pri.88), averaged over the current observation window. EPSD measurement, however, faces subtle metrologic problems and the technical literature does not provide standard restitution procedures, limiting the consistency of actual results (as compared to the stationary situations) due to the individual processing instruments. The approach is based on evaluating the quantity $\bar{\tilde{\psi}}(\omega, t)$ at time t, using the raw estimate over the local observation window (of duration T_w):

$$\Phi(\omega,t) = \left|\tilde{x}(\omega;T_w)\right|^2 \qquad \text{where:} \quad \tilde{x}(\omega;T_w) = \int_{t-T_w/2}^{t+T_w/2} w(t-\tau)\times(\tau)e^{-j\omega\tau}\,d\tau \tag{5.4}$$

and introducing a modulation window-wave $v(t)$ over the time span T_V:

$$\bar{\bar{\tilde{\psi}}}(\omega, t) = \int_{t-T_v/2}^{t+T_v/2} v(t-\tau)\Phi(\omega, \tau)d\tau \tag{5.5}$$

One can further define: T_o, overall recording window; T_x, periodogramme span, within which phenomena are referred to as quasi-stationary; and Δf_x, expected spectral resolution, that is, conventional frequency-band assuring that the signature detection is fulfilled with reliable assessment of each individual trouble. Then, synthetically, the following inequality chain holds:

$$T_o > T_x > T_v > T_w > (\Delta f_x)^{-1} \tag{5.6}$$

meaning that (see Fig. 5.33):

- The local observation window must be wide enough to ensure the required spectral resolution; therefore, $T_w > 1/\Delta f_x$.
- The modulation averaging window should yield to a stable spectral estimate and, thus, $T_v > T_w$.
- The averaging window, however, should not obscure dithering trends or process-inherent time evolution, requiring that $T_v > T_x$.
- The overall record should grant the possibility of verifying the quasi-stationarity conditions, making $T_o > T_x$.

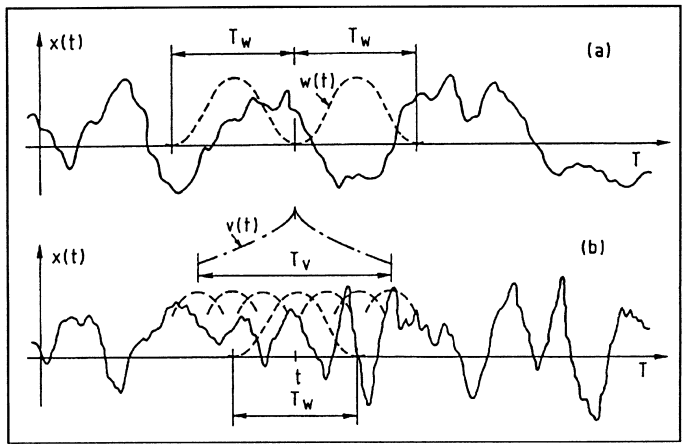


FIGURE 5.33 Characterisation of windowing combined effects.

Further note that on one side, to fade away the random noise, T_v should be much wider than T_w; on the other side, to reduce the systematic errors, time and frequency resolution should both be optimised, making: $T_w >> (\Delta f_x)^{-1}$ and , $T_w << T_X$, and therefore bringing T_v as close as possible to T_w.

Referring to the selection of the windows' shape, the existing technical literature on the subject is quite loose; the following suggestions can be pointed out:

- The round-off of the running observation window $w(t)$ is important for smoothing away the mapped side lobes, provided the systematic discrepancy of the main lobe is kept under control. A cosinussoidal window seems to be a practical reference, in view of standardising the restitution procedure.
- The local weights may usefully refer to uniform main averaging and exponential-decay bilateral averaging. The fading symmetrical effects of modulation, when referred to any central point (consistent to quasi-stationary conditions), provide unbiased information on the trend behaviour.

Within the previously quoted considerations, the EPSD is measured according to the expression:

$$E[\hat{\tilde{\psi}}(\omega, t)] \cong \int_{-\infty}^{\infty} |w(\omega - \xi)|^2 \bar{\tilde{\psi}}(\xi, t)\, d\xi; \tag{5.7}$$

where

$$\bar{\tilde{\psi}}(\omega, t) = \int_{-\infty}^{\infty} v(t - \tau)\tilde{\psi}(\omega, \tau)\, d\tau$$

This leads, in terms of expected value, to a spectral figure locally averaged with respect to time (with weighing factor v) and with respect to frequency (with windowing effect w). In addition:

$$\frac{var[\hat{\tilde{\psi}}]}{\tilde{\psi}^2} = 2\pi \left[\int_{-T/2}^{T/2} v^2(\tau)\, d\tau\right]\left[\int_{-\infty}^{\infty} |\tilde{w}(\xi)|^4\, d\xi\right] \tag{5.8}$$

provides a guess of the (relative) variance of the current estimates. On such grounds, the restitution can be put forward on standard processing equipment, with unified scales for the EPSD estimates and for the related uncertainty figures. The technical literature on the evolutionary spectral analysis is quite extended; typical references dealing with time-frequency decomposition include: (AuG.9), (Bar.97), (Coh.89), (CrR.83), (CWD.91), (CWF.76), (DaG.91), (Gab.46), (GFG.91), (HlB.92), (JiD.90), (Juu.97), (Kay.87), (Mar.86), (Mec.92), (MiR.90a), (Pri.66), (Pri.88), (Wor.90), among many others.

The Wavelet Transform of Timbric Signatures

Wavelet transform, on the other hand, is an already known data-processing technique, with specialised application in the domain of the timbre analysis of sounds (for voice recognition, synthetic music generation, etc.). It is characterised by the ability of assessing non-persistent signatures even if these represent very limited power fractions of the overall acoustic emission. The property depends on the use of modulated observation windows: the width span is depicted by a scaling parameter a; the timbre feature localisation is defined by the running parameter t. The scaling factor performs a type of frequency translation: small a evidences high-frequency features; large a evidences low-frequency trends.

The wavelet transform can be done with different functional families bases, typically selected to be orthogonal (for computation benefits) and complete (for smoothed back-syntheses). The modulated Gaussian waves are widely used as a starting reference, being good processing means to establish a standard for quality assessment schemes. Their functional characterisation is found in literature (Mey.90), for example, in the form:

$$H(t;\alpha,\tau) \;=\; \alpha^{1/2} h[\alpha^{-1}(t-\tau)]; \tilde{H}(\omega;\alpha,\tau) \;=\; \alpha^{1/2} \exp(j\omega\tau)\tilde{h}(\alpha,\omega) \tag{5.9}$$

expressed moving from time waveforms or from the related Fourier transform; where, in particular, the functional base is described by:

$$h(t) \;=\; \exp(-j\alpha\omega t)\ \exp(-t^2/2);\ \tilde{h}(\omega) \;=\; \exp\left[\frac{1}{2}(1-\alpha)^2\omega^2\right] \tag{5.10}$$

The continuous wavelet transform of a real finite-energy signal $f(t)$ is then given by:

$$\hat{f}(\alpha,\tau) \;=\; \int f(t)\, H(t;\alpha,\tau)dt \;=\; (2\pi)^{-1}\int \alpha^{1/2} \exp(j\omega\tau)\tilde{h}(\alpha,\omega)\tilde{f}(\omega)d\omega \tag{5.11}$$

The time-frequency selectivity of the wavelet mapping is easily perceived, referring to the shape of the weighing factors for two functions of a discrete-parameter collection (Fig. 5.34) having scaling factor $a_o = 2$ and a not vanishing translation factor ($t_o = 0$), so that:

$$a \;=\; a_o^{-m}; \qquad n \;=\; a_m(t/t_o), \qquad \text{and } t_o, \text{ unit shift;} \tag{5.12}$$

where the ordering factor m can be both positive and negative. With (large and) negative m, the function is wide and big translation steps t/a are appropriate; with (large and) positive m, the function is steep and small translation steps are needed for covering the same interval.

By now, several wavelet-transform codes exist, with mapping algorithms based on alternative functional bases. Aimed at standardising the restitution procedures, the use of quantised (or even binary) orthogonal sets of functions looks promising. For example, the wavelet-based images, reckoned with the Shensa modification of the Mallat scheme (Wal.92), are issued by an efficient computer procedure that avoids spectral limitation (with mapping at the dyadic lattice only) by interpolating along the current time t in such a way to provide suitable resolution and to localise every relevant timbre feature.

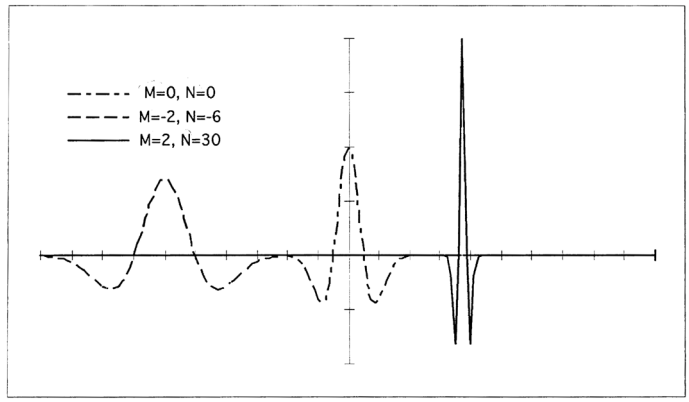


FIGURE 5.34 Shapes of typical wavelets used for the mapping.

The technical literature on wavelet transforms cover quite extended fields; example contributions are: (ABM.91), (AGH.88), (BCD.92), (Bea.75), (BCD.92), (Chi.92), (CMW.91), (DaG.91), (DaL.88), (Dau.88), (Dau.90), (Dav.91), (FMZ.90), (FoS.91), (FrJ.90), (Haa.10), (LeM.86), (MaH.92), (Mal.89a), (Mal.89b), (MaM.91), (Mar.86), (MaZ.89), (Mey.90), (RiV.91), (She.90), (Str.89), (Tut.88), (VeH.90), (Wal.92), (WoO.92), (Wor.90), (ZaZ.91), (ZHL.90).

Sample Outputs and Case Discussion

The two restitution schemes, leading to evolutionary spectra or to wavelet mapping, have been tested and validated using as reference data the sets of signals obtained by the vibro-acoustic monitoring of engines (MiR.93), (MiR.94), on a fixtured stand, as mentioned (Fig. 5.30). The comparative investigation of the restitution procedures requires a general-purpose setup (Fig. 5.35) for assessing, respectively, the influences (and for mastering the effects, with due concern for standardisation) of weighed averaging for the EPSD estimation and of time-frequency modulation for wavelet mapping. The setting of the diagnostics requires extended experimentation on several engines. This preliminary step cannot be omitted for enabling company-wide quality control plans; it corresponds to the initial assessing of the quality metrics to be used for on-process monitoring of the engine's totality. For mass-production, the quality data are, in addition to the enterprise-based goal of pace-wise betterment, also (MMC.96) a customer-oriented request, as market will turn to be ruled by certification paradigms that will limit (or even suppress) trade vehicles out of approved (and standardised) technical tolerances.

For practical purposes, the instrumental restitution of the characterising images bears great importance to grant unambiguous interpretation. This should be considered when establishing the standards for plotting the EPSD signatures. The graphic presentation of wavelet-transformed spectra, as well, is not without problems, because mapping of a two-dimensional function vs. two independent variables is required. An example EPSD image is shown in Fig. 5.36, where results, related to an engine with valve rattle, are given. A simple linear scale is inadequate, due to the quick increase in the vibration amplitude. A good discrimination is offered by a normalised linear scaling (as shown on the right of the figure), which provides the time trends of the spectral shape (and not its absolute value). The information on the actual amplitude is provided by a logarithmic scaling (on the left of the figure), with the drawback of the compression of the dynamical range. The two displays have, indeed, complementary properties. The spectral signatures allow reliable detection of valve rattle, even if the free acceleration test (from 800 to 6000 rpm) is reduced to last as short as 5 sec. The signature has an impulsive-like behaviour, not strictly related to the engine speed; it is thus evident all along the test.

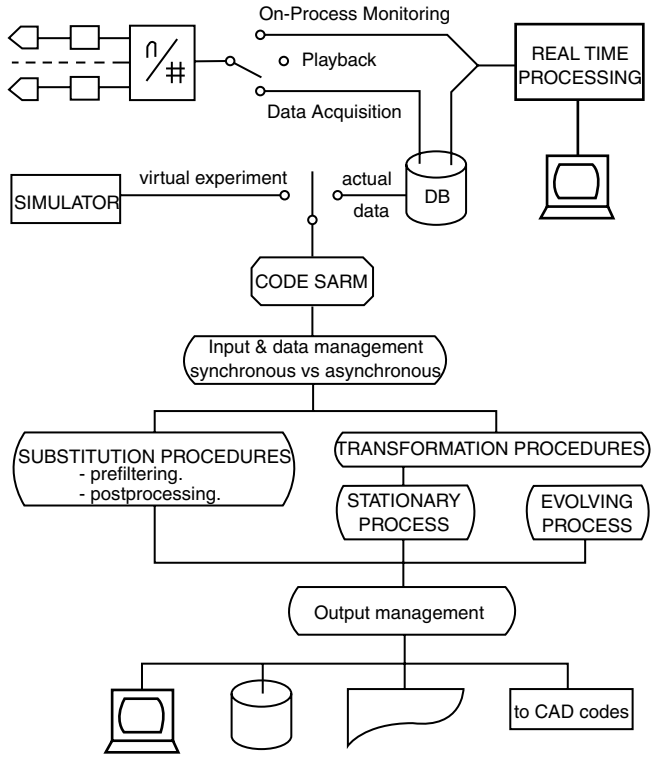


FIGURE 5.35 Example of block schema for the investigation of diagnoses frames.

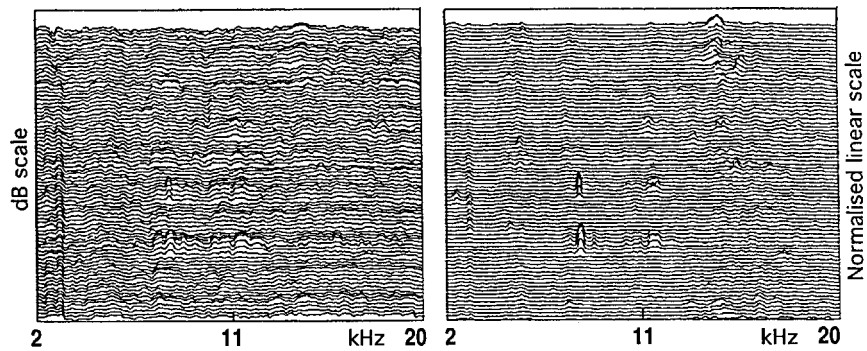


FIGURE 5.36 Evolutionary spectra of an engine with valve rattle: (a) dB scale; (b) linear scale, normalised to unit power.

The features given by wavelet mapping present further problems because their patterns are not usual as compared with the spectral display. A parametric representation can be used, giving the level plotting of amplitude and phase vs. the time sweep t and the width scale a. The level plotting is obtained by means of a chromatic mapping (actually [LMR.93] with an 8-colour resolution scale, from red to violet), which provides useful indications after black-and-white reproduction of the example outputs of Figs. 5.37 and 5.38. Both diagrams have the independent time sweep t (in ms) on the horizontal axis and the nondimensional width scale a (expressed as f_c/f, where f_c is the sampling frequency of 50 kHz) on the

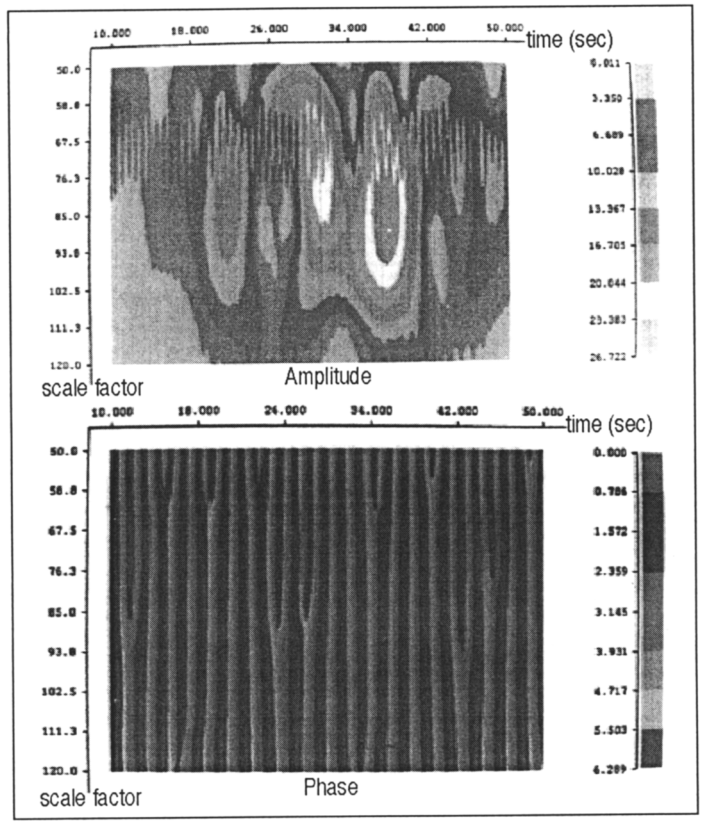


FIGURE 5.37 Wavelet image of an engine with rumble in the range 3560 to 3650 rpm.

vertical axis. The signal processed for the first presentation (Fig. 5.37) corresponds to engines having strong crankshaft rumble in the 3560–3650 rpm range. As compared with the reference behaviour of good-quality engines, the phase mapping shows level bifurcations at resonances between the engine bench transversal waves and crankshaft torsional vibrations. The wavelet spectral image of Fig. 5.38 assesses the different situation of an engine with the rumble in the 4200–4400 rpm range, which does not correspond to a crank-shaft torsional eigen frequency. Internal energy storages do not build up and the phase diagram will not show the characteristic bifurcations.

The example results show that product monitoring can become a feasible option, under conditions that "sufficient" knowledge on phenomena and instrumental setups are assembled. Special attention should focus on "intelligent" measurements and on the new opportunities offered by combined "product-and-process" monitoring, jointly with the "measurement" monitoring. Checks to accept/reject internal combustion engines in terms of functional properties are, at the moment, performed for enterprise internal goals, in view of manufacturing betterments. Measurement efficiency is the primary business and the mentioned developments deserve special interest.

5.6 Maintenance of the Monitoring System

Aimed at "quality" assessed by weak indicators, user satisfaction is mediated by business constraint, such as the one stated by the quality function deployment (QFD) approach (Fig. 5.39); namely, by enabling operation on-the-field logic, that, in keeping with the established praxis, organises on sets of

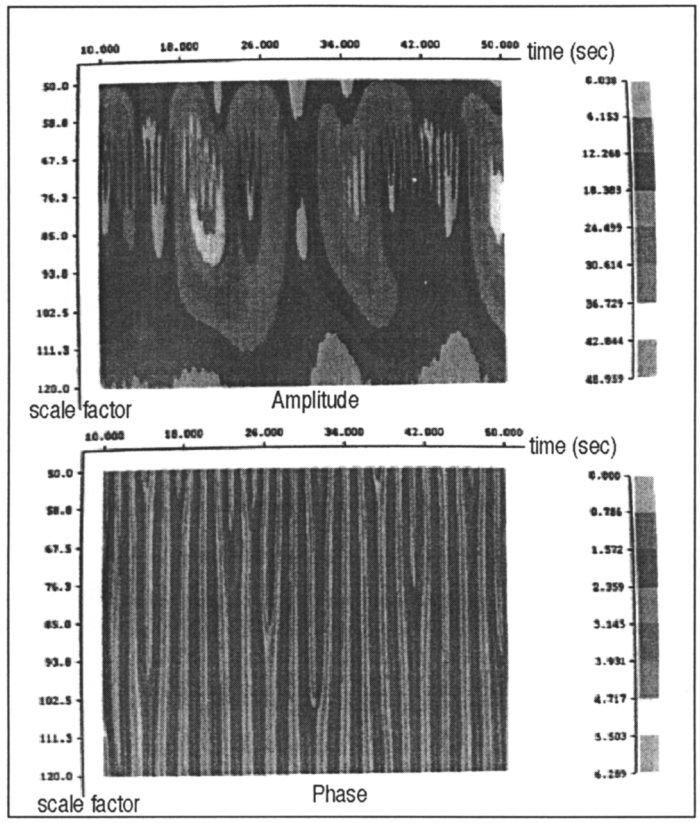


FIGURE 5.38 Wavelet image of an engine with rumble in the range 4200 to 4400 rpm.

Deployment of tolerated quality issues:
- Deployment of technologies
- Deployment of cost factors
- Deployment of reliability factors

Deployment of assured quality:
- Deployment of expected quality:
  - Total quality programming
  - Secured quality management
  - Life-cycle quality planning
- Deployment of tested quality:
  - Assessment of quality indicators
  - Selection of resources, parts, fixtures
  - Selection of processes, cycles, methods
- Deployment of intrinsic quality:
  - Deployment of piece-wise continuous improvements
  - Deployment of cooperative knowledge processing
  - Deployment of lean-engineering audit report
  - Deployment of proactive monitoring maintenance

Etc.

FIGURE 5.39 Breakdown of quality function deployment schedules.

hierarchically linked tasks and sub-tasks, rather than on the measurement of sets of quantitatively achieved issues.

The breakdown shows the information frames of strategic or tactical or executional actions, as they appear to headquarters, during the management of the enterprise goals, with business-driven objectives to exploit the potentialities of internal quality audits. As soon as strong consumer movements and worldwide competition is established, however, the market will require metrologic consistency and will impose third-party ruled standards and technology-driven certified quality (Mic.92a). This means that, as long as a self-consistent legal metrology is defective, to certify quality one must undergo the extra cost of the acknowledgment by external accreditation systems, which only may sustain fair trade conditions, with protection of the *bona fide* manufacturers and consumers.

The outlines show that quality management presents evident economical returns, on condition that product-and-process properties are assessed by measuring meaningful indices, liable of yielding quantitative information for the upgrading operations. This may suggest an engineering (quantitative) approach to the QFD method; as soon as the enterprise strategies are approved, in fact, the method is worth to provide tactical details for monitoring process and product and for acknowledging the current status provided that appropriate metrics are available. The QFD scheme, indeed, is a means (not a goal); the quality assessment by quantitative restitution is a step ahead which, when enabled, splits over two accomplishments, namely:

- Situational analyses, to give visibility of the local situations by measuring the quality indices on statistical consistent samples (to specify the performance over tactical spans)
- Relational analyses, to transparently account for the conditioning effects on the process by means of models that, to some extent, need presume causality and driving agents

Thereafter, the management and the maintenance of the measuring instrument by means of *quality systems* standards will establish registered metrics and, at both levels, will provide formal references for the enterprise fulfillment; basic hints address existing deep-knowledge conditional frames. The extension to "cognitive" assessments and to quantitative mappings adds charges on the monitoring and diagnostics of the measurement process itself, to grant objective opposability of results, since the visibility of the decisional manifold overseeing the data processing flow will simultaneously be assured; sample comments are introduced below. The next section reviews options of "intelligence" in instrumentation.

Intelligent Measurements and Instrumental Reliability

The connection between manufacturing and measuring in monitoring maintenance justifies the role played by instrument innovation for return on investment obtained by exploiting knowledge-intensive environments on full quantitative bases. The terms "manufacturing" and "measuring" are given broad reach, to cover all material constructive duties (including artifact design) and all data processing duties (aimed at managing quality). Innovation leads to "intelligent" measurements (Fig. 5.40) meaning the extension of computer-aided testing (CAT) techniques by interlacing data flow (with detection, processing, and restitution) and control flow (with rig shaping and consistency check). Such interlacing makes possible the automation of sets of actions originally charged to operators, with standard consistency and reliability of the applied procedures and of the delivered results. On these grounds, the most important properties of intelligence (Fig. 5.41) are:

- The automatic setting and resetting ability for the measuring chain: the data flows take tracks singled out by series of requirements; for example, kind of results to be obtained; environmental constraints; supervening occurrences; need for stratifying and comparing issues; etc.
- The automatic evaluation of the measurement quality: the uncertainty (intrinsic of the measurand, due to external influence quantities or to instrumental deficiencies) is processed concurrently with (nominal) data and provided with standard formats.

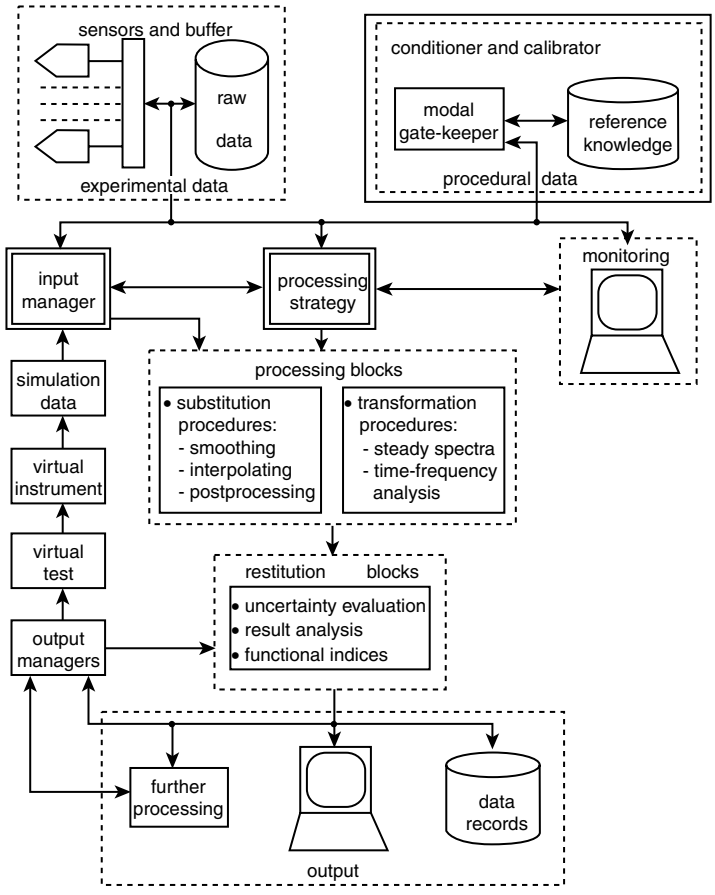


FIGURE 5.40 Block schema of intelligent instrumentation.

=> self-verification, self-diagnosis and self-calibration, for instrument validation;
=> pre-conditioning of monitoring data, based on system hypothesis;
=> integration of different sensors data or other transmitted information;
=> flexibility in visualisation, archival and printing of data;
=> multiple signal processing alternatives, with parallel/branching options;
=> possibility to introduce operator decision in the choice of the measurement strategy;
=> support for the data restitution and uncertainty evaluation;
=> other similar actions for data processing or for decision-keeping.

FIGURE 5.41 Functional adaptivity of intelligent instruments.

As general rule, the CAT setups evolution is characterised at different levels, namely:

- Material resources: hardware equipment, measurement chains, etc.
- Processing blocks: programmes, coded instructions and addressing logic, etc.
- Decision methods and judgmental modes: rules for self-calibration, resetting, etc.
- Modal gatekeeper: supervising, steering, govern, management units, etc.

The availability of explicit control flows provides visibility to the acquisition, detection, filtering, mapping, restitution, etc. procedures applied to measurement signals. Resorting to intermediate buffers and

specialised databases, distributed and multiple processing resources, and hybrid (algorithmic/heuristic) programming, etc. makes it easy to establish *virtual* instruments or to run *virtual* tests, by modifying the real-time bonds, authorising data stratification and supplying decision aids to select transform and representation frames.

Diagnoses develop, thereafter, with "smart" sensors, information "fusion" abilities, "function" adaptivity, etc., thus, in general, with a series of (merely) instrumental options that simplify features drawing, pattern recognition, and retrofit selection. The formal approach of the representational theory of measurement helps to standardise the functional adaptivity of intelligent instruments; and this (Fig. 5.40) mainly means:

- Representation adaptivity: by supplying metrologic scales, automatically modified with due account for the traceability of the reference standards
- Mapping adaptivity: by addressing, in the data processing library, the right transforms with regard to the desired signatures

Moreover, CAT fitting shares advantages (and drawbacks) of the digital mode. The signals have properly coded formats, with transparent mastering of the resolution and reliable propagation of the (local) uncertainty. The processing blocks are (actually) sets of instructions, with properly assessed input/output characteristics; their modifications, due to programmes, can be monitored and/or back-tracked to ensure visibility to the (actually used) method. The keeping of CAT devices is easily established by defining proper virtual tests, performing the related checks, and enabling self-restoring and self-calibration activities, whenever required, in order to support instrument management (Fig. 5.42) through automatic accomplishments. This option is very important because the maintenance of the monitoring setup, according to proactive mode rules, in a nondisposable requirement to ensure reliable prognoses.

Monitoring and diagnostics of the measurement process further provide information on the currently enabled conditional contexts and this helps, in the learning phases, to verify appropriateness and effectiveness of the (hypothesised) relational frames, when, in particular, knowledge-based architectures are prearranged for concurrently adapting processing blocks and decision aids. The switch to "intelligent measurements" is an option offered by computer technology to increase monitoring potentialities in view of complex duties; a reference instance might be phrased as follows (CMR.97):

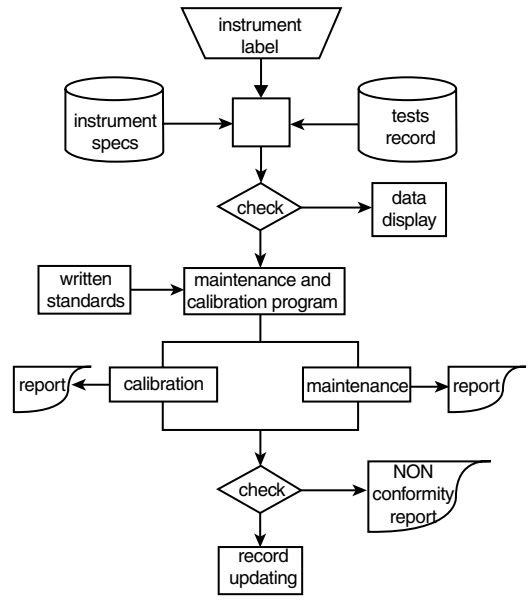


FIGURE 5.42 Flowchart for instrument management.

- Instead of the routine request: ⟨*obtain the measurement of given quantities, within a specified process*⟩
- It should be answered to the request: ⟨*perform the quantitative characterisation of the observed process, from a selected standpoint*⟩

For diagnostics of mechanical systems, the acoustic, fatigue, thermal, vibratory, wear-out, etc. standpoints are common requirements for goal-attuned assessments. Complexity is understood realising that, to answer the first request, one presumes that: the standards for the metrological mapping of the physical process is established and thus the scale of the quantities to be measured is consistently fixed; the instrumental chain uses devices whose attitude of correctly performing the measurements has been assessed and it is thus simply required to check their calibration. With the second request, convenient decisional and processing autonomy of the intelligent system is required because: the current selection of models to represent phenomena is deferred after on-process checks and can be continuously updated; instrument appropriateness is obtained by *in situ* tests, with the possibility of reducing the goals within actually reachable bounds. On these premises, the capabilities of the intelligent measuring techniques clearly appear; the development of standards precisely in the area of the maintenance of the monitoring systems is an obvious extension in bringing, as previously pointed out, the QFD method a step ahead.

Quality Maintenance of the Measuring Instrumentation

The assignment of quality to products or services (at competitive costs) is strongly influenced by manufacturing, inspection, confirmation, and settling actions. Company-based requisites are very important elements that need be improved for the efficient control of quality and cost of the output. A reference assessment needs to consider typical conditioning facts, such as the following.

- The final quality and cost of a product or service are directly related to requirements as defined at the initial presetting and development stage.
- The actual quality and cost are specifically related to the processes and organisation used during the generation of the product or service.
- The process quality strongly depends on the characteristics of instruments, resources, and technologies, and it is affected by enterprise management.
- The quality costs tend to migrate from more controlled to less controlled phases (with the side condition that: "you can't control what you can't measure")

All in all, a knowledge-intensive setup exists and runs efficiently, on the condition that the monitoring system itself complies with the clauses of total quality, supplemented by rules for quantitative assessment established on metrologic consistency with provision of pertinent confirmation aids (Fig. 5.43). The current reference document for the requirements is the ISO 10012 Standard, which is organised in two parts:

1. The system for the metrology confirmation of the equipment
2. The management and control rules of the measurement process

The first part deals with the technical assessment of the instruments, the second with the monitoring, diagnostics, and certification of the measurement process. Both stem from paragraph 4.11 of the ISO 9000s series, concerning the equipment for measurement and testing. ISO 10012 explores the subject in more detail than the ISO 9000s, giving guidelines for implementation.

Let us discuss the first part. It essentially pursues different kinds of requirements that apply to all the involved (traditional or "intelligent") instruments:

1. *Design of the monitoring system*. The equipment should be reliable and capable of the accuracy sufficient to assess the product quality (conformance to specification).
2. *Technical correspondence*. The measurement engineer should establish, maintain, and document the organised record to assess, confirm, and use the instrumentation, stating the responsibility and including the internal auditing; within such a system, technical decisions are involved, concerning:

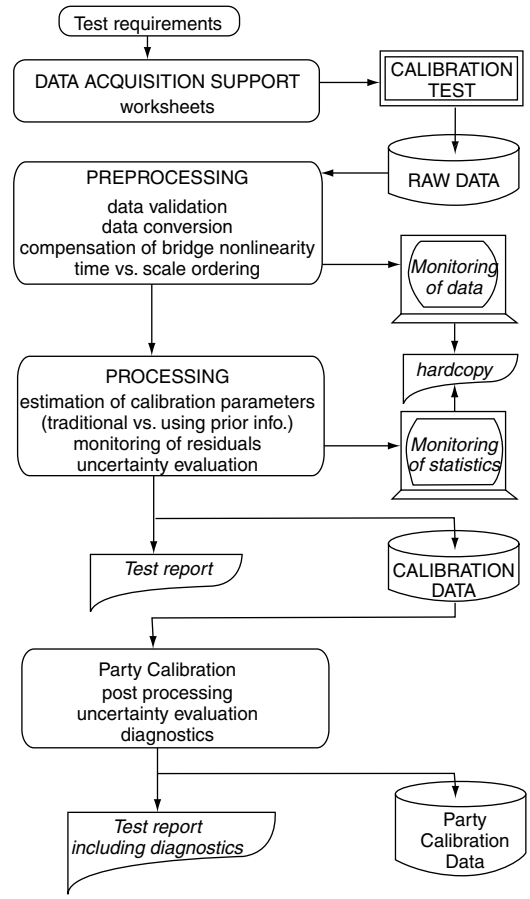


FIGURE 5.43 Example computer aided calibration setup.

a. The scheduling of calibration tests;
b. The drawing up of written procedures;
c. The retrieval and use of data, for the evaluation of the uncertainty;
d. The recovery procedures, to face the non-conformity;
e. The logistics prescriptions: labelling, handling, and safeguard.

3. *Technical certification*. The supplier should be able to document every property of the results, including the data traceability and the uncertainty evaluation. Implementing such a system, according to registered metrics rules, requires two main activities:
   a. The interfacing with external, properly accredited, metrological infrastructures;
   b. The setting of an internal, duly empowered, dissemination system.

The balancing of these two activities, tailored to the enterprise's characteristics, becomes essential to optimise cost and efficiency. Concerning the internal dissemination service, proper facilities, qualified personnel, and pertinent procedures should be provided. The assessment of the measuring instrumentation will, accordingly, be stated (Fig. 5.42) once the programmed maintenance strategy is acknowledged and scheduled.

The concern about equipment metrologic confirmation involves routine maintenance rules. Among all actions needed for correctly using the measuring instruments, calibration plays a central role; it aims, first, at the following three non-removable goals:

- To verify the conformity of an instrument to some acknowledged specifications (this is more properly called "verification," and gives a YES/NO result)
- To provide the updated input-output restitution mapping or gauging relationship
- To give information for the evaluation of the current (instrumental) uncertainty

The overall set of calibration data, when properly stored and processed, provides information on the state of instrument health, while still within tolerance bounds, to establish proactive upkeeping plans. This is a strict request, as proper running of the measuring system is a reliability precondition for any diagnostical setting.

A fourth goal for the instrument calibration follows thereafter; namely:

- To provide information to preserve conservative normal conditions for every device

In view of instrument proactive maintenance, the effective setting of calibration procedures should take into account the following:

- To reduce the number of the calibration tests to a minimum and to reduce the time of each test; this may be done by providing automation support to the operator
- To reduce the risk of mistake, by ergonomic interfaces
- To make proper use of the *a priori* information, again to reduce costs
- To use innovative ideas, such as party calibration, to improve the diagnostics settings

The computer aided calibration (for example, Fig. 5.43) can benefit from integrating different strategies, including prior information and party calibration (MiR.96a), with the comparison of performance directly enabled as an instrumental attribute.

Summing up, aiming at assessing the quality data of given artifacts consignments, the ISO 9001 (point 4.11) is the guide to establishing the main prerequisites (Fig. 5.44) to grant instrumental fitness; these precepts need include combined checks, with integration of:

- Design phases: to select consistent metrological methods and to set the related equipment
- Administration rules: to have resort to reference standards for the traceabilty of the obtained results
- Execution sequences: to document testing appropriateness and results-reliable compatibility

Measurement Process Control: Sample Results and Assessments

The interlacing of manufacturing and measuring is not without purport in view of managing the related functions. In fact, the technical competencies are always the basic prerequisites for effectiveness validation based on economical checks; the fuzziness is, possibly, wider when dealing with "total quality," or similar subjects, quite extensively proposed, by Western World corporations, as business or administration means to equip the enterprises with more effective visibility of the inside organisation. With quality engineering and metrology standards issued for fair trade legalisation, the situation is likely to change, as observed, whether or not, for example, the QFD method is further extended to deal with quantitatively assessed

- To design the testing environment: "the approval checks to assess the conformance to the given specifications of an artifact need: - to establish the measurements to be performed and the related confidence bounds; - to select the testing equipment fit to obtain the information with the proper accuracy and consistency;"
- To guarantee instrumental fit-out correctness: "the assessment of the instrumental fitness requires the previous calibration of all equipment by means of approved standards (or, these lacking, the gauging criteria ought be specified) and the concomitant acknowledgement of every disturbing factors or biasing effects;"
- To support measurement duties uprightness: "the checks on the experimental surroundings imply to verify that: - all instruments are properly employed as for duty ranges as for uncertainty consistency; - the environmental conditions are at any time compatible with the testing plans."

FIGURE 5.44 Precepts for the Metrological Confirmation of Measurements.

schemes. The overall subject is, therefore, complex and in evolution. The management and the control rules of the measurement process are, similarly, not yet fully specified; individual cases only can be faced properly and referring to particular solutions.

We will present few ideas always aiming at registered metrics assessments only, with special example discussions for explanatory purposes. The standard ISO 10012, part 2: "Measurement Process Control" is the written reference document to be considered. In that standard, the measurement is viewed as the main process to be addressed for doing overseeing actions, being aware, of course, that "the measuring equipment is only one of many factors affecting measurements," once place of work and condition of use are included. The precepts (Fig. 5.44) previously given at the equipment level should be reconsidered, again at the process level. On these premises, for consistency, the management and control of the measurement process in based on the monitoring of every relevant fact, according to series of accomplishments, such as:

- Proper design of the equipment and related procedures, accounting for all factors that may influence its performance
- Before-hand confirmation of all the involved instrumentation, as specified in the previous paragraph
- Establishment of a surveillance system, capable of complying with prospective risks of failure, with the record of normal conditions departure, as cases arise
- Presetting corrective actions in case of threshold overpass, as compared to the accepted operating conditions
- Keeping records of every relevant events
- Certification that staff has proper qualification and that the overall system is audited periodically

Leaving aside the many other organisational aspects of quality systems in view of only the metrological consistency of the accomplishments, it is stressed that the above monitoring system has a twofold purpose:

- To ensure the reliability of the measuring process (instrument quality being presumed)
- To provide information for evaluating the measurement uncertainty

As already pointed out, due to quick changes in the domain, the topic is explored by means of sample problems.

While implementing the effective instrument control and diagnostics for on-duty operation, major problems are encountered in order to dispose of suitable standards for online check and to ensure proper reference for estimating the local measurement error:

$$e(t) = y(t) - x_0 \tag{5.15}$$

ideally, for every x_0 to be measured.

The difficulty of the task highly depends on the kind of the manufacturing process and of the checks that need be accomplished. For example, consider the following cases:

1. For the dimensional tests, on spot-like arrangements: the following situations arise:
   a. For limited flexibility instruments (such as calipers), the usual dimensional gauges may be appropriate;
   b. For highly flexible instruments, such as the CMM, simplified self-calibrating methods can be foreseen;
   c. For both cases, an already tested replica of the object to be measured may be used; for instance, for a CMM, a simple three-ball artifact (Fig. 5.45) may be monitored in different positions,

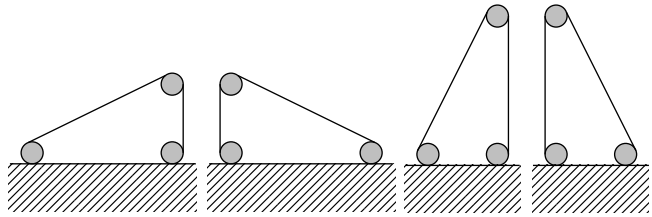


FIGURE 5.45 Simple tricks for CMM diagnostics, based on self-calibration checks.

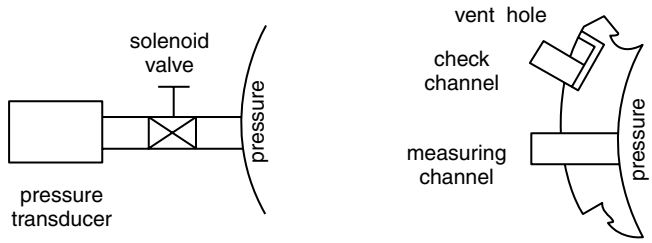


FIGURE 5.46 Pressure measuring system check by remote-control connection.

so that after proper processing of data, based on angle assessments, the influence on the measuring machine performance is obtained.

2. For the measurement on continuous processes, the problem may be more critical because it may be difficult to have online, actual standards of quantities, such as temperature, pressure, flow rate, and the like; possible solutions might be:

   a. The timely connection with portable calibrators, which however requires one to switch off the sensors from the measurand;
   b. The addition of check channels, which grant automatic remote bypassing or dummy connection; for instance, source check-channel solutions (Fig. 5.46), as suggested by P. Stein (Ste.96), can be used: a remotely operated valve (left) makes easy to disconnect a pressure sensor from the measurand environment to check the instrument zero; alternatively, a dummy channel is used, which duplicates a pressure sensor, being exposed to the same influence quantities (e. g., exposed to the same temperature and vibrations), then, the monitoring of disturbing variables is accomplished with continuty;
   c. The exploitation of redundant instrumentation:
      i. By means of transducers of the same kind, providing a virtual standard obtained by proper averaging over the output of the set of redundant transducers;
      ii. By means of transducers of different kind, which may be even more reliable (Fig. 5.47) but that sometimes require special (and not yet commercial) restitution devices.

Self-calibration based on sensor redundancy is the only feasible track, when extreme measurement conditions are met, as it happens (CMR.97) in the example procedure doing the instrumental gauging by averaging over several accelerometers channels; the self-tuning, proposed in this example, exploits the information (Fig. 5.48) on the estimated frequency response (gain and phase) and on the related coherence function. In this application, performance of the accelerometers had to be checked for the monitoring at very low levels (below 1 μg) and low frequency (under 10 Hz). The calibration was performed by averaging over six measuring channels of the same kind. Once the reference gauge for the error monitoring was established, data could be processed by suitable statistical tools, such as variance analysis (CIS.97).

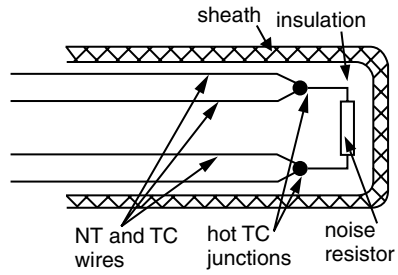


FIGURE 5.47 Thermocouple plus noise thermal sensor, for redundant measurements.

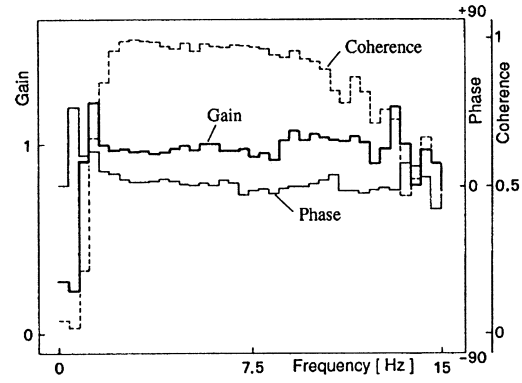


FIGURE 5.48 Example of sensor self-calibration results (useful band: 2–10 Hz).

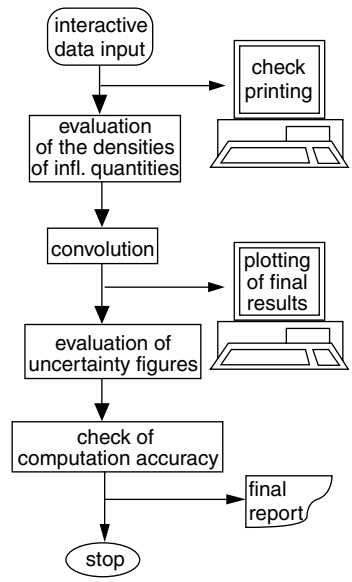


FIGURE 5.49 Flowchart of a user-oriented package for uncertainty evaluation.

Finally, after that the outputs of the measurement process are checked for reliability within the on-duty condition, the estimation of uncertainty can take place. This again is an emerging task in an industrial environment and proper user-oriented procedures should be provided and implemented. One such procedure (CMR.97b) is shown in Fig. 5.49.

The above remarks show that quality and measurement connections deserve growing interest and the technical literature on the subject is quickly expanding; a few references are: (CIS.97), (FrR.95), (HoE.88), (KuM.93), (Mic.92a), (MiR.87), (MiR.90a), (MMC.96), (Shi.89b), (TDR.97), (Yau.97), (ZNF.96).

5.7 Conclusions

Quality manufacturing appears to be the winning option on worldwide markets, where competing enterprises face educated and demanding buyers, within fair-trade conditions fixed by legal metrology and environmental protection provided by authorities' rules. For effectiveness, the issue is strictly related to the ability of joining product diagnosis, process diagnosis, and measurement diagnosis so that visibility is granted, at every level, by quantitative statements. High standing artifacts could be (mainly, in the past) the object of special care, with extra costs from conception and development, to construction and approval checks. With mass production, industrial artifacts have reached constant average specification due to standardardisation of the manufacturing means. Pursuit of consumer wants is, today, sought by items variability, jointly looking for merit and cheap offers. The achievement stems from "intelligent" setups, namely computer integrated outfits. These, in fact, make possible "knowledge-intensive" environments and help in defining extensions of the conventional (legal) metrology, to provide objective effectiveness to "quality data."

The evolution, brought forth by information technology, is in the chapter related to issues in computer integrated testing, with example achievements; for example:

- For measurement diagnostics, showing, in Section 5.6, the prescriptions aiming at the **metrological confirmation** of the instruments
- For process diagnostics, giving the supporting ideas, in Section 5.2, of the **maintenance policies** looking to proactive settings
- For product diagnostics, stating, in Section 5.3, the **quality-at-the-large** concept, based on representational theory of measurements

The achievements are mainly related to the ability of standardising CAT fit-outs and, through them, of providing visibility on the accomplished processing operation *and* on the enabled decision logic.

Quite happily, information technology supplies hardware and software fixtures up to the high sophistication range; efficient exploitation of options is, perhaps, lagging behind in terms of integrated organisation, with inconsistency zones remain on critical topics such as:

- How to rule measurement methods and devices, aiming at certified quality
- How to manage measurement monitoring and related quality diagnostics

Correct statement of the problems and balanced planning out of the solving actions are nondisposable requirements to obtain return on investment, along with clever tricks such as monitoring maintenance setups.

Manufacturing and measuring tasks are means to obtain certified quality products. On reciprocal sides, artifact quality testifies to the fitness-for-purpose of productive cycles, and measurements quality proves the appropriateness of monitoring flows. Clearly, the correct setting of the measuring process rises to implicit accomplishment; the precepts on the quality and reliability of the monitoring systems are specified by the ISO 9000 series and they are further explained by the Guide ISO 10012. Then, aiming at the instrumental requisites, the ISO 10012, First Part (Fig. 5.50) distinguishes, for process reliability, between the charges to procure the appropriate services and the ones needed to preserve their on-duty fitness.

The proper statement of measurement monitoring precepts rises to higher relevance, in front of complex instrumental rigs, when multiple information needs to be collected and combined to work out quality data, concurrently assessing the properties of the product (to assure certified quality), of the process (to check the fit-with-norms condition), of the process-to-product relational frame (to enable the upkeeping actions), and, of the instrumental setup (to oversee on measurement quality). The subject is facing quick evolving contexts, with lots of new opportunities arising, with the goal being to have properly assessed (instrument, process and product) diagnostics.

- The equipment suppliers are required, in terms of delivered components, to:
  - Guarantee that the on-duty instruments possess fit-for-purpose metrological characteristics;
  - Establish and upkeep documentation records for the tests planning, with specification of operators' responsibility and of confirmation requirements;
  - Technically oversee calibration charges by: confirmation planning out; current uncertainty checking; traceability upkeeping; records up-dating; etc.;
  - Face the maintenance, restoring, etc. actions, as un-fitness arises;
  - Supplement the instruments shopfloor logistics: - storing and dispatching; - labelling; - sealing up; - etc.
- The equipment users are required, in terms of performed tests, to:
  - Analyse the observation schemes in terms of output demands and to design the fit-for-use measuring chain;
  - Ensure the instruments metrologic fitness (as previously stated);
  - Accomplish measurement monitoring and reintegration, with records of trends and assessment of the uncertainty;
  - Arrange in advance the procedures for the treatment of the (possible) misfits;
  - Vouch for the personnel capacity, as for qualification levels, with responsibility empowerment and audit charges provision.

FIGURE 5.50 Precepts to grant the measuring fit-out.

The challenging nature of the evolving contexts provides important prospects for measurement theory and practice; notably, whether the representational approach is considered. This has introduced measurement of "quality-at-the-large" figures, as the case happens when vibro-acoustic signatures are exploited for characterising mechanical device behaviour. Sound features, indeed, are related to tone and to timbre analysis abilities, but only pitch analysis can be fulfilled by conventional phonometry. Timbric features are, instead, possibly depicted by phase or frequency modulation of broad-band waves; if modulation and modulated signals have spectral overlapping, however, their restitution by pitch analysis quickly grows to a redundant set of information, as resolution needs be properly assessed. Whether these circumstances arise, thus, data compression effectiveness remains a subject of study (and standards might, in the future, be issued for series of bounded classes of phenomena).

The capability of providing metrologic consistency to "weak measurement" by means of *cognitive* scales gives alternatives to acknowledge the sound signatures. As soon as measurement effectiveness becomes a central requirement, "new" (e.g., wavelet-based) decompositions will develop, as compared to the conventional frequency analysis, to introduce more general rank figures (based on order-tracking, dyadic-sequentiality, etc. techniques), 'automatically' assigned by the instrumental restitution (e.g., by the spectral mapping with orthonormal sequentially grouped binary wavelets). The "new" options, indeed, are an ineluctable choice for monitoring maintenance applications, as information density and time constraint pop in as technical requirements. The transition between the different metrics depends on several facts. Harmonic analysis is possibly preferred, at the release of "certified" quality devices. The example of Section 5.3 is thus carried out with conventional spectra, obtained by phonometric measurements. This is necessary, as noise emission (legal) characterisation is now performed that way, with eventually, formal extensions (e.g., Wigner-Ville distribution) which do not give up with the quite traditional duality: 'time-trends-harmonic-patterns.'

The example presentation in Section 5.5 deals, instead, with the ability to establish an objective metric with sets of nonconventional signatures. Engine monitoring will possibly become an imperative request (in front of environmental protection acts) and data quality will have to be released for the selling of cars; then 'new' measurement methods and 'new' rigs will be required (and will spread over), based on techniques such as the one discussed. At the moment, the developments have been pushed to the level of a feasibility study and a prototypal measuring process has undergone experiment. It is quite obvious that the final selection of a processing method for standardisation has to move across several further validations.

The discussion in Section 5.4, on the automatic dimensional measuring process, was limited to example situations. In this case, the metrologic context is fully assessed and the basic innovation is represented

by the integration of the detected information for attuning the manufacturing process in order to preserve zero-defect production. The trimming of the monitoring system has many technicalities (e.g., pieces compliance secludes data accuracy, unless reshaping fixtures are used) and the implementation of total-quality diagnostics becomes a business for concurrently managing the data flow (of the dimensional measurements) and the control flow (of the consistency checks). CAT equipment, in principle, provides the hardware and software requisites: data mapping, into scaled representations, is transparently done, with account for measured data and conditioning transforms, so that standardisation is fully embedded in the instrumental (hardware and software) chain. The innovation moves toward the fusion of measuring functions into manufacturing with due account for the decision support which could be provided automatically by "expert" modules.

The control and diagnostics of a measurement process are facing more and more subtle requirements as the options of "intelligent" measurements expand. The topics discussed in this chapter, therefore, are quickly changing. In fact, the data restitution procedures, grounded on shallow-knowledge conditioning frames, are enabled by AI methods, which make it possible to emulate mental processes and to automatically perform a series of judgmental operations. The outputs of weak-measurement schemes (based on cognitive psychology elements) should be tested, to verify their level of consistency with reference to both the *a priori* system hypotheses and the previously collected experimental results. The tests of meaningfulness may, sometimes, defer the acceptance of given results until the reference conditioning knowledge is *sufficiently* expanded to assert the trustfulness of the consistency analysis. When performed by means of weak measures, the evaluation of quality becomes the result of adaptive procedures, with the requirement of keeping transparency of the data processing operations in order to give evidence to the judgmental conditioning mappings. With "intelligent instruments and automatic data processing schemes, the pitfalls of conclusion inconsistency are avoided, by deferring the final statements to further analysis and by iterating or diversifying the collection of experimental data with modified and adapted measurement methods and rigs.

Acknowledgments

We acknowledge the support of CNR (Italian National Research Council), project PFT2 (Progetto Finalizzato Trasporti 2), for the studies on innovative vibration signatures. We also thank all the people who have contributed to this chapter through ideas, suggestions, and technical information; in particular, Dr. M. Ercole of DEA-Brown & Sharpe, Torino, for all the material and information relative to the coordinate measurement machines; and Dr. G. Ruspa of CRF, Orbassano, with whom we collaborate on several diagnostical and quality assessment problems.

References

(ABM.91) - M. Antonini, M. Barlaud, P. Mathieu: Image coding using lattice vector quantisation of wavelet coefficients, *Int. Conf. Acoust. Speech, Sign. Proc.,* Toronto, May 1991, pp. 2273–2276.

(AGH.88) - A. Arneodo, G. Grasseau, M. Holschneider : Wavelet transform analysis of invariant measures of some dynamical systems, *Proc. Wavelets, Time-Frequency Methods and Phase-Space,* Springer Verlag, 1989, pp. 182–196.

(AGU.97) - V. Augutis: Measuring high frequency vibroacoustic diagnostic signals of friction pairs, *Proc. XIV IMEKO World Congress,* Tampere, Finland, 1–6 June 1997.

(AlN.91) - B. Al Najjar: On the selection of condition based maintenance for mechanical systems, K. Holmberg, A. Folkeson, Eds., in *Operational Reliability and Systematic Maintenance*, Elsevier, London, 1991, pp. 153–173.

(AlP.79) - A. Alpini, G. Ruspa: Rear-axle automatic quality control system through noise analysis and diagnostics, *ISATA Proc.,* Firenze, Sept. 1979.

(AMM.88) - G.M. Acaccia, R.C. Michelini, R.M. Molfino, A. Ragonese: Failure analysis and maintenance model for a flexible manufacturing cell, *7th. Intl. IASTED Conf. Modelling, Identification and Control,* Grindelwald, CH, Feb. 16–18, 1988, pp. 80–86.

(AuG.91) - L. Auslander, I. Gertner: The discrete Zak transform: application to time-frequency analysis and synthesis of non-stationary signals, *IEEE Proc. Signal Processing,* 39(4), 1991.

(BaK.97) - D. Barschdorff, M. Kronmüller: Design of a model to detect the excitation of a mechanical wave-guide by measuring at the termination, *Proc. XIV IMEKO World Congress,* Tampere, Finland, 1–6 June 1997.

(Bar.91) - J.T. Barclay: Keep your uptime up: condition monitoring, *Engineers Digest,* March 1991.

(Bar.97) - W. Bartelmus: Visualisation of vibration signal generated by gearing obtained by computer simulation, *Proc. XIV IMEKO World Congress,* Tampere, Finland, 1–6 June 1997.

(BaS.94) - A. Balsamo, S. Sartori: Principles of calibration of intelligent instruments, *Proc. XIII IMEKO World Congress,* Torino, Italy, 1–6 June 1994, 2, 887–892.

(BCD.92) - G. Beylkin, R. Coifman, I. Daubechies, S. Mallat, Y. Meyer, L. Raphael, B. Ruskai, Eds.: *Wavelets and their Applications,* Jones and Bartlett, Cambridge, MA, 1992.

(BDI.95) - G. Byrne, D. Dornfeld, I. Inasaki, G. Ketteler, W. Konig, R. Teti: Tool condition monitoring: the status of research and industrial applications, *CIRP Annals,* 44(2), 541–567, 1995.

(Bea.75) - K.G. Beauchamp: *Walsh Functions and their Applications,* Academic Press, London, 1975.

(BeB.97) - A. Beghi, M. Bertocco: A combined GLR/Kalman filter technique for fault detection in power systems, *Proc. XIV IMEKO World Congress,* Tampere, Finland, 1–6 June 1997.

(BDL.97) - G. Betta, M. Dell'Isola, C. Liguori, A. Pietrosanto: An artificial intelligence-based instrument fault detection and isolation scheme for air conditioning system, *Proc. XIV IMEKO World Congress,* Tampere, Finland, 1–6 June 1997.

(BiK.97) - S.D. Bittle, T.R. Kurfess: An active piezoelectric probe for precision measurements on a CMM, *Intl. J. Mechatronics,* 7(4), 337–354, 1997.

(BLS.97) - G. Bucci, C. Landi, T. Scozzafava: On-line measurement for continuous NDT of steel ropes, *Proc. XIV IMEKO World Congress,* Tampere, Finland, 1–6 June 1997.

(BLW.97) - D. Blackmore, M.C. Leu, L.P. Wang: The sweep-envelope differential equation algorithm and its application to NC machining verification, *Computer-Aided Design,* 29(9), 629–637, 1997.

(Box.1965) - M.J. Box: A new method of constrained optimisation and a comparison with other methods, *Computer J.,* no 8, 1965, pp. 42–52.

(BrG.97) - A. Bruzzone, P. Giribone: Design of a study to use neural networks and fuzzy logics in maintenance planning, *Simulation Multiconference,* Atlanta, April 6–10, 1997.

(Bro.97) - S. Brons: Motor current sensing for indirect measurement of cutting forces: options and limitations, *Royal Inst. Technology,* Dr. Thesis, no 21, 1997, Stockholm.

(CDH.97) - C. Commault, J.M. Dion, V. Hovelaque: A geometric approach for structured systems: application to disturbance decoupling, *Automatica,* 33(3), 403–409, 1997.

(Cem.97) - C. Cempel: Energy processor as a model of condition evolution of systems in operation, *Proc. XIV IMEKO World Congress,* Tampere, Finland, 1–6 June 1997.

(CGM.97) - F. Crenna, L. Giancaterino, R.C. Michelini, D. Repetto, G.B. Rossi: Determinazione sperimentale dell'immagine acustica di componenti maeccanici, *III Congresso Nazionale di Misure Meccaniche e Termiche,* Portonovo (AN) 30 Giugno-2 Luglio 1997.

(CHA.92) - S.H. Carpenter, C.R. Heiple, S.H. Armentrout: Relationship between acoustic-emission and experimental variables during sliding friction, *6th Progress in Acoustic Emission,* Jap. Soc. NDT, 1992, pp. 353–360.

(Cha.97) - S.B. Chang : Sub-micrometer overshoot control of rapid and precise positioning, *J. Precision Engineering,* 20(3), 161–170, 1997.

(ChB.97) - Y. Chih Hsieh, D.L. Bricker: The scheduling of linear deteriorating jobs on multiple machines, *J. Computers and Industrial Engineering,* 32(4), 727–734, 1997.

(ChC.97) - K. Chmelik, V. Cech: Technical diagnostic of AC motors by nonsinusoidal supplying, *Proc. XIV IMEKO World Congress,* Tampere, Finland, 1–6 June 1997.

(Chu.92) - C.K. Chui: *An Introduction to Wavelets,* Academic Press, Boston, 1992.
(CIS.97) - M. Catelani, G. Iuculano S. Sartori: Surveillance and qualification of a measurement process in the ISO 9000 context, *Proc. XIV IMEKO World Congress,* Tampere, Finland, 1–6 June 1997.
(CJM.90) - K.H. Concannon, A.K.S. Jardine, J. McPhee: Balance maintenance costs against plant reliability with simulation modelling, *Industrial Engineering,* 22(1), 22–26, 1990.
(CKJ.97) - B.K. Choi, D.H. Kim, R.B. Jerard: C-space approach to tool-path generation for die and mould machining, *Computer-Aided Design,* 29(9), 657–669, 1997.
(CMM.97) - F. Crenna, R.C. Michelini, R.M. Molfino, R.P Razzoli, M. Widmer: The design-for-comfort: reference issues and a case example, *13th ADM Conf. Design Tools and Methods in Industrial Engineering,* Florence, September 17–19, 1997, pp. 7–14.
(CMP.97) - F. Crenna, Michelini, R.C., F. Pampagnin, R.P Razzoli: The management of quality standards in integrated design, *13th ADM Conf. Design Tools and Methods in Industrial Engineering,* Florence, September 17–19, 1997, pp. 467–473.
(CMR.96) - F. Crenna, R.C. Michelini, G.B. Rossi: Measuring vibration patterns for rotor diagnostics, *13th IMTC-IMEKO TC-7,* Brussels, June 4–6, 1996, 1, 597–602.
(CMR.97a) - F. Crenna, R.C. Michelini, G.B. Rossi: Techniques for the integrated monitoring of rotor dynamics, Proceedings *XIV IMEKO World Congress,* Tampere, June 1997.
(CMR.97b) - F. Crenna, R.C. Michelini, G.B. Rossi: Generalised procedure for expressing uncertainty in measurement, *Proceedings XIV IMEKO World Congress,* Tampere, June 1997.
(CMR.97) - F. Crenna, R.C. Michelini, G.B. Rossi: Hierarchical intelligent measurement set-up for characterising dynamical systems, *IMEKO J. Measurement,* in press.
(CMW.91) - R.R. Coifman, Y. Meyer, V. Wickerhauser.: *Wavelet Analysis and Signal Processing*, Yale Univ. Press, 1991.
(Coh.89) - L. Cohen: Time-frequency distribution: a review, *Proc. IEEE,* 77(7), 941–981, 1989.
(CrR.83) - R.E. Crochiere, L.R. Rabiner: *Multi-rate Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1983.
(CWD.91) - G.S Choi, Z.X. Wang, D.A. Dornfeld: Adaptive optimal control of machining process using neural networks, *IEEE Conf. Robotics and Automation,* Sacramento, 1991, pp. 1567–1572.
(CWF.76) - R.E. Crochiere, S.A. Weber, J.L. Flanagan: Digital coding of speech in subbands, *Bell Syst. Tech. J.*, 55, 1069–1085, 1976.
(DaG.91) - M.R. Davenport, H. Garudadri: A neural net acoustic-phonetic feature extractor, based on wavelets, *Proc. IEEE Pacific Rim Conf.,* Victoria BC, May 1991.
(DaL.88) - C. D'Alessandro J. S. Lienard: Decomposition of the speech signal into short-time waveforms using spectral segmentation, *IEEE Int. Conf. Acoustic, Speech Signal,* New York, Apr. 11–14, 1988, pp. 351–354.
(Dau.88) - I. Daubechies: Orthonormal bases of compactly supported wavelets, *Communic. on Pure and Applied Mathematics,* 41(7), 909–996, 1988.
(Dau.90) - I. Daubechies: The wavelet transform, time-frequency localisation and signal analysis, *IEEE Trans. Info. Theory,* 36(5), 961–1005, 1990.
(Dav.91) - G. David: *Wavelets and Singular Integrals on Curves and Surfaces*, Springer-Verlag, Berlin, 1991.
(DBC.97) - N.C. Do, S.M. Bae, I.J. Choi: Constraint maintenance in engineering design system: an active object oriented approach, *J. Computers and Industrial Eng.,* 33(3/4), 649–652, 1997.
(DrC.95) - J.H. Drew, C.A. Castrogiovanni: Quality management for services: issues in using customer input, *Quality Engineering,* 7(3), 551–566, 1995.
(Ell.97) - B.R. Ellis: The combined use of remote laser measurements and simple spectral analysis, *J. NDT. E. Intl.,* 33(4), 231–238, 1997.
(Fan.97) - H. Fan: An efficient order recursive algorithm with lattice structure for estimating continuous-time AR process parameters, *Automatica,* 33(3), 305–317, 1997.
(Fer.97) - P.M. Ferreira: LMS parameter estimates for volumetric error models of machine tools, *J. Precision Engineering,* 20(3), 179–187, 1997.

(FMZ.90) - P. Flandrin, F. Magand, M. Zakharia: Generalised target description and wavelet decomposition, *IEEE Trans. Acoust. Speech Signal Proc.,* 38(2), 350–352, 1990.
(FoS.91) - M.L. Fowler, L.H. Sibul: A unified formulation for detection, using time-frequency and time-scale methods, *Asilomar Conf. Sig. Syst. and Comp.,* Pacific Grove, Nov. 1991, pp. 637–642.
(FrJ.90) - M. Frasier, B. Jawerth: A discrete transform and decomposition of distribution spaces, *J. Func. Anal.,* 93(1), 34–170, 1990.
(FrR.95) - F. Franceschini, S. Rossetto: QFD: the problem of comparing technical/engineering requirement, *Research in Engineering Design,* 7, 270–278, 1995.
(Gab.46) - D. Gabor: Theory of Communication, J. IEE, 93(3), 429–457, 1946.
(GFG.91) - N. Gache, P. Flandrin, D. Garreau: Fractal dimension estimators for fractional Brownian motions, *Int. Conf. Acoust. Speech, Sign. Proc.,* Toronto, May 1991, pp. 3557–3560.
(Haa.10) - A. Haar: Zur Theorie der orthogonalen Funktionen-Systeme, *Math. Annal.,* 69, 331–371, 1910.
(HlB.92) - F. Hlawatsch, G.F. Boudreaux Bartels: Survey on linear and quadratic time-frequency signal representations, *IEEE SP Magazine,* April 1992, pp. 21–67.
(HoE.88) - Y.A. Hosni, A.K. Elshennawy: Quality control and inspection, *J. Computers and Industrial Engineering,* 15(1), 331–337, 1988.
(Hwa.97) - M.S. Hwang: A cryptographic key assignment scheme in hierarchy for access control, *Math. Computer Modelling,* 26(2), 27–31, 1997.
(HWD.97) - L. Hong, C. Wang, Z. Ding: Multi-resolutional decomposition and modelling with an application to joint probabilistic data association, *Math. Computer Modelling,* 25(12), 19–32, 1997.
(IMC.97) - G. Iuculano, M. Menchetti, M. Catelani, R. Sasdelli: On the measurement of the quality of the electricity supply service, *Proc. XIV IMEKO World Congress,* Tampere, Finland, 1–6 June 1997.
(IMR.93) - G. Irato, R.C. Michelini, G.B. Rossi: Tecniche innovative di collaudo e diagnostica per motori per auto veicoli, *Convegno Naz. di Misure Meccaniche e Termiche,* Cagliari, 10–11 Giu., 1993, pp. 193–202.
(IwM.77) - K. Iwata, T. Moriwaki: An application of acoustic-emission measurement to in-process sensing of tool wear, *CIRP Annals,* 26(1), 1977.
(Jan.88) - A.J.E. M. Janssen: The Zak transform: a signal transform for sampled time-continuous signals, *Philips J. Res.,* 43, 23–69, 1988.
(JiD.90) - C.L. Jiaa, D.A. Dornfeld: Experimental studies in sliding friction and wear via acoustic emission signal analysis, *J. Wear,* 39, 403–424, 1990.
(JLF.97) - S.J. Joo, R.R. Levary, M.E. Ferris: Planning preventive maintenance for a fleet of police vehicles using simulation, *Simulation J.,* 48(2), 93–99, 1977.
(JoG.97) - M. Jouaneh, P. Ge: Modelling and control of a micro-positioning tower, *Mechatronics,* 7(5), 465–478, 1997.
(Juu.97) - E.K. Juuso: Intelligent methods in diagnostical process analysis, *Proc. XIV IMEKO World Congress,* Tampere, Finland, 1–6 June 1997.
(JVR.90) - D. Jaume, M. Verge, A. Rault: A model-based diagnosis in machine tools: application to milling cutting process, *CIRP Annals,* 39(1), 443–446, 1990.
(KaL.97) - Y.C. Kao, G.C.I. Lin: CAD/CAM collaboration and remote machning, *J. Computer Integrated Manufacturing Systems,* 9(3), 149–160, 1997.
(Kay.87) - S.M. Kay: *Modern Spectral Estimation: Theory and Application,* Prentice-Hall, Englewood Cliffs, NJ, 1987.
(KhS.97) - D.V. Khablov, A.S. Sovlukov: Contactless microwave detection and diagnostics of dynamic objects in industry and medicine, *Proc. XIV IMEKO World Congress,* Tampere, Finland, 1–6 June 1997.
(KJE.91) - R.J. Kuoppala, E.O. Jantunen, P.A. Enwald: Condition monitored methods for rotating machinery, K. Holmberg, A. Folkeson, Eds., *Operational Reliability and Systematic Maintenance,* Elsevier, London, 1991, pp. 175–198.

(KoK.97) - D. Kozanecka, Z. Kozanecki: Experimental method of identification of torsional vibrations of rotating systems in turbomachines, *Proc. XIV IMEKO World Congress,* Tampere, Finland, 1–6 June 1997.

(Kul.97) - R. Kulhavy: System modelling and identification, *Automatica,* 33(3), 477–478, 1997.

(KuM.93) - T. Kuo, A. Mital: Quality control expert systems: a review of pertinent literature, *J. Intelligent Manufacturing,* 4, 245–257, 1993.

(LaD.84) - M.S. Lan, D.A. Dornfeld: In-process tool fracture detection, *Trans. ASME, J. Eng. Mater. Technol.,* 106(2), 111–118.

(Lar.97) - J.E. Larson: Cognitive systems engineering, *Automatica,* 33(3), 479, 1997.

(LMR.93) - S. Lavezzo, R.C. Michelini, G.B. Rossi, W. Siboldi: Monitoraggio a fini diagnostici dell'emissione vibroacustica dei motori per autoveicoli, *Convegno Naz. Progetto Finalizzato CNR Trasporti* 2°, Roma, 19–21 Ottobre 1993.

(LeM.86) - P.G. Lemaire, Y. Meyer: Ondelettes et bases hilbertiennes, *Rev. Mathematica Ibero-americana,* 2, 1–18, 1986.

(MaH.92) - S. Mallat, W.L. Hwang: Singularity detection and processing with wavelets, *IEEE Trans. on Information Theory,* 38(2), 617–643, 1992.

(Mal.89a) - S. Mallat: A theory for multiresolution signal decomposition: the wavelet representation, *IEEE Trans. on Pattern Analysis and Machine Intelligence,* 11(7), 674–693, 1989.

(Mal.89b) - S. Mallat: Multi-resolution approximations and wavelet orthonormal bases of $L^2[R]$, *Trans. Amer. Math. Soc.,* 315(1), 69–87, 1989.

(MaM.91) - F.J. Malassenet, R.M. Mersereau: Wavelet representation and coding of self-affine signals, *IEEE Int. Conf. Acoustic, Speech Signal,* Toronto, May 1991.

(Mar.86) - N. Martin: An AR spectral analysis of nonstationary signals, *J. Signal Processing,* 10(1), 61–74, 1986.

(MaZ.89) - S. Mallat, S. Zhong: Complete signal representation with multiscale edges, *Courant Inst. of Math. Sci.,* Tech. Rpt. 483, Dec. 1989.

(MCB.90) - R. McBride, T.A. Carolan, J.S. Barton, S.J. Wilcox, W.K.D. Borthwick, J.D.C. Jones: Detection of acoustic emission in cutting processes by fiber optic interferometry, *Meas. Sci. Technol.,* no 4, 1990, pp. 1122–1128.

(MCR.96) - R.C. Michelini, F. Crenna, G.B. Rossi: Quality certification of product and personnel, *Quality '96: New Quality for a New Market,* Kosice, May 20–21, 1996. pp. 70–74.

(Mec.92) - W.F.G. Mecklenbräucher, Ed.: *The Wigner Distribution: Theory and Applications in Signal Processing*, Elsevier Science, Amsterdam, 1992.

(Mey.90) - Y. Meyer, *Ondelettes et Operateurs: Tome I, Ondelettes*, Herrmann Ed., Paris, 1990.

(MCR.96) R.C. Michelini, M. Callegari, G.B. Rossi: Robots with uncertainty and intelligent automation, *Proc. 2nd ECPD Int. Conf. Advanced Robotics, Intelligent Automation and Active Systems,* Vienna, Sept. 26–28, 1996, pp. 31–39.

(MHF.97) - J.L. Maryak, L.W. Hunter, S. Favin: Automated system monitoring and diagnosis via singular value decomposition, *J. Automatica,* 33(11), 2065–2069, 1997.

(Mic.83) - R.C. Michelini: Probabilistic formulation of uncertainty on the processing of time dependent data, *IMEKO Workshop on Fundamental Logical Concepts of Measurements,* Torino, 5–6 Sept., 1983.

(MiC.85a) - R.C. Michelini, A. Capello: *Misure E Strumentazione Industriali*, UTET SpA, Torino, 1985, pp. XV 1–400.

(Mic.85b) - R.C. Michelini: Monitoraggio delle prestazioni e diagnosi del degrado nei sistemi integrati di lavorazione, *Riv. L'Industria Meccanica,* no 401, dic. 1985, pp. 719–723.

(Mic.92a) - Michelini, R.C.: Assessing quality measurements for technical certification, *Intl. Conf. Automation '92,* Budapest, Feb. 18–20, 1992, Vol. 2, pp. 321–332.

(Mic.92b) - R.C. Michelini: Decision-anthropocentric manifold for flexible-specialization manufacturing, *Proc. of Japan-USA Symp. on Flexible Automation,* Ed.: Ming Leu, ASME Book no 10338A, 1992, pp. 467–474.

(MiK.94) - R.C. Michelini, G. Kovacs: Knowledge organisation and govern-for-flexibility in manufacturing, *3rd. Intl. Workshop Cooperative Knowledge Processing for Engineering Problem Solving,* Rovereto, Italy, May 29–June 1, 1994, pp. 7·1–14.

(MiR.86a) - R.C. Michelini, G.B. Rossi: Interactive computer testing code for the identification of vibrating systems, *5th Intl. Symp. IMEKO Intelligent Measurements,* Jena, 10–14 June, 1986, pp. 326–329.

(MiR.86b) - R.C. Michelini, G.B. Rossi: IMIMS: an interactive CAT-code for the modelling and the identification of multmass-vibrating-systems, *Intl. Conf. IASTED-AFCET: Identification & Pattern Reco,* Toulouse, 18–20 June, 1986, pp. 507–522.

(MiR.87) - R.C. Michelini, G.B. Rossi: Computerised test-schema for motion recognition and evaluation of dynamic systems, *Measurement,* 5, 61–68, 1987.

(MiR.89) - R.C. Michelini, G.B. Rossi: Digital processing of vibration data for turbomachinery trend monitoring, *IMEKO Intl. Symp. Technical Diagnostics '89,* Prague, May 29–31, 1989, pp. 83–89.

(MiR.90a) - R.C. Michelini, G.B. Rossi: Generalized spectral analysis of measurement data for on-process diagnostics, *7th. IMEKO Intl. Conf. Technical Diagnostics,* Helsinki, 17–19 Sept., 1990, pp. 341–347.

(MiR.90b) - R.C. Michelini, G.B. Rossi: Signature analysis of vibration signals for turbomachinery diagnostics, *7th Intl. IMEKO Symp. on Technical Diagnostics 90,* Helsinki, Sept. 17–19, 1990, pp. 309–318.

(MiR.93) - R.C. Michelini, G.B. Rossi: The measurement of non-stationary vibrations for the trend-monitoring of vehcles engines, *26th ISATA Symp. Mechatronics in Automotive Industries,* Aachen, 13–17 Sept. 1993, pp. 693–699.

(MiR.94a) - R.C. Michelini, G.B. Rossi: The evaluation of measurement uncertainty, *13th IMEKO World Conf. 'From Measurement to Innovation',* Torino, 16–18 Sept 1994, pp. 1062–1068.

(MiR.94b) - R.C. Michelini, G.B. Rossi: Vibroacoustic signatures detection and trend-monitoring of vehicles engines characterization, *4th. ATA Intl. Conf. New Design Frontiers for More Efficient, Reliable and Ecologiocal Vehicles,* Firenze, 16–18 March, 1994.

(MiR.95a) - R.C. Michelini, G.B. Rossi: Sviluppo e sperimentazione di un sistema di misura intelligente per il monitoraggio vibrazionale dei rotori, *II Congr. Naz. Misure Meccaniche e Termiche,* Bressanone, 19–21 Giugno 1995, pp. 59–68.

(MiR.95b) - R.C. Michelini, G.B. Rossi: Measurement uncertainty: a probabilistic theory for intensive entities, *IMEKO J. Measurement,* 15, 143–157, 1995.

(MiR.95c) - R.C. Michelini, G.B. Rossi: Sviluppo e sperimentazione di tecniche di analisi spettrale tempovariante per indagini vibro-acustiche, *II Conv. Naz. PF Trasporti,* Genova, 29–31 Maggio, 1995, Vol. 2, pp. 709–722.

(MiR.96a) - R.C. Michelini, G.B. Rossi: Computer aided testing and confirmation for strain measurement instrumentation, *IMEKO J. Measurement,* 18(2), 89–99, 1996.

(MiR.96b) - R.C. Michelini, G.B. Rossi: Assessing measurement uncertainty in quality engineering, *13th IMTC-IMEKO TC-7,* Brussels, June 4–6, 1996, Vol. 2, pp. 1217–1221.

(MMC.96) - R.C. Michelini, R.M. Molfino, M. Callegari, F. Crenna, G.B. Rossi: Indicatori di parte terza per la misurazione della qualità, *Seminario SIRI: Qualità Totale nella Produzione di Componenti,* Milano, 5 Giugno, 1996, pp. 9·1–25.

(Mor.84) - T. Moriwaki: Sensing and prediction of cutting tool failure, *Bull. Jap. Soc. Precision Engineering,* 18(3), 1984.

(Mor.94) - T. Moriwaki: Intelligent machine tool: perspective and themes for future development, *Manufact. Sci. and Engineering,* 2(68), 841–849, 1994.

(Mor.97) - K. Moriwaki: The development of a 'walking drive' ultraprecision positioner, *Intl. J. Precision Engineering,* 2(2), 85–92, 1997.

(MRC.97) - R.C. Michelini, G.B. Rossi, F. Crenn: Techniques for the integrated monitoring of rotors dynamics, *Proc. XIV IMEKO World Congress,* Tampere, Finland, 1–6 June, 1997.

(MUU.97) - S. Mandayam, L. Udpa, S.S. Udpa, W. Lord: Wavelet-based permeability compensation technique for characterising magnetic flux leakage images, *J. NDT. E. Intl.,* 30(5), 297–303, 1997.

(NAE.97) - M. Niskida, T. Adaki, T. Endo, H. Matsumoto: Measurements of local elastic moduli by amplitude and phase acoustic microscope, *J. NDT. E. Intl.,* 30(5), 271–277, 1997.
(Nag.97) - S. Nagasawa: Application of fuzzy theory to value engineering, *J. Computers and Industrial Eng.,* 33(3/4), 569–572, 1997.
(NaL.97) - S.V. Nagalingam, G.C.I. Lin: A unified approach towards CIM justification, *J. Computer Integrated Manufacturing Systems,* 10(2), 133–145, 1997.
(Nar.85) - L. Narens: *Abstract Measurement Theory*, MIT Press, Cambridge, MA, 1985.
(Nov.97) - O. Novaski: Utilisation of Voronoi diagrams for circularity algorithms, *J. Precision Engineering,* 20(3), 188–195, 1997.
(OND.97) - J. Ondrouch, A. Chrápková, J. Bilos: Crack detection of turbine blades by means of modal analysis method, *Proc. XIV IMEKO World Congress,* Tampere, Finland, 1–6 June, 1997.
(PeG.94) - D.B. Percival, P. Guttorp: *Long-Memory Processes, Wavelets in Geophysics,* Academic Press, 1994.
(PhO.95) - D.T. Pham, E. Oztemel: An integrated neural network and expert system for statistical process control, *Proc. Inst. Mechanical Engineers,* 209, 91–97, 1995.
(Pri.66) - M.B. Priestley: Design relations for non-stationary processes, *J. Royal Statistical Society,* Part B, 28, 228–240, 1966.
(Pri.88) - M.B. Priestley: *Non-Linear and Non-Stationry Time Series Analysis*, Academic Press, London, 1988.
(PoT.86) - C. Ponti, G. Turino: Diagnosica su motori Diesel in condizioni di accelerazione e decelerazione libera, *Conf. Naz. ATA: Affidabilità & Diagnostica Industriale,* Torino, 26–27 Giugno, 1986.
(Pul.91) - U. Pulkinen, A stochastic model for wear prediction through condition monitoring, K. Holmberg, A. Folkeson, Eds., *Operational Reliability and Systematic Maintenance*, Elsevier, 1991, pp. 223–243.
(Rao.97) - B.K.N. Rao, Ed.: *Handbook of Condition Monitoring*, Elsevier, New York, 1996.
(RaS.97) - M. Rajala, T. Savolainen, A framework for customer oriented business modelling, *J. Computer Integrated Manufacturing Systems,* 9(3), 127–135, 1997.
(RiD.91) - O. Rioul, P. Duhamel: Fast algorithms for discrete and continuous wavelet transform, *IEEE Trans. on Information Theory,* 38(2), 569–586, 1992.
(RIL.92) - J. Richard, Z. Idelmerfaa, F. Lepage, M. Veron: Quality control in a flexible manufacturing cell, *CIRP Annals,* 41(1), 1992.
(RiV.91) - O. Rioul, M. Vetterli: Wavelets and signal processing, *IEEE SP Magazine,* Oct. 1991, 14–38.
(RKP.95) - T. Ramdén, P. Krus, J.O. Palmgren: Condition monitoring of pumps, using vibration signals and neural networks trained with complex optimisation, *Intl. Symp. Vibration and Noise*, Venice, 1995.
(Rob.79) - F.S. Roberts: Measurements Theory, Addison-Wesley, Reading, MA, 1979.
(RuT.82) - G. Ruspa, G. Turino: Engine automatic quality control system through noise analysis and diagnostic, *ISATA Proc.,* Wolfsburg,Vol. 2, 13–17 Sept., 1982.
(RuT.76) - G. Ruspa, G. Turino: L'analisi di 'sigla' e sue applicazioni, *Convegno Controlli non distruttivi,* Orbassano (Torino), 15 April, 1976.
(SaA.97) - M.B. Saintey, D.P Almond: An artificial neural network interpreter for transient thermography image data, *J. NDT. E. Intl.,* 30(5), 291–295, 1997.
(SaL.97) - S. Sajin, V. Lukonin: Semi-conductor leak detectors sensitive element researching, *Proc. XIV IMEKO World Congress,* Tampere, Finland, 1–6 June, 1997.
(ScA.97) - R.M. Scala, S. Arnalte: An information system for computer integrated manufacturing systems, *Robotics and Computer Integrated Manufacturing,* 13(3), 217–228, 1997.
(SeM.90) - D. Seidel, E. Menzel: A modern concept for the design of machine-tool monitoring and diagnosis systems, *Adv. Manufacturing Engineering,* no 2, 1990, pp. 76–82.
(Shi.88) - M. Shiraishi: Scope of in-process measurement, monitoring and control techniques in machining, t. 1. Tools, *Precision Engineering,* 10(4), 179–189, 1988.
(Shi.89a) - M. Shiraishi: Scope of in-process measurement, monitoring and control techniques in machining, 2: Workpieces, *Precision Engineering,* 11(1), 27–37, 1989.

(Shi.89b) - M. Shiraishi: Scope of in-process measurement, monitoring and control techniques in machining, 3: Processes, *Precision Engineering,* 11(1), 39–47, 1989.
(She.90) - M.J. Shensa: The discrete wavelet transform: wedding the *à trous* and Mallat algorithms, *IEEE Trans. Acoustic, Speech and Signal Proc.,* 1990.
(ShN.97) - Y. Shao, V. Nezu: Monitoring and diagnosis of bearing using laser sensor, *Proc. XIV IMEKO World Congress,* Tampere, Finland, 1–6 June, 1997.
(SKS.97) - K.H. Shin, K. Kobayashi, A. Suzuki: Tafel-Musik: formatting algorithm of tables, *Math. Computer Modelling,* 26(1), 97–112, 1997.
(SMG.97) - A. Seeliger, J. Mackel, D. Georges: Measurement and diagnosis of process-disturbing oscillations in high-speed rolling plants, *Proc. XIV IMEKO World Congress,* Tampere, Finland, 1–6 June, 1997.
(SMN.73) - T. Sata, K. Matsushima, T. Nagakura, E. Kono: Learning ad recognition of the cutting states by spectrum analysis, *CIRP Annals,* 22(1), 41–42, 1973.
(SoB.97) - E. Sokansky, J. Bilos: Possibilities of diagnostics of mechanical failures by means of current spectra of asynchronous motors, *Proc. XIV IMEKO World Congress,* Tampere, Finland, 1–6 June, 1997.
(Str.89) - G. Strang: Wavelets and dilation equations: a brief introduction, *SIAM Review,* 31(4), 614–627, 1989.
(Sue.97) - D.S. Suen: Application of neural network interval regression method for minimun zone straightness and flatness, *J. Precision Engineering,* 20(3), 196–207, 1997.
(SuM.85) - N.C. Suresh, J.R. Meredith: Quality assurance information systems for factory automation, *Intl. J. Production Research,* 23(3), 479–488, 1985.
(SwM.95) - J.A. Swift, J.H. Mize: Out-of-control pattern recognition and analysis for quality control charts using LISP-based systems, *Computers and Industrial Engineering,* 28(1), 81–91, 1995.
(TeB.94) - R. Teti, P. Buonadonna: Neural network applications for process and tool condition monitoring, *Conf. AI Techniques in Engineering,* Napoli, 1994, pp. 90–98.
(TDR.97) - W. Trier, W. Drews, J. Reibiger: Strategies for using robots in high precision 3D measurements, *Intl. J. Mechatronics,* 7(4), 385–411, 1997.
(Tut.88) - F.B. Tuteur: Wavelet transformations in signal detection, *IEEE Int. Conf. Acoustic, Speech Signal,* NewYork, Apr. 11–14, 1988, pp. 1435–1438.
(UOS.92) - G. Ulosoy, I. Or, N. Soydan: Design and implementation of a maintenance planning and central system, *Intl. J. Production Economics,* 24(3), 263–272, 1992.
(VeH.90) - M. Vetterli, C. Herley: Wavelets and filter banks, *IEEE Int. Conf. Acoustic, Speech Signal,* Albuquerque, Apr. 3–6, 1990, pp. 1723–1726.
(VoS.97) - Volkovas, V. Sukackas: Technical diagnostics of pipelines by the approach of wave interference, *Proc. XIV IMEKO World Congress,* Tampere, Finland, 1–6 June, 1997.
(VoW.97) - G. Vosniakos, J. Wang: A software system frame for planning and operation of quality control in discrete part manufacturing, *J. Computer Integrated Manufacturing Systems,* 10(1), 9–25.
(VRS.97) - G. Vandersteen, Y. Rolain, J. Schoukens: Non-parametric estimation of the frequency response functions of the linear blocks of a Wiener-Hammerstein model, *Automatica,* 33(7), 1351–1355, 1997.
(Wal.92) - G.G. Walter: A sampling theorem for wavelet subspaces, *IEEE Trans. on Information Theory,* 38(2), 881–892, 1992.
(Whi.91) - M.F White: Expert systems for fault diagnosis of machinery, *Measurement,* 9(4), 1991.
(WoO.92) - G.W. Wornell, A.V. Oppenheim, Wavelet-based representations for a class of self-similar signals, with application of fractal modulation, *IEEE Trans. Info. Theory,* Mar. 1992.
(Wor.90) - G.W Wornell: A Karhunen·Loeve-like expansion for 1/f processes via wavelets, *IEEE Trans. Info. Theory,* 36(4), 859–861, 1990.
(WSH.92) - G. Wu, F. Shao, B. Hu: Hierarchical structure of a computer integrated quality management system in a CIM environment, *Computers in Industry,* 20, 177–185, 1992.

(XRB.95) - P. Xu, T. Ramdén, R. Burton, P. Krus, C. Sargent: Feasibility of training a neural network based hydraulic component simulator using the complex method, C.R. Burrows, K.E. Edge, Eds., *Innovation in Fluid Power,* Research Studies Press, 1955.

(Yau.97) - D. Yau: Evaluation and uncertainty analysis of vectorial tolerances, *Intl. J. Precision Engineering,* 2(2), 123–137, 1997.

(YiC.97) - H. Ying, G. Chen: Necessary conditions for some typical fuzzy systems as universal approximator, *Automatica,* 33(7), 1333–1344, 1997.

(YuH.88) - G. Yu, D.V Hutton: Liquid coupled acoustic emission measurement for milling operation, *Proc. XVI NAMRC,* 1988, pp. 403–407.

(YWS.92) - X. Yang, K. Wang, S.A. Shamma: Auditory representation of acoustic signals, *IEEE Trans. Info. Theory,* March 1992.

(ZHL.90) - W. Zettler, J. Huffmann, D. Linden: Applications of compactly supported wavelets to image compression, *SPIE Conf. Image Proc. Alg. and Tech.,* Santa Clara, Feb. 13–15, 1990, pp. 150–160.

(ZhZ.91) - Y. Zang, S. Zafar: Motion-compensated wavelet transform coding for color video compression, *SPIE Conf. Image Proc. Alg. and Tech.,* Boston, Nov. 10–13, 1991, pp. 301–316.

(ZNF.96) - Y.F Zhang, A.Y.C. Nee, J.Y.H. Fuh, K.S. Neo, H.K. Loy: A neural network approach to determine optimal inspection sampling size for CMM, *J. Computer Integrated Manufacturing Systems,* 9(3), 161–169, 1996.

6

Reverse Engineering and Inspection of Machined Parts in Manufacturing Systems

6.1 Introduction
6.2 Objectives and Questions
6.3 Sensing for Inspection and Reverse Engineering
Discrete Event Dynamic Systems • Sensing Strategy • The Dynamic Recursive Context for Finite State Machines
6.4 Sensory Processing
Two-dimensional Image Processing • Visual Observation of States • Deciding Feature Relationships • Constructing the Recursive Relation • Extraction of Depth Information and World Coordinates • Camera Calibration • Depth Estimation Using an Illumination Table
6.5 Sensing to CAD Interface
Contours to Splines • Contours to Machined Features
6.6 System Integration
The Initiation Experiment • The Second Experiment • Running the Experiment • Experimental Results, Automated Inspection
6.7 Summary of Current Developments
Goals and Methodology • Current Developments • Past, Current, and Future Activities
6.8 Integration Efforts
Robotics and Sensing • Computer Aided Design and Manufacturing • VLSI, Uncertainty Modeling, and Languages
6.9 Conclusions
Appendix A
Appendix B
Appendix C

Tarek Sobh
University of Bridgeport

Jonathan Owen
University of Bridgeport

Mohamed Dekhil
Rain Infinity

We discuss a design for inspection and reverse engineering environments. We investigate the use of the dynamic recursive context of discrete event dynamic systems (DRFSM DEDS) to guide and control the active exploration and sensing of mechanical parts for industrial inspection and reverse engineering, and utilize the recursive nature of the parts under consideration. In our work, we construct a sensing to CAD interface for the automatic reconstruction of parts from visual data. This chapter includes results and describes this interface in detail, demonstrating its effectiveness with reverse-engineered machined parts.

6.1 Introduction

Developing frameworks for inspection and reverse-engineering applications is an essential activity in many engineering disciplines. Usually, too much time is spent in designing hardware and software environments in order to be able to attack a specific problem. One of the purposes of this work is to provide a basis for solving a class of inspection and reverse-engineering problems.

CAD/CAM (computer aided design/manufacturing) typically involves the design and manufacture of mechanical parts. The problem of reverse engineering is to take an existing mechanical part as the point of departure and to inspect or produce a design, and perhaps a manufacturing process, for the part. The techniques that we explore can be used for a variety of applications. We use an observer agent to sense the current world environment and make some measurements, then supply relevant information to a control module that will be able to make some design choices that will later affect manufacturing and/or inspection activities. This involves both autonomous and semi-autonomous sensing.

The problem of inspection typically involves using a CAD representation of the item to be inspected, and using it to drive an inspection tool such as the coordinate measuring machine (CMM). An example of this is found in [9]. While the work described there is intended to allow the inspection of complicated sculpted surfaces, we limit ours to an important class of machined parts. Within this class, we hope to reduce the time necessary for inspection by more than tenfold, taking advantage of the part's recursive nature and its feature description.

We use a recursive dynamic strategy for exploring machine parts. A discrete event dynamic system (DEDS) framework is designed for modeling and structuring the sensing and control problems. The dynamic recursive context for finite state machines (DRFSM) is introduced as a new DEDS tool for utilizing the recursive nature of the mechanical parts under consideration. This chapter describes what this means in more detail.

6.2 Objectives and Questions

The objective of this research project is to explore the basis for a consistent software and hardware environment, and a flexible system that is capable of performing a variety of inspection and reverse-engineering activities. In particular, we will concentrate on the adaptive automatic extraction of some properties of the world to be sensed and on the subsequent use of the sensed data for producing reliable descriptions of the sensed environments for manufacturing and/or description refinement purposes. We use an observer agent with some sensing capabilities (vision and touch) to actively gather data (measurements) of mechanical parts. We conjecture that discrete event dynamical systems (DEDS) provide a good base for defining consistent and adaptive control structures for the sensing module of the inspection and reverse-engineering problem. If this is true, then we will be able to answer the following questions:

- What is a suitable algorithm to coordinate sensing, inspection, design, and manufacturing?
- What is a suitable control strategy for sensing the mechanical part?
- Which parts should be implemented in hardware vs. software?
- What are suitable language tools for constructing a reverse-engineering and/or inspection strategy?

DEDS can be simply described as: dynamic systems (typically asynchronous) in which state transitions are triggered by discrete events in the system.

It is possible to *control* and *observe* hybrid systems (systems that involve continuous, discrete, and symbolic parameters) under uncertainty using DEDS formulations [13, 16].

The applications of this work are numerous: automatic inspection of mechanical or electronic components and reproduction of mechanical parts. Moreover, the experience gained in performing this research will allow us to study the subdivision of the solution into reliable, reversible, and easy-to-modify software and hardware environments.

6.3 Sensing for Inspection and Reverse Engineering

This section describes the solution methodology for the sensing module and discusses the components separately. The control flow is described and the methods, specific equipment, and procedures are also discussed in detail.

We use a vision sensor (B/W CCD camera) and a coordinate measuring machine (CMM) with the necessary software interfaces to a Sun Sparcstation as the sensing devices. The object is to be inspected by the cooperation of the observer camera and the probing CMM. A DEDS is used as the high-level framework for exploring the mechanical part. A dynamic recursive context for finite state machines (DRFSM) is used to exploit the recursive nature of the parts under consideration.

Discrete Event Dynamic Systems

DEDS are usually modeled by finite state automata with partially observable events, together with a mechanism for enabling and disabling a subset of state transitions [3, 12, 13]. We propose that this model is a suitable framework for many reverse-engineering tasks. In particular, we use the model as a high-level structuring technique for our system.

We advocate an approach in which a stabilizable semi-autonomous visual sensing interface would be capable of making decisions about the *state* of the observed machine part and the probe, thus providing both symbolic and parametric descriptions to the reverse-engineering and/or inspection control module. The DEDS-based active sensing interface is discussed in the following section.

Modeling and Constructing an Observer

The tasks that the autonomous observer system executes can be modeled efficiently within a DEDS framework. We use the DEDS model as a high-level structuring technique to preserve and make use of the information we know about the way in which a mechanical part should be explored. The state and event description is associated with different visual cues; for example, appearance of objects, specific 3-D movements and structures, interaction between the touching probe and part, and occlusions. A DEDS observer serves as an intelligent sensing module that utilizes existing information about the tasks and the environment to make informed tracking and correction movements and autonomous decisions regarding the state of the system.

To know the current state of the exploration process, we need to observe the sequence of events occurring in the system and make decisions regarding the state of the automaton. State ambiguities are allowed to occur; however, they are required to be resolvable after a bounded interval of events. The goal will be to make the system a strongly output-stabilizable one and/or construct an observer to satisfy specific task-oriented visual requirements. Many 2-D visual cues for estimating 3-D world behavior can be used. Examples include image motion, shadows, color, and boundary information. The uncertainty in the sensor acquisition procedure and in the image processing mechanisms should be taken into consideration to compute the world uncertainty.

Foveal and peripheral vision strategies could be used for the autonomous "focusing" on relevant aspects of the scene. Pyramid vision approaches and logarithmic sensors could be used to reduce the dimensionality and computational complexity for the scene under consideration.

Error States and Sequences

We can utilize the observer framework for recognizing error states and sequences. The idea behind this recognition task is to be able to report on *visually incorrect* sequences. In particular, if there is a predetermined observer model of a particular inspection task under observation, then it would be useful to determine if

something goes wrong with the exploration actions. The goal of this reporting procedure is to alert the operator or autonomously supply feedback to the inspecting robot so that it can correct its actions. An example of errors in inspection is unexpected occlusions between the observer camera and the inspection environment, or probing the part in a manner that might break the probe. The correct sequences of automata state transitions can be formulated as the set of strings that are *acceptable* by the observer automaton. This set of strings represents precisely the language describing all possible visual task evolution steps.

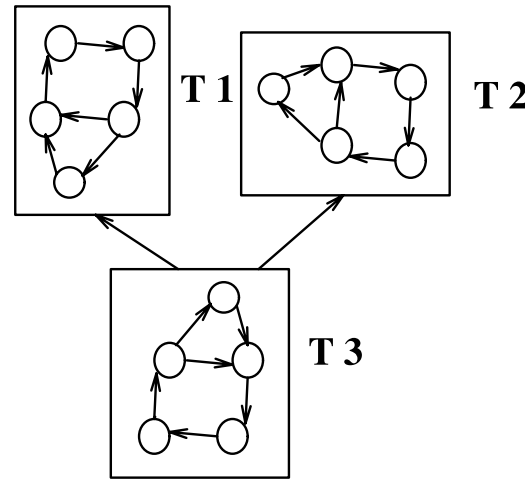


FIGURE 6.1 Hierarchy of tasks.

Hierarchical Representation

Figure 6.1 shows a hierarchy of three submodels. Motives behind establishing hierarchies in the DEDS modeling of different exploration tasks include reducing the search space of the observer and exhibiting modularity in the controller design. This is done through the designer, who subdivides the task space of the exploring robot into separate submodels that are inherently independent. Key events cause the transfer of the observer control to new submodels within the hierarchical description. Transfer of control through the observer hierarchy of models allows coarse to fine shift of attention in recovering events and asserting state transitions.

Mapping Module

The object of having a mapping module is to dispense with the need for the manual design of DEDS automata for various platform tasks. In particular, we would like to have an off-line module, which is to be supplied with some symbolic description of the task under observation and whose output would be the code for a DEDS automaton that is to be executed as the observer agent. A graphical representation of the mapping module is shown in Fig. 6.2. The problem reduces to figuring out what is an appropriate form for the task description. The error state paradigm motivated regarding this problem as the inverse problem of determining acceptable languages for a specific DEDS observer automaton. In particular, we suggest a skeleton for the mapping module that transforms a collection of input strings into an automaton model.

The idea is to supply the mapping module with a collection of strings that represents possible state transition sequences. The input highly depends on the task under observation, what is considered as relevant states and how coarse the automaton should be. The sequences are input by an operator. It should be obvious that the "garbage-in-garbage-out" principle holds for the construction process; in particular, if the set of input strings is not representative of all possible scene evolutions, then the automaton would be a faulty one. The experience and knowledge that the operator have would influence

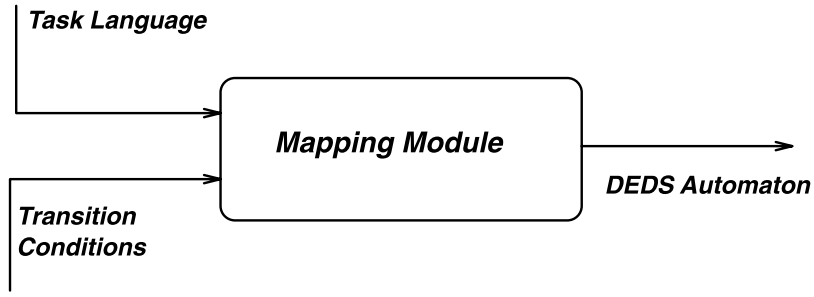


FIGURE 6.2 The mapping module.

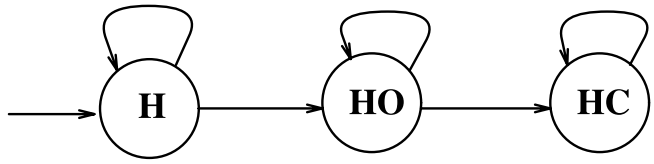


FIGURE 6.3 An automaton for simple grasping.

the outcome of the resulting model. However, it should be noticed that the level of experience needed for providing these sets of strings is much lower than the level of experience needed for a designer to actually construct a DEDS automaton manually. The description of the events that cause transitions between different symbols in the set of strings should be supplied to the module in the form of a list.

As an illustrative example, suppose that the task under consideration is simple grasping of one object and that all we care to know is three configurations: whether the hand is alone in the scene, whether there is an object in addition to the hand, and whether enclosure has occurred. If we represent the configurations by three states h, h_o, and h_c, then the operator would have to supply the mapping module with a list of strings in a language, whose alphabet consists of those three symbols, and those strings should span the entire language, so that the resulting automaton would accept all possible configuration sequences. The mapping from a set of strings in a regular language into a minimal equivalent automaton is a solved problem in automata theory.

One possible language to describe this simple automaton is:

$$L = h^{+} h_o^{+} h_c^{+}$$

and a corresponding DEDS automaton is shown in Fig. 6.3.

The best-case scenario would have been for the operator to supply exactly the language L to the mapping module with the appropriate event definitions. However, it could be the case that the set of strings that the operator supplies does not represent the task language correctly, and in that case some learning techniques would have to be implemented which, in effect, augment the input set of strings into a language that satisfies some predetermined criteria. For example, y^* is substituted for any string of y's having a length greater than n, and so on. In that case, the resulting automaton would be correct up to a certain degree, depending on the operator's experience and the correctness of the learning strategy.

Sensing Strategy

We use a B/W CCD camera mounted on a robot arm and a coordinate measuring machine (CMM) to sense the mechanical part. A DRFSM implementation of a discrete event dynamic system (DEDS) algorithm is used to facilitate the state recovery of the inspection process. DEDS is suitable for modeling robotic observers as it provides a means for tracking the *continuous, discrete*, and *symbolic* aspects of the scene under consideration [3, 12, 13]. Thus, the DEDS controller will be able to *model* and *report* the state evolution of the inspection process.

In inspection, the DEDS guides the sensing machines to the parts of the objects where discrepancies occur between the real object (or a CAD model of it) and the recovered structure data points and/or parameters. The DEDS formulation also compensates for noise in the sensor readings (both ambiguities and uncertainties) using a probabilistic approach for computing the 3-D world parameters [16]. The recovered data from the sensing module is then used to drive the CAD module. The DEDS sensing agent is thus used to collect data of a *passive* element for designing *structures*; an exciting extension is to use a similar DEDS observer for moving agents and subsequently design *behaviors* through a learning stage.

The Dynamic Recursive Context for Finite State Machines

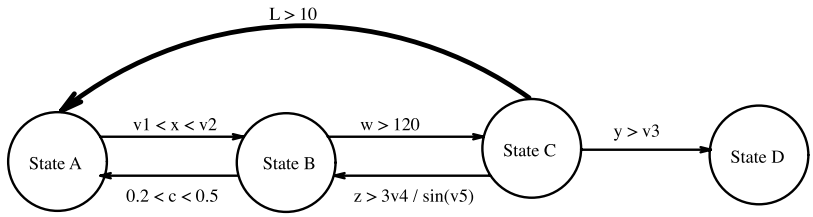
The dynamic recursive context for finite state machines (DRFSM) is a new methodology to represent and implement multi-level recursive processes using systematic implementation techniques. By multi-level process, we mean any processing operations that are done repetitively with different parameters. DRFSM has proved to be a very efficient way to solve many complicated problems in the inspection paradigm using an easy notation and a straightforward implementation, especially for objects that have similar multi-level structures with different parameters. The main idea of the DRFSM is to reuse the conventional DEDS finite state machine for a new level after changing some of the transition parameters. After exploring this level, it will retake its old parameters and continue exploring the previous levels. The implementation of such machines can be generated automatically by some modification to an existing reactive behavior design tool, called GIJoe [4], that is capable of producing code from state machine descriptions (drawings) by adding a recursive representation to the conventional representation of finite state machines, and then generating the appropriate code for it.

Definitions

Variable transition value: Any variable value that depends on the level of recursion.
Variable transition vector: The vector containing all variable transitions values, and is dynamically changed from level to level.
Recursive state: A state calling another state recursively, and this state is responsible for changing the variable transition vector to its new value according to the new level.
Dead-end state: A state that does not call any other state (no transition arrows come out of it). In DRFSM, when this state is reached, it means to go back to a previous level, or quit if it is the first level. This state is usually called the error-trapping state. It is desirable to have several dead-end states to represent different types of errors that can happen in the system.

DRFSM Representation

We will use the same notation and terms of the ordinary FSMs, but some new notation to represent recursive states and variable transitions. First, we permit a new type of transition, as shown in Fig. 6.4, (from state C to A); this is called the recursive transition (RT). A recursive transition arrow (RTA) from one state to another means that the transition from the first state to the second state is done by a recursive call to the second one after changing the variable transition vector. Second, the transition condition



trans. Variables	V1	V2	V3	V4	V5
Level 1	12	15	0.03	170	25
Level 2	10	12	0.07	100	35
Level 3	6	8	0.15	50	40

FIGURE 6.4 A simple DRFSM.

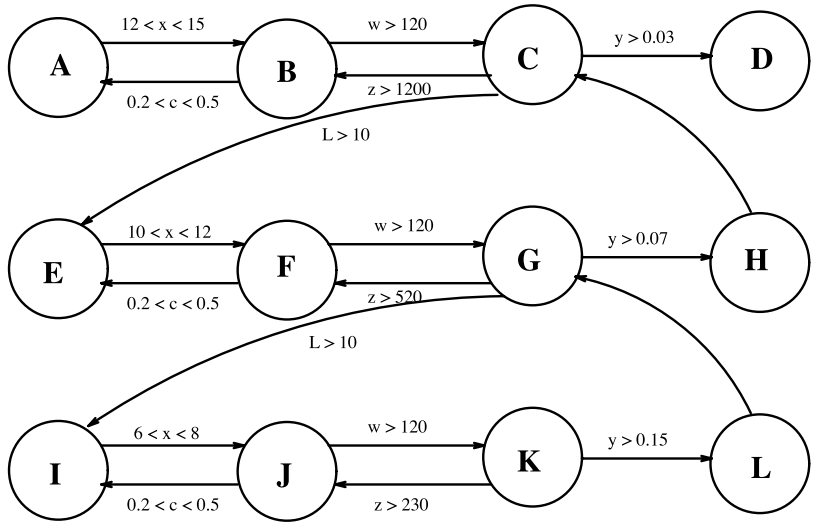


FIGURE 6.5 Flat representation of a simple DRFSM.

from a state to another may contain variable parameters according to the current level; these variable parameters are distinguished from the constant parameters by the notation V (parameter name). All variable parameters of all state transitions constitute the variable transition vector. It should be noticed that nondeterminism is not allowed, in the sense that it is impossible for two concurrent transitions to occur from the same state. Figure 6.5 is the equivalent FSM representation (or the flat representation) of the DRFSM shown in Fig. 6.4, for three levels, and it illustrates the compactness and efficiency of the new notation for this type of process.

A Graphical Interface for Developing DRFSMs

In developing the framework for reverse engineering, it has proven desirable to have a quick and easy means of modifying the DRFSM that drives the inspection process. This was accomplished by modifying an existing reactive behavior design tool, GIJoe, to accommodate producing the code of DRFSM DEDS.

GIJoe [4] allows the user to graphically draw finite state machines, and output the results as C code. GIJoe's original method was to parse transition strings using lex/yacc-generated code. The user interface is quite intuitive, allowing the user to place states with the left mouse button, and transitions by selecting the start and end states with left and right mouse buttons. When the state machine is complete, the user selects a state to be the start state and clicks the Compile button to output C code.

The code output by the original GIJoe has an iterative structure that is not conducive to the recursive formulation of dynamic recursive finite state machines. Therefore, it was decided to modify GIJoe to suit our needs. Modifications to GIJoe include:

- Output of recursive rather than iterative code to allow recursive state machines
- Modification of string parsing to accept recursive transition specification
- Encoding of an event parser to prioritize incoming events from multiple sources
- Implementation of the variable transition vector (VTV) acquisition (when making recursive transitions)

Example code from the machine in Fig. 6.6 can be found in Appendix A. We used this machine in our new experiment, which will be mentioned in a later section.

The event parser was encoded to ensure that the automaton makes transitions on only one source of input. Currently acceptable events are as follows:

- Probe: probe is in the scene
- NoProbe: no probe is in the scene

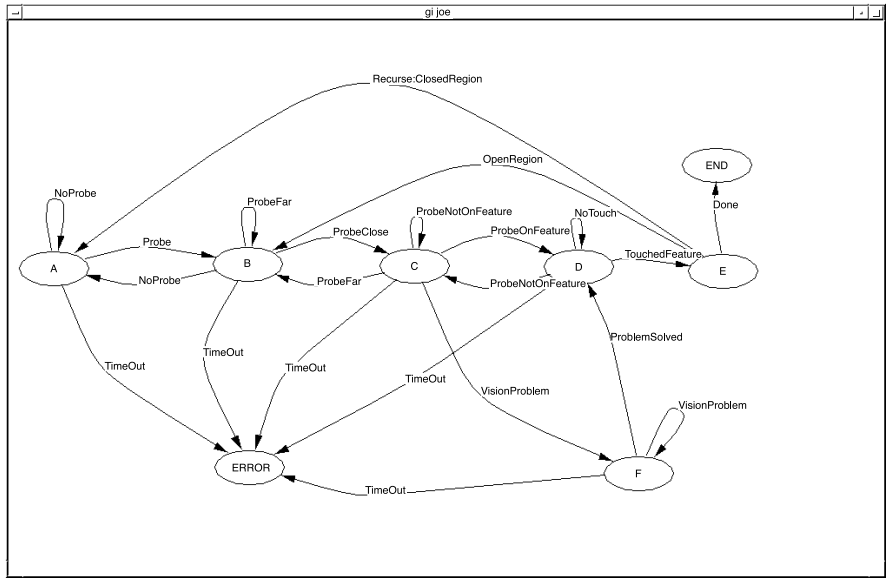


FIGURE 6.6 GIJoe window w/DRFSM.

- ProbeClose: probe is within the "close" tolerance to the current feature specified by the VTV
- ProbeFar: probe is farther from the current feature than the "close" tolerance specified by the VTV
- ProbeOnFeature: probe is on the feature (according to vision)
- ProbeNotOnFeature: probe is close, but not on the feature (according to vision.)
- VisionProblem: part string has changed, signifying that a feature is occluded (need to move the camera)
- ProblemSolved: moving the camera has corrected the occlusion problem
- TouchedFeature: probe has touched the feature (according to touch sensor)
- NoTouch: probe has not touched the feature (according to touch sensor)
- ClosedRegion: current feature contains closed region(s) to be inspected (recursively)
- OpenRegion: current feature contains open region(s) to be inspected (iteratively)
- TimeOut: machine has not changed state within a period of time specified by the VTV
- Done: inspection of the current feature and its children is complete; return to previous level

Additional events require the addition of suitable event handlers. New states and transitions may be added completely within the GIJoe interface. The new code is output from GIJoe and may be linked to the inspection utilities with no modifications.

The VTV, or variable transition vector, is a vector containing variables that may be dependent on the current depth of recursion. It is currently read from a file.

The code produced by the machine in Fig. 6.6 was first tested using a text interface before being linked with the rest of the experimental code. The following is a transcript showing the simulated exploration of two closed regions A and B, with A containing B:

```
inspect [5] ~/DEDS => bin/test_drfsm
   enter the string: A(B())
A(B())

                THE VARIABLE TRANSITION VECTOR

        100.000000          50.000000
in state A
```

```
   has the probe appeared? n                          % NoProbe is true
   has the probe appeared? n
   has the probe appeared? y                          % Probe is true
in state B
   has the probe appeared? y
   enter the distance from probe to A: 85             % Probe is Far
   has the probe appeared? y
   enter the distance from probe to A: 45             % Probe is Close
in state C
   enter the string: A (B())
   enter the distance from probe to A: 10
   is the probe on A? y
in state D
   is the probe on A? y                               % Probe on Feature
   has touch occurred? y
in state E
Making recursive call...

                THE VARIABLE TRANSITION VECTOR

        100.000000          50.000000

in state A
   has the probe appeared? y                          % Probe is true
in state B
   has the probe appeared? y
   enter the distance from probe to B: 95             % Probe is Far
   has the probe appeared? y
   enter the distance from probe to B: 45             % Probe is Close
in state C
   enter the string: A(B())
   enter the distance from probe to B: 10
    is the probe on B? y
in state D
   is the probe on B? y                               % Probe on Feature
   has touch occurred? y
in state E
in state END
in state END

Inspection Complete.

inspect[6] ~/DEDS =>
```

The obtained results, when linked with the rest of the experimental code, were as expected. Future modifications may include the addition of "output" on transitions, such as "TouchOccurred/Update-Model," allowing easy specification of communication between modules. It should be clear, however, that the code generated by GIJoe is only a skeleton for the machine, and has to be filled by the users according to the tasks assigned to each state.

In general, GIJoe proved to be a very efficient and handy tool for generating and modifying such machines. By automating code generation, one can reconfigure the whole inspection process without being familiar with the underlying code (given that all required user-defined events and modules are available).

How to Use DRFSM

To apply DRFSM to any problem, the following steps are required:

1. Problem analysis: divide the problem into states, so that each state accomplishes a simple task.
2. Transition conditions: find the transition conditions between the states.
3. Explore the repetitive part in the problem (recursive property) and specify the recursive states. However, some problems may not have this property; in those cases, a FSM is a better solution.
4. VTV formation: if there are different transition values for each level, these variables must be defined.
5. Error trapping: using robust analysis, a set of possible errors can be established; then, one or more dead-end state(s) are added.
6. DRFSM design: use GIJoe to draw the DRFSM and generate the corresponding C code.
7. Implementation: the code generated by GIJoe has to be filled out with the exact task of each state, the error handling routines should be written, and the required output must be implemented as well.

Applying DRFSM in Features Extraction

An experiment was performed for inspecting a mechanical part using a camera and a probe. A predefined DRFSM state machine was used as the observer agent skeleton. The camera was mounted on a PUMA 560 robot arm so that the part was always in view. The probe could then extend into the field of view and come into contact with the part, as shown in Fig. 6.19.

Symbolic representation of features: for this problem, we are concerned with open regions (O) and closed regions (C). Any closed region may contain other features (the recursive property). Using parenthesis notation, the syntax for representing features can be written as follows:

$$< \text{feature} > :: \text{C}(< \text{subfeature} >) \,|\text{C}()$$
$$< \text{subfeature} > :: < \text{term} >, < \text{subfeature} > \,| < \text{term} >$$
$$< \text{term} > :: \text{O} \,| < \text{feature} >$$

For example, the symbolic notation of Fig. 6.7 is:

$$\text{C(O,C(O,C(),C(O)),C())}$$

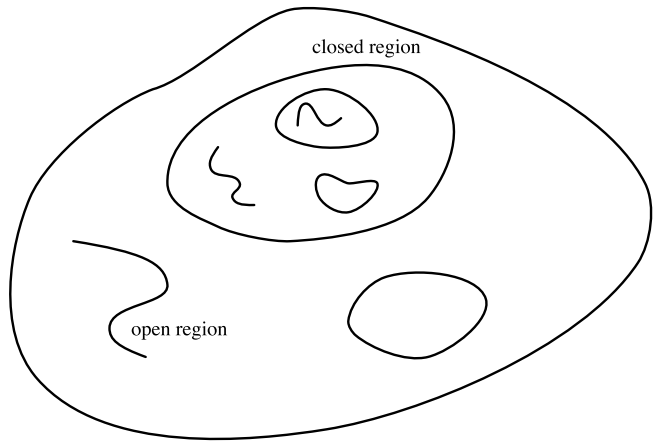


FIGURE 6.7 An example for a recursive object.

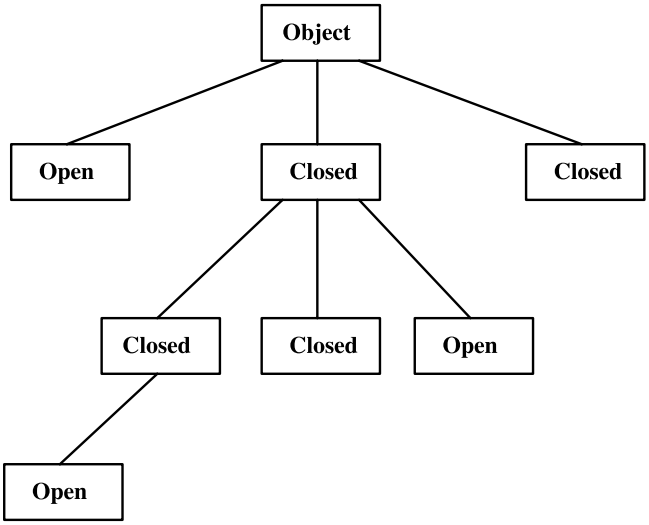


FIGURE 6.8 Graph for the recursive object.

Figure 6.8 shows the graphical representation of this recursive structure which is a tree-like structure. Future modifications to DRFSMs includes allowing different functions for each level.

6.4 Sensory Processing

For the state machine to work, it must be aware of state changes in the system. As inspection takes place, the camera supplies images that are interpreted by a vision processor and used to drive the DRFSM.

A B/W CCD camera is mounted on the end effector of a Puma 560 robot arm. The robot is then able to position the camera in the workplace, take stereo images, and move in the case of occlusion problems. The part to be inspected is placed on the coordinate measuring machine (CMM) table. The probe then explores the part while the mobile camera is able to observe the inspection and provide scene information to the state machine.

The vision system provides the machine with specific information about the current state of the inspection. This is done through several layers of vision processing and through the cooperation of 2-D, $2\frac{1}{2}$-D, and 3-D vision processors.

The aspects of the image that need to be given to the state machine include:

- Number of features
- Contour representation of each feature
- Relationships of the features
- Depth of features
- Approximate depth estimates between features
- Location of the probe with respect to the part

Two-dimensional Image Processing

Two-dimensional (2-D) features of the part are extracted using several different visual image filters. The camera captures the current image and passes it to the 2-D image processing layer. After the appropriate processing has taken place, important information about the scene is supplied to the other vision layers and the state machine.

The images are captured using a B/W CCD camera, which supplies 640 × 480 pixels to a VideoPix video card in a Sun Workstation. The 2-D processing is intended to supply a quick look at the current state of the inspection and only needs a single image, captured without movement of the Puma.

The images are copied from the video card buffer and then processed. The main goal of image processing modules is to discover features. Once features are discovered, contours are searched for among the feature responses.

Extracting Contours

Contours are considered important features that supply the machine with information necessary to build an object model and drive the actual inspection. There are two types of contours, each with specific properties and uses, in the experiment:

1. Open contour: a feature of the part, like an edge or ridge that does not form a 'closed' region. Lighting anomalies may also cause an open contour to be discovered.
2. Closed contour: a part or image feature that forms a closed region; that is, it can be followed from a specific point on the feature back to itself. A typical closed contour is a hole or the part boundary.

We are concerned with finding as many "real" contours as possible while ignoring the "false" contours. A real contour would be an actual feature of the part, while a false contour is attributed to other factors such as lighting problems (shadows, reflections) or occlusion (the probe detected as a part feature).

If we are unable to supply the machine with relatively few false contours and a majority of real contours, the actual inspection will take longer. The machine will waste time inspecting shadows, reflections, etc.

We avoid many of these problems by carefully controlling the lighting conditions of the experiment. The static environment of the manufacturing workspace allows us to provide a diffuse light source at a chosen intensity. However, simple control of the lighting is not enough. We must apply several preprocessing steps to the images before searching for contours, including:

1. Threshold the image to extract the known probe intensities.
2. Calculate the Laplacian of the Gaussian.
3. Calculate the zero-crossings of the second directional derivative.
4. Follow the "strong" zero-crossing edge responses to discover contours.

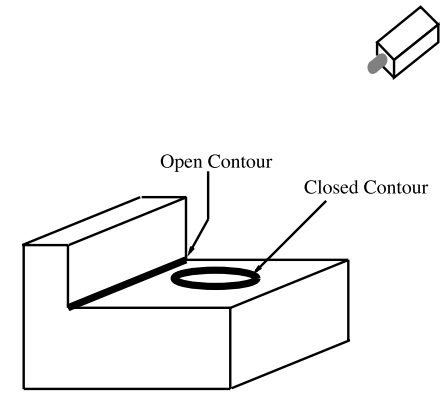


FIGURE 6.9 Edge finding in the two-dimensional image can give hints about where to look for three-dimensional features. The open contour here is generated where two faces meet. The corresponding contour is then explored by both the stereo layer and the CMM machine.

Zero-crossings

The Marr-Hildreth operator [11] is used to find areas where the gray-level intensities are changing rapidly. This is a derivative operator, which is simply

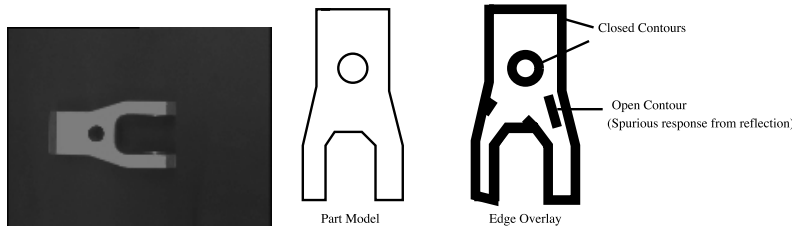


FIGURE 6.10 A contour discovery example.

the thresholded image convolved with the Laplacian of a Gaussian. The operator is given by:

$$\Delta^2 G(x, y) = \frac{1}{\sigma} \frac{x^2 + y^2}{\sigma^2} - 2e^{-\frac{x^2 + y^2}{2\sigma^2}}$$

where σ is a constant that scales the Gaussian blur over the image. For large numbers, σ acts as a low-pass filter. Smaller values retain more localized features but produce results that are more susceptible to noise. This scale can be related to an image window by:

$$\sigma = \frac{w}{2\sqrt{2}}$$

where w is the actual window size in pixels. On average, we trade more accuracy for noise and rely on a robust edge follower and the intrinsic properties of contours to eliminate noise.

The zero-crossing operator calculates orientation, magnitude, and pixel location of all edge responses. This is helpful for the contour following algorithm that uses all three pieces of information.

Contour Properties

An edge response is only considered to be a contour if it satisfies two conditions: (1) each response must exceed a previously specified minimum value, and (2) the length of each edge must exceed a previously specified minimum pixel count.

Edges are followed iteratively. An edge is followed until its response falls below the minimum or we arrive at our starting position, in which case the contour is known to be closed. If a branch in the contour is encountered, the branch location is saved and following continues. We attempt to follow all branches looking for a closed contour. Branches are considered to be part of a contour because they may represent an actual feature of the part (a crack extending from a hole, for example) and should be inspected.

Once the region contours are found, they can be used in the stereo vision correspondence problem for model construction. They are also given to the machine to help drive the actual inspection process. Some closed contours and the image in which they were found are seen in Fig. 6.11.

Visual Observation of States

The visual processor supplies the proper input signals to the DRFSM DEDS as the inspection takes place. These signals are dependent on the state of the scene and are triggered by discrete events that are observed by the camera.

The visual processor layer is made up of several filters that are applied to each image as it is captured. Several things must be determined about the scene before a signal is produced: the location of the part,

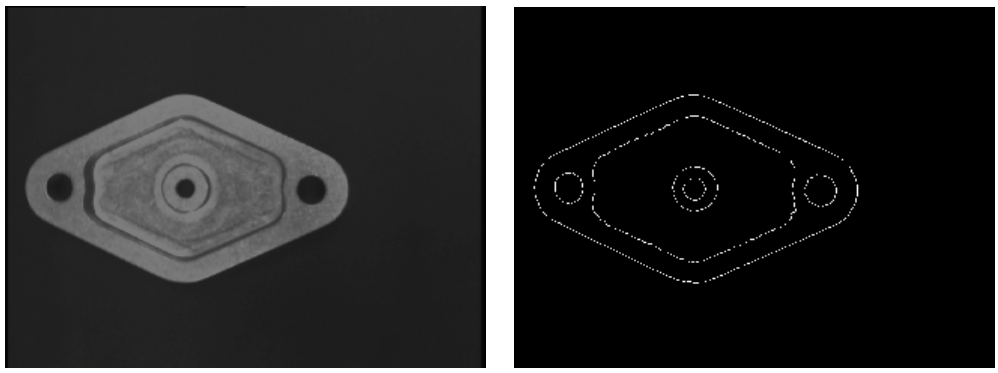


FIGURE 6.11 An image and its contours.

the location of the probe, the distance between them, the number of features on the part, and the distance to the closest feature.

First, the image is thresholded at a gray-level that optimizes the loss of background while retaining the part and probe. Next, a median filter is applied that removes small regions of noise. The image is then parsed to find all segments separated by an appropriate distance and labels them with a unique region identifier.

We are able to assume that the probe, if in the scene, will always intersect the image border. The probe tip is the farthest point on the probe region from the border. This holds true because of the geometry of the probe. An image with one region that intersects the border is the case in which the probe is touching the part.

If we have more than one region, we must discover the distance between the tip of the probe region and the part. This is done through an edge following algorithm that gives us the x, y positions of the pixels on the edge of each region. We then find the Euclidean distances between the edge points and the probe tip. The closest point found is used in producing the signal to the state machine.

Once this information is known, we are able to supply the correct signal that will drive the DRFSM DEDS. The machine will then switch states appropriately and wait for the next valid signal. This process is a recursive one, in that the machine will be applied recursively to the closed features. As the probe enters a closed region, another machine will be activated that will inspect the smaller closed region with the same strategy that was used on the enclosing region.

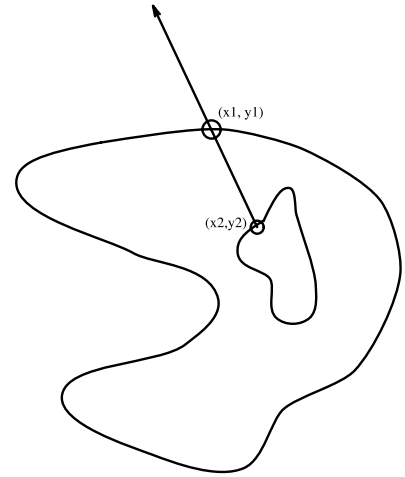


FIGURE 6.12 A closed region within another.

Deciding Feature Relationships

Having found all of the features, we now search for the relationships between them. In the final representation of intrinsic information about the part, it is important to know which feature lies "within" another closed feature.

Consider a scene with two features, a part with an external boundary and a single hole. We would like to represent this scene with the string: "C(C())". This can be interpreted as a closed region within another closed region.

To discover if feature F_2 is contained within F_1 given that we know F_1 is a closed feature, we select a point (x_2, y_2) on F_2 and another point (x_1, y_1) on F_1. Now, we project the ray that begins at (x_2, y_2) and passes through (x_1, y_1). We count the number of times that this ray intersects with F_1. If this is odd, then we can say F_2 is contained within F_1; otherwise it must lie outside of F_1 (see Figs. 6.12 and 6.13)

This algorithm will hold true as long as the ray is not tangential at the point (x_1, y_1) of feature F_1. To avoid this case, we simply generate two rays that pass through (x_2, y_2) and a neighboring pixel on F_1. If either of these have an odd number of intersections, then F_2 is contained in feature F_1.

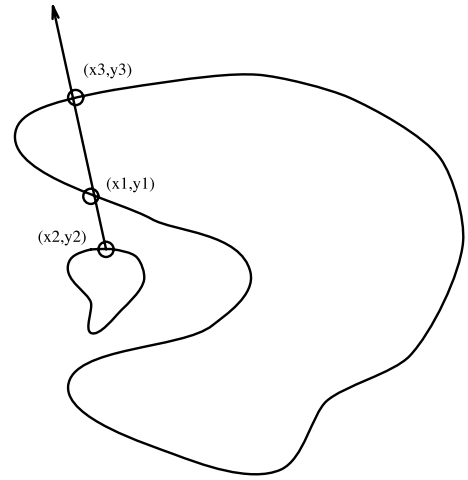


FIGURE 6.13 A closed region outside another.

An alternate method was also used to determine whether a region is inside another. A point on the contour to be checked was grown. If the grown region hit the frame, that would imply that the region is not contained; otherwise, it would be contained inside the bigger contour, and the grown region would be all the area within the bigger contour.

Knowing what features are present in the part and their relationships with each other will allow us to report the information in a string that is sent to the state machine. This process will be explained in detail in the next section.

Constructing the Recursive Relation

One of the problems encountered in this experiment was converting the set of relations between closed regions to the proposed syntax for describing objects. For example, the syntax of Fig. 6.14 is:

$$C(C(C(),C()),C())$$

and the relations generated by the image processing program are:

$$\begin{aligned} B \in A &\rightarrow (1)\\ C \in A &\rightarrow (2)\\ D \in B &\rightarrow (3)\\ D \in A &\rightarrow (4)\\ E \in B &\rightarrow (5)\\ E \in A &\rightarrow (6) \end{aligned}$$

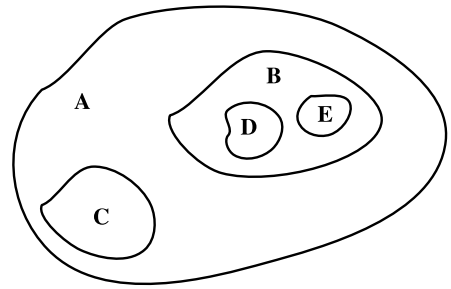


FIGURE 6.14 A hierarchy example.

These relations can be represented by a graph as shown in Fig. 6.15. The target is to convert this graph to an equivalent tree structure, which is the most convenient data structure to represent our syntax.

As a first attempt, we designed an algorithm to convert from graph representation to tree representation by scanning all possible paths in the graph and putting weights to each node according to number of visits to this node. In other words, update the depth variable of each node by traversing the tree in all possible ways and then assign the nodes the maximum depth registered from a traversal, and propagate that depth downward. Then, from these depth weights, we can remove the unnecessary arcs from the graph by keeping only the arcs that have a relation between a parent of *maximum* depth and a child, and eliminating all other parent arcs, thus yielding the required tree (Fig. 6.16).

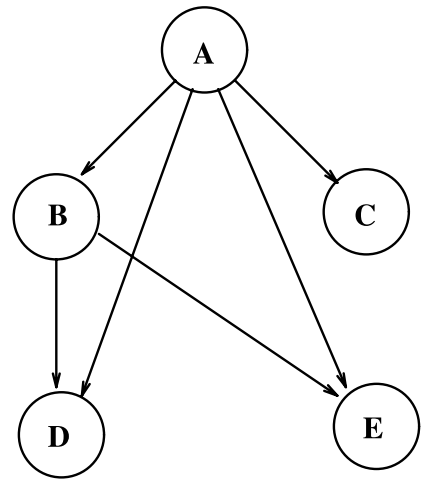


FIGURE 6.15 The graph associated with the example in Fig. 6.14.

However, we have developed a better algorithm that scans the relations; counts the number of occurrences for each closed region name mentioned in the left side of the relations, giving an array RANK(x), where $x \in$ {A,B,C,…}; and selects the relations ($x_1 \in x_2$) that satisfy the following condition:

$$\text{RANK}(x_1) - \text{RANK}(x_2) = 1$$

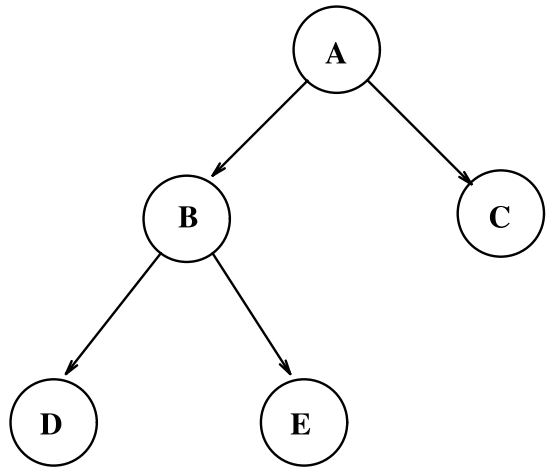


FIGURE 6.16 The tree associated with the example in Fig. 6.14.

This guarantees that all redundant relations will not be selected. The complexity of this algorithm is $O(n)$, where n is the number of relations. Applying this algorithm to the relations of Fig. 6.14, we obtain:

$$\begin{aligned} \text{RANK(A)} &= 0 \\ \text{RANK(B)} &= 1 \\ \text{RANK(C)} &= 1 \\ \text{RANK(D)} &= 2 \\ \text{RANK(E)} &= 2 \end{aligned}$$

The selected relations will be:

$$\begin{aligned} &\text{B} \in \text{A} \\ &\text{C} \in \text{A} \\ &\text{D} \in \text{B} \\ &\text{E} \in \text{B} \end{aligned}$$

Now, arranging these relations to construct the syntax gives:

$$\text{A(B())} \rightarrow \text{A(B(),C())} \rightarrow \text{A(B(D()),C())} \rightarrow \text{A(B(D(),E()),C())}$$

which is the required syntax. A tree representing this syntax is easily constructed and shown in Fig. 6.16. The next step would be to insert the open regions, if any, and this is done by traversing the tree from the maximum depth and upward. Any open region can be tested by checking any point in it to see whether it lies within the maximum-depth leaves of the closed regions' tree hierarchy (the test is easily done by extending a line and checking how many times it intersects a closed region, as in the test for closed regions enclosures). Then, the upper levels of the hierarchy are tested in ascending order until the root is reached or all open regions have been exhausted. Any open region found to be inside a closed one while traversing the tree is inserted in the tree as a son for that closed region. It should be noticed that this algorithm is *not* a general graph-to-tree conversion algorithm; it only works on the specific kind of graphs that the image processing module recovers. That is, the conversion algorithm is *tailored* to the visual recursion paradigm.

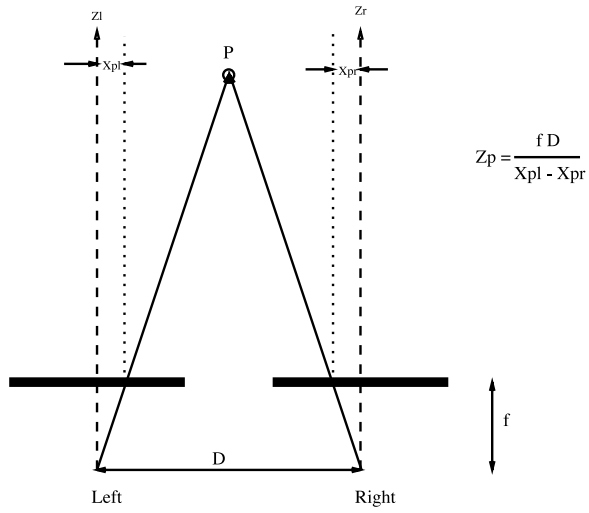


FIGURE 6.17 Pinhole camera model.

Extraction of Depth Information and World Coordinates

A crude initial model is found using stereo vision. The camera model used is a pinhole camera as shown in Fig. 6.17, corrected for radial distortion. Depths are found with such models using the disparity between feature locations in a pair of views according to the following formula:

$$Z = fD/(x_l - x_r)$$

where x_l and x_r are coordinates on the image plane, f is the focal length, and D is the disparity. Additional aspects of the camera model are discussed in the section on camera calibration.

The stereo algorithm currently in use requires no correspondence between individual features in the stereo image pair [1]. Instead, corresponding regions are found, and the disparity in their centers of mass is used for depth computation. In our experiment, closed regions are found in two images and their relationships are determined. Each closed region is described by its boundary, a contour in image coordinates. It is assumed for the initial model that these contours are planar. Given this assumption, the parameters p, q, and c of a plane must be solved for in the equation

$$Z = pX + qY + c$$

To do this, each region is split into three similar sections in both left and right images. The center of mass is computed for each section, and the system of equations solved for p, q, and c. These values are stored with the region for later output of the CAD model (we use the α_1 CAD package). It should be noted that if the centers of mass are collinear, this system will not be solvable (three non-collinear points define a plane). Also, if the centers of mass are close together, the error in discretization will cause substantial error in computation of plane parameters. In other words, if the three points are close together, an error of one pixel will cause a substantial error in the computed orientation of the plane. The effect of a one-pixel error is reduced by selecting points that are "far" apart. Thus, the technique used to split regions, determining the locations of these points, is a crucial part of the algorithm.

The most obvious and perhaps simplest technique splits contours by dividing them into three parts horizontally (see Fig. 6.18.) Because many machined features (such as holes) will produce collinear centers of mass when partitioned this way, a different technique is used. It is attempted to divide each contour into three parts of equal length (see Fig. 6.18). One region may partitioned purely by

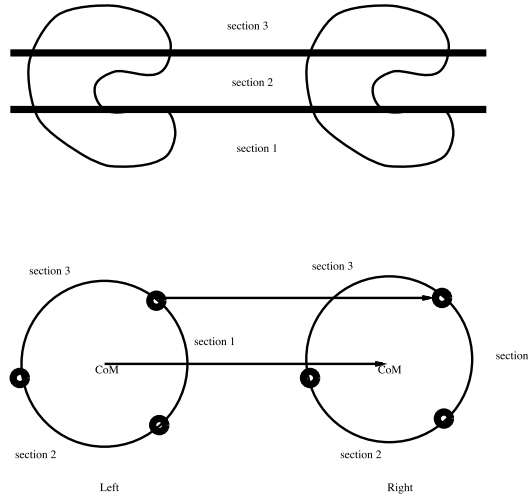


FIGURE 6.18 Region splitting algorithms.

length, but to partition the other exactly would require solution of the correspondence problem. Fortunately, the effects of error in correspondence are made minimal when averaged over a section. The first pixel in the left image's region is matched with one in the right image by translating it along a vector between the centers of mass of the regions and finding the closest pixel to this position in the right image.

In practice, this was found to work fairly well for the outermost region. However, using the same technique for smaller inner regions, error is much greater because the three centers of mass used to determine the plane parameters are closer together. A further assumption may be made, however: that the inner regions are in planes parallel to the outer region. Using this assumption, it is not necessary to split the regions into three parts, and the plane equation can be solved for c using the center of mass of the entire region. If it is assumed that all planes are parallel to the table (world x-y plane), the outermost region can be treated in like manner.

For our initial experiment, the following assumptions were made.

- The robot z axis is perpendicular to the table on which the robot is mounted.
- The table is a planar surface, parallel to the floor.
- The CCD plane of the camera is parallel to the back of the camera case.
- All object contours are in planes parallel to the table.

The experimental setup is shown in Fig. 6.19. The camera was oriented with its optical axis approximately perpendicular to the table. This was first done by visual inspection. Then, for more accuracy, a level was used on the back of the camera case and the robot tool was rotated until the camera appeared level. The robot tool frame was then recorded (as Left). This frame was used consistently to capture images for the remainder of the experiment. At that point, the problem had been constrained to finding the angle between robot x and image x. This was necessary because the stereo algorithm is based on disparity only in the image x direction.

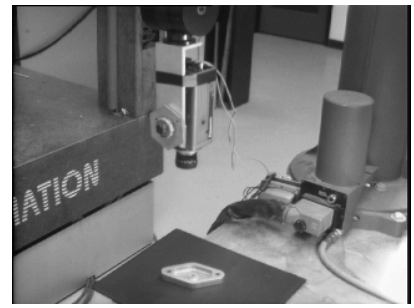


FIGURE 6.19 Experimental setup.

To accomplish the constrained motion, the following algorithm was implemented:

- Move the camera to the recorded frame.
- Take an image.
- Threshold it.
- Compute the center of mass of an object in the scene (there should be only one) in image coordinates.
- Move the camera in the robot-*x* direction.
- Take an image.
- Threshold it.
- Compute the new center of mass of an object in the scene in image coordinates.
- Compute the angle between the vector (new-old) and the image *x*-axis.
- Compute a new frame accounting for this angle and record it.
- Move to the new frame, recompute the center of mass, and display it.

At this point, the rotation part of the transform from camera coordinates to world coordinates is known, and the translational part must be determined. X- and Y-components of the translation are taken to be zero, making the world coordinate origin centered in the left frame image. The Z-component was determined by taking an image pair of a paper cut-out (thickness assumed to be zero). The Z-coordinate of this object should be the distance from the image plane to the table. This was then used to complete the homogeneous transform from camera coordinates to world coordinates:

$$\begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & -1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & -1.0 & 234.1 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

Several stereo image pairs were then captured using the Left and Right frames, and then used by the stereo code to produce α_1 models with the objects described in world coordinates. For a cube measuring 1 inch (25.4 mm) on a side, the resulting α_1 model was similar to a cube (lighting effects are observable), and dimensioned to 26.74 mm × 25.5 mm × 25.7 mm (h × l × w). This corresponds to percent errors of 5.2, 0.4, and 1.2, respectively. Some example images and corresponding models are shown in later sections.

Camera Calibration

Real-world cameras differ substantially from the ideal camera model typically used for discussion of stereo vision. Lens distortion, offsets from the image center, etc. are sources of error in computing range information. The camera calibration technique chosen for this project takes many of these factors into account. Developed by Roger Tsai, and implemented by Reg Willson of CMU [18, 20], this technique has been described as:

- Accurate
- Reasonably efficient
- Versatile
- Needing only off-the-shelf cameras and lenses
- Autonomous (requiring no operator control, guesses, etc.)

The technique solves for the following parameters:

- *f*-focal length
- k-lens distortion coefficient

- (Cx, Cy) image center
- Uncertainty scale factor (due to camera timing and acquisition error)
- Rotation matrix
- Translation vector

of which we use the focal length, distortion coefficient, and image center. The equations used are as follows:

$$X_u = f \cdot \frac{x}{z}$$

$$Y_u = f \cdot \frac{y}{z}$$

$$X_u = X_d \cdot (1 + k \cdot (X_d^2 + Y_d^2))$$
$$Y_u = Y_d \cdot (1 + k \cdot (X_d^2 + Y_d^2))$$

$$X_d = dx \cdot (X_f - C_x)$$
$$Y_d = dy \cdot (Y_f - C_y)$$

where dx and dy are the center-to-center distances between adjacent sensor elements on the CCD plane in the x and y directions (obtained from Panasonic); X_u and Y_u are undistorted image plane coordinates; x, y, and z are in the camera coordinate system; and X_d and Y_d are distorted image plane coordinates. The effects of the distortion coefficient can be seen in Figs. 6.20 to 6.23.

In the classical approach to camera calibration, computing the large number of parameters requires large-scale nonlinear search. In Tsai's method, however, the problem's dimensionality is reduced by using the radial alignment constraint to split the search into two stages [18]. In the first stage, extrinsic parameters such as Translation and Rotation parameters are found. The second solves for the intrinsic parameters (f, k, etc.).

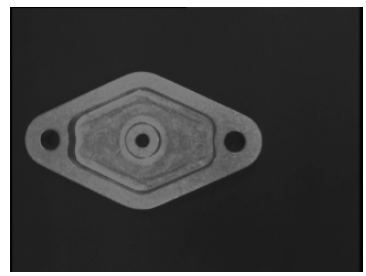


FIGURE 6.20 Raw image.

The implementation used accepts a data file containing points that are known in both image coordinates and world coordinates. For Tsai's original paper, data was obtained from a calibration grid measured with a micrometer and 400X microscope. For our purposes, a paper grid of 1-mm diameter dots spaced 1 cm apart was made using AutoCad and a plotter (see Fig. 6.24). A plotter was used rather than a laser printer in hopes of minimizing such errors as the stretching effect found in the output of worn laser printers. The calibration algorithm is quite sensitive to systematic errors in its input. It should be noted that for Tsai's algorithm, the camera's optical axis should be at an angle greater than 30° from the plane in which the calibration points occur. The calibration data was generated in the following manner:

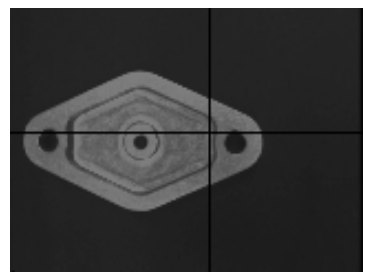


FIGURE 6.21 Corrected image, kappa = .0005.

- Capture calibration image (dot locations are known in world coordinates)
- Threshold calibration image.
- Visit each calibration image "dot" in a systematic way:
  - — Select a pixel interior to the dot.
  - — Compute the center of mass of the 8-connected region.
  - — Output the world *x-y-z*, image *x-y* information to the calibration data file.

For our first experiment, 25 calibration points were used. Typical results are shown in Appendix B.

For future experiments, a more refined calibration data collection system may be used, possibly using the CMM as a tool to generate data points. This will facilitate outputting stereo range information in the CMM's coordinate system.

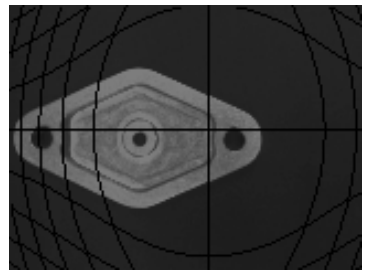


FIGURE 6.22 Corrected image, kappa = .00357 (used in our experiment).

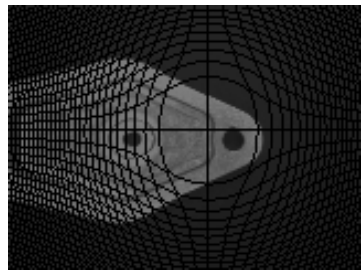


FIGURE 6.23 Corrected image, kappa = .05.

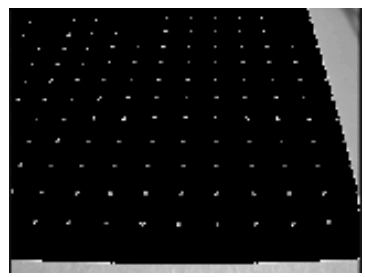


FIGURE 6.24 Thresholded calibration grid.

Depth Estimation Using an Illumination Table

Using stereo techniques for estimating the depths of an object's contours can be very accurate, but it is limited in that it cannot compute the depth of an occluded contour (i.e., the bottom of a hole or pocket). As shown in Fig. 6.25, the algorithm will give the depths for both contours correctly in case A, while in case B the depth of both contours will be the same.

It was attempted to solve this problem using a predefined illumination table that relates the intensity of a point on the object to the distance between this point and the camera. When the stereo algorithm detects two nested contours with the same depth, this table would be used to estimate the depth of the inner region. This method is very simple to implement, but it proved to have some drawbacks. For example, it is very sensitive to the lighting conditions, i.e., any variation in the lighting conditions will result in the invalidation of the lookup table. Also, objects being observed must have consistent surface properties. In the following section, attempts to overcome these problems are described.

Table Construction

This table is constructed off-line before running the experiment. The following assumptions were made:

- The object is formed of the same material, hence the illumination at any point is the same (assuming well distributed light and no shadows).
- The same camera with the same calibration parameters are used during the experiment.
- The lighting conditions will be the same during the experiment.

We can consider these to be valid assumptions because the manufacturing environment is totally controlled, thus, we know the object material and we set the lighting conditions as desired.

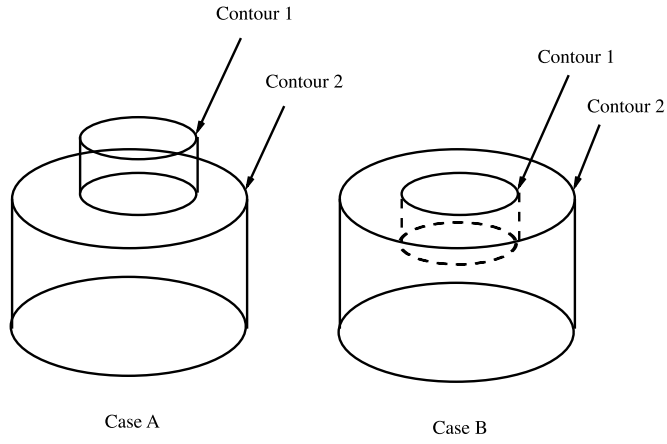


FIGURE 6.25 The problem when using stereo in depth estimate.

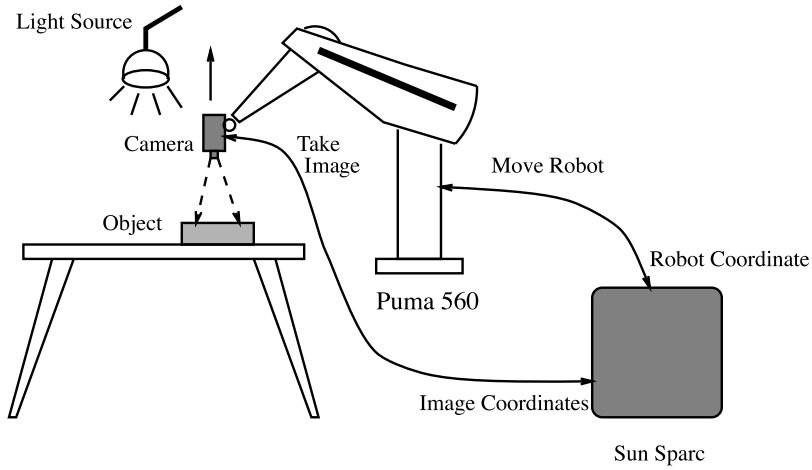


FIGURE 6.26 Constructing the illumination table off-line.

This table will be constructed only once, and will then be used for all our experiments, as long as they satisfy our assumptions. However, if we wish to examine an object with different materials, or we want to change the lighting conditions, we will have to construct a new table using the new object and the new lighting conditions. To construct this table, the robot arm that holds the camera is moved vertically in incremental steps, according to the required accuracy. At each increment, an image is taken and the intensity at the center of the image is measured; see Fig. 6.26 for the experimental setup.

Modifications

The initial implementation of this method did not produce the expected results because of the noise in the images taken at each depth. Several enhancements were added to this method to reduce the effect of noise. First, instead of measuring the intensity at one point, we take the average of the intensities of a set of points that constitutes a rectangular window. By changing the window size, we can control the smoothing degree of the measured intensity. The second enhancement is also based on averaging, by taking several images at each height and taking the average of the calculated average window intensities. After applying these two modifications, the effect of noise was greatly reduced.

Another modification was to move the light source with the camera to increase the difference in the measured intensity at each height, which, in turn, should have increased the resolution of our table.

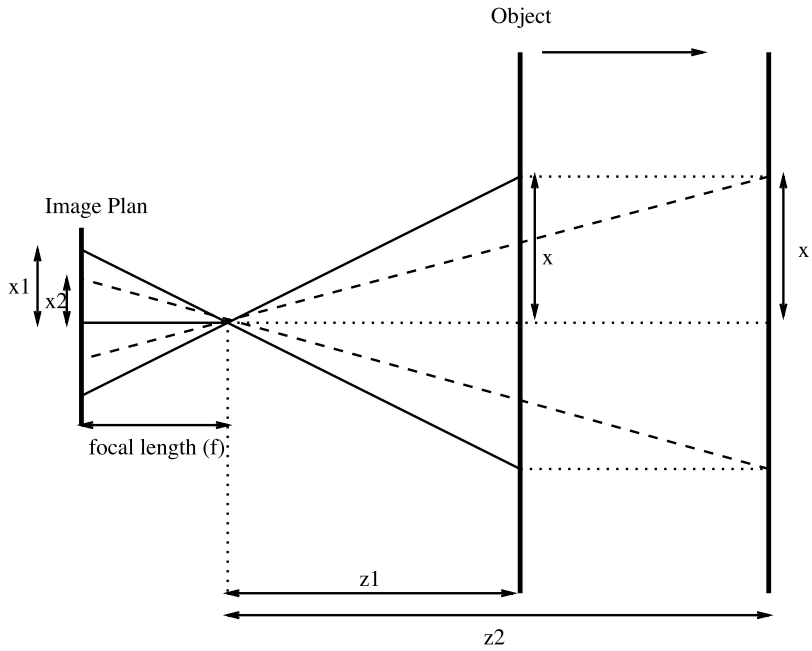


FIGURE 6.27 Changing window size assuming pinhole camera model.

One last enhancement was incorporated based on the perspective-projective transform from world points to image points. The window size used to calculate the average intensity at each height should be the same; but according to the image formation equations using the pinhole camera model, the corresponding window in the image will change according to the distance between the camera (image plane) and the object. From Fig. 6.27, using simple trigonometry, we obtain the following relation between the image window size and the distance z between the object and the camera:

$$\frac{x1}{x2} = \frac{z2}{z1}$$

which shows that the image window size is inversely proportional to the distance between the object and the camera. Thus, we must calculate the new window size at each height, which will be the number of pixels used for averaging.

Figure 6.28 shows a graph for the constructed illumination table used in our experiment. It shows that the intensity decreases when the distance between the object and the camera increases, but it also shows that any change in the lighting condition will give different results for the illumination table.

This method also has some pitfalls. First, it is very sensitive to any light change, as shown in Fig. 6.28. Second, the difference in illumination values for two close depths is very small. For example, in our experiment, the total range of differences within 10 cm was less than 30 gray-levels. Finally, it still has small amounts of noise at some points. We are now developing another method for determining depth from focus. This method involves calculating distances to points in an observed scene by modeling the effect that the camera's focal parameters have on images acquired with a small depth of field [7].

6.5 Sensing to CAD Interface

An important step in the reverse-engineering process is the accurate description of the real-world object. We generate an α_1 model from a combination of three types of scene information:

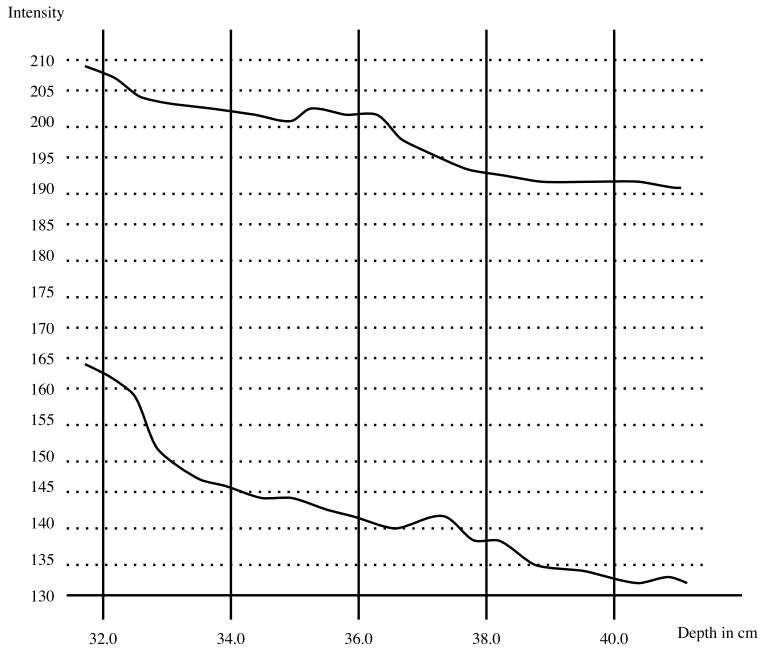


FIGURE 6.28 Two different results when changing the lighting conditions.

- Two-dimensional images and feature contours
- Stereo vision depth information
- Touch information from the CMM (still to be implemented)

Using each sensing method, we are able to gather enough data to construct the accurate CAD model necessary for reverse engineering (see Fig. 6.29.) The two-dimensional images provide feature detection that is in turn used by the stereo system to build feature models. Finally, the CMM eliminates uncertainty through the exploration of the features.

The state machine and the sensors are able to produce a set of data points and the respective enclosure relationships. Each feature is constructed in α_1 independently, and the final model is a combination of these features. This combination is performed using the recursive structure of the object by forming the corresponding string for that object and generating the code by parsing this string recursively. The third dimension is retrieved from the stereo information and the illumination table as described previously. An example for a reconstructed part is shown in Fig. 6.30.

This interface is one of the most important modules in this work, because it is the real link between inspection and reverse engineering. We have chosen α_1 as the CAD language because it has very powerful features, in addition to the fact that it has interfaces with some manufacturing machines, which allows us to actually manufacture a hard copy of the part. This will be our next step, so that the output of our next experiment will be another part, hopefully identical to the original part.

Contours to Splines

In the initial stage of our sensing to CAD interface, we translated the ranged contours we found into spline curves.

Both closed and open contours are represented as ordered sets of points. The contour points are used as control points on a spline curve in the α_1 system. It is important not to use all of the contour points while fitting the spline. In many cases, there are more than a thousand points in the original image. This gives an over-constrained solution.

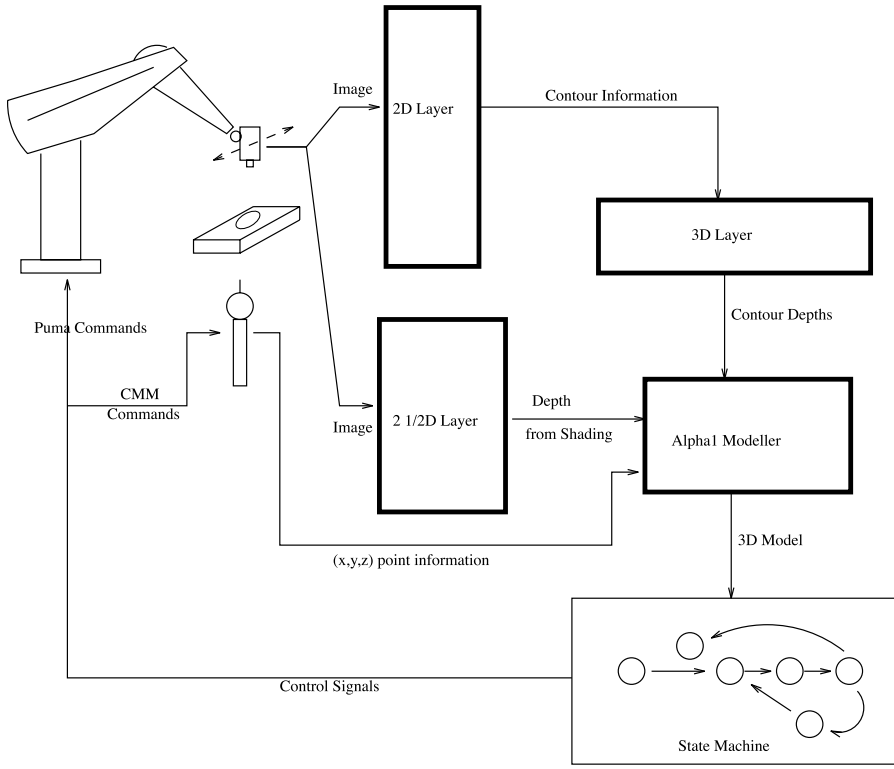


FIGURE 6.29 The role of an internal CAD model in the inspection machine.

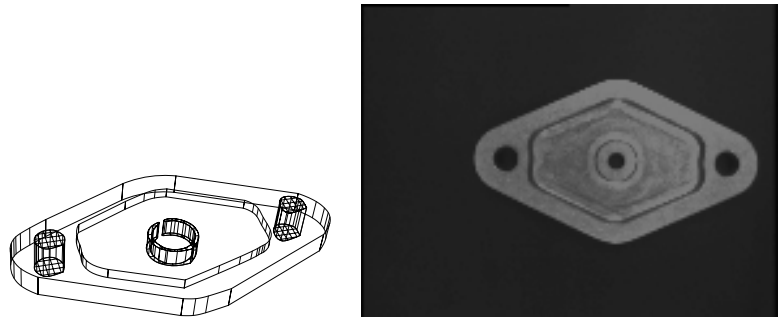


FIGURE 6.30 A rough α_1 surface model extracted from the machine vision.

Thinning Contours

We must supply a list of control points to α_1 that will fit a spline accurately to the original real-world feature. Therefore, we must decide which points contribute to the actual shape of the feature and which points are simply redundant.

Obviously, regions of high curvature are important to the overall shape of the feature, while low curvature regions will not play as important a role. We fit lines to each contour and represent them as polyline segments in α_1. Each line only consists of its endpoints rather than all the image points along its length. All of the line segments and splines that make up a particular contour are combined together using the α_1 profile curve.

An example is shown in Fig. 6.31. The straight lines in this closed contour are found, and the corner points are used as "important" points to the α_1 model. Points along the top arc are all used so that

a spline can be fit to them accurately. The final region is represented as the combination of the lines and curves that makes up its length.

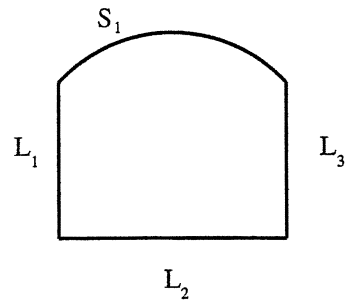


FIGURE 6.31 L_1, L_2, and L_3 are fit to lines before they are used in α_1. S_1 has too much curvature and all of its points are used to describe a piecewise spline.

Contours to Machined Features

Although the lines and splines representation proved useful, another representation was found to be needed to describe our machined parts. A set of machinable features, as implemented in the α_1 modeling system [2], has been selected for this representation. This set includes profileSides (for describing the outside curve), profilePockets (for describing interior features generated by milling), and holes (for describing features generated by drilling). Although not within the scope of current research, the method being described is extensible to include other features, such as slots, bosses, etc. Using this subset, most parts which meet the requirements of the visual processing algorithms can be transformed to a machinable α_1 representation.

The Algorithm

The transformation algorithm is as follows:

- Input: raw images, part string, and ranged contour representation
- Convert part string to tree representation (see Fig. 6.32)
- Generate stock using bounding box (see Fig. 6.33)
- Generate profileSide for outermost contour (see Fig. 6.34)
- Class each subfeature as positive or negative relative to its predecessors (see Fig. 6.35)
- Recursively descend the part tree, starting with the outermost contour's children:
  - If negative, check for positive subfeatures relative to it (if there are none, produce a hole or profile pocket depending on curvature; otherwise, there must be a profile pocket with an island; produce both)
  - Otherwise, check to see if this island needs to be trimmed
- Output: α_1 model composed of machinable features

Note that this algorithm assumes that the outermost contour is the highest. This limitation can be overcome by a simple check at the start, and subsequent treatment of the outer contour as a feature within a blockStock feature.

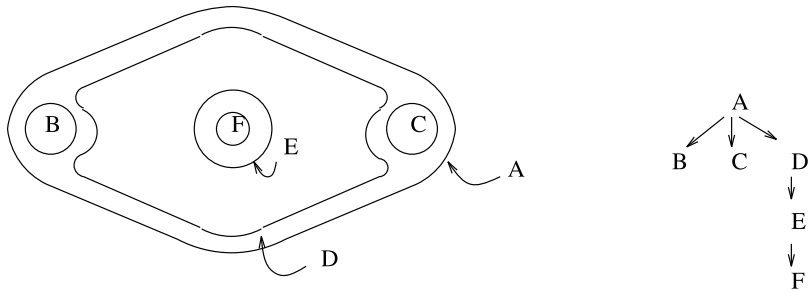


FIGURE 6.32 Sample tree representation.

Data Points to Arcs and Lines

Contours are converted to holes or α_1 profile curves, defined by combinations of arcs and lines. This is accomplished using the curvature along the contour. Areas in which the curvature is zero (or below a small threshold) for a specified distance along the contour are considered to be line segments. Areas in which the curvature is constant for a specified distance are considered to be arcs. Curvature, k, at a point on a curve is defined to be the instantaneous rate of change of the curve's slope, ϕ, with respect to curve length, s [15]:

$$k(s) = d\phi(s)/ds$$

where

$$ds = \sqrt{dx^2 + dy^2}$$

Slope is taken to be the orientation of the gradient of the difference of Gaussians (DOG) function. The DOG function is an approximation to the Laplacian as mentioned in the Zero-Crossings section of this chapter. The derivatives of slope are computed using a forward difference technique, and the results are smoothed a user-controlled number of times. A graph of curvature vs. distance along a curve can be seen in Fig. 6.36. For each arc segment, a circle is fitusing a least-squares fit [14], and then the endpoints of the arc segment are grown until the distance from the contour to the fitted circle exceeds a tolerance. This process is repeated until growing has no effect or another segment is reached. A similar method is used for the line segments. Segment data is stored as a linked list (see Fig. 6.37).

A Hough transform technique was considered for fitting curves. Although very easy to implement (one was implemented in about an hour for comparison), it was found to be too expensive in terms of memory. See Appendix C for a comparison between the technique used and the Hough transform.

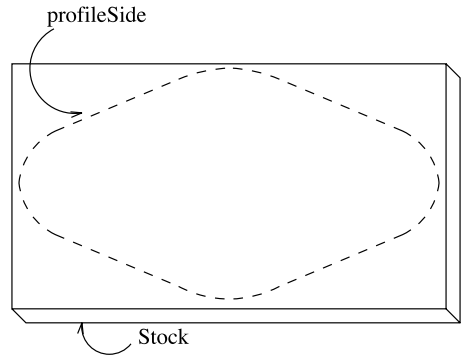


FIGURE 6.33 Stock and profileSide.

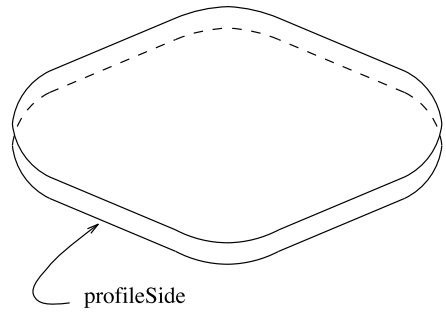


FIGURE 6.34 ProfileSide.

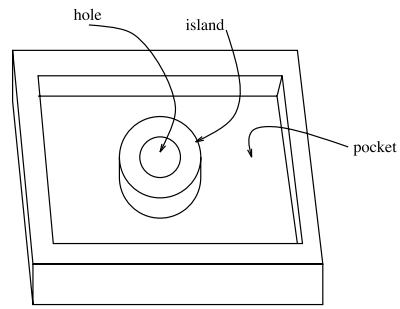


FIGURE 6.35 Positive (island) and negative (hole and pocket) features.

Arcs and Lines to Machined Features

If a negative feature contains a positive feature, then it must be a profilePocket with an island (the positive feature). The island is trimmed to the height of the positive feature with a profileGroove. If the negative feature contains no positive feature and is composed of only one arc segment, then it can be represented by a hole. To be a hole, the arc's radius must match one of a list of drill sizes within a tolerance. If a hole contains no other features, and the interior of the raw image is below a threshold, it can be considered to be a through-hole.

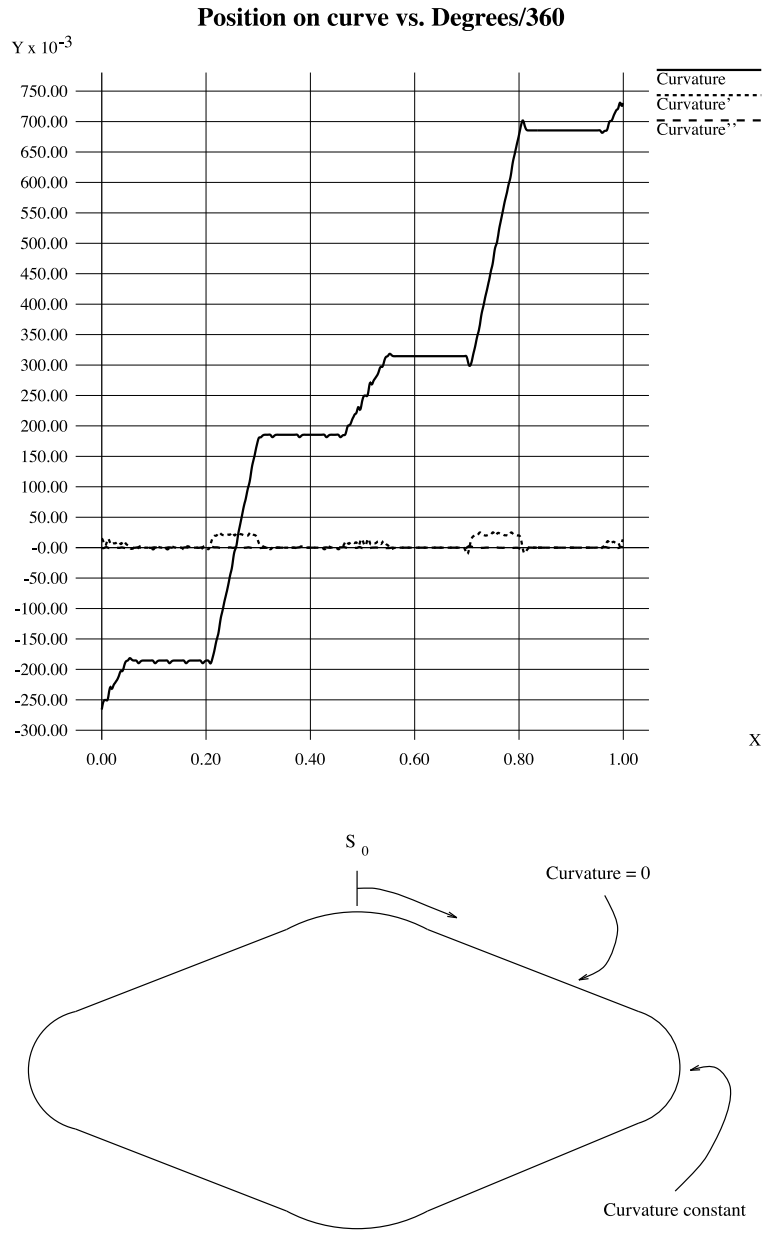


FIGURE 6.36 Curvature, and first and second derivatives.

Some aspects of machined features are difficult to measure accurately using our current image processing algorithms. For example, fillets between line segments in a profilePocket may not be within the accuracy of our vision, but are necessary for machineability. In cases such as these, default values are selected so as to have minimal deviation from the reverse-engineered model, yet allow the model to be machined. It is anticipated that some aspects (such as chamfers and threads) may be detected, although not accurately measured with vision.

In its current implementation, each contour is treated as, at most, one machined feature (some contours may be islands and therefore part of another contour's feature). Future work will allow contours to be made from multiple features if appropriate. For example, combinations of drilled

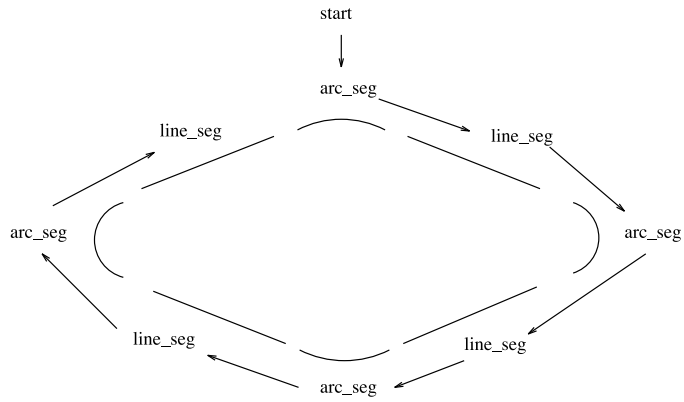


FIGURE 6.37 Contour segment data structure.

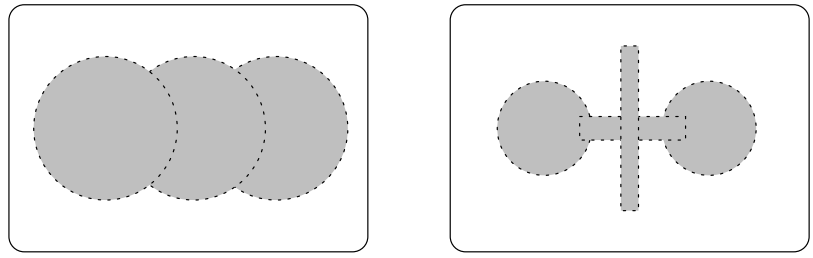


FIGURE 6.38 Possible combinations.

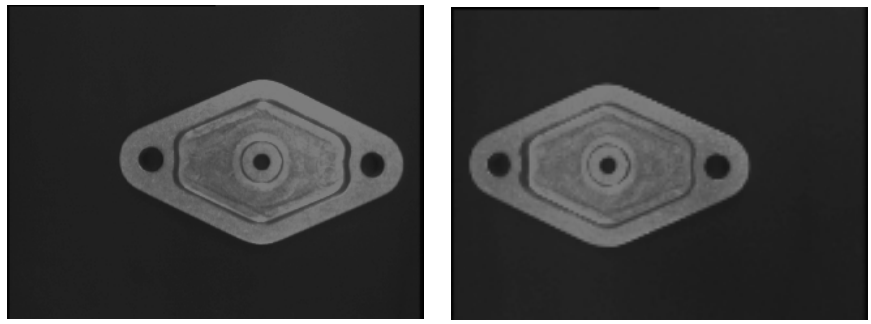


FIGURE 6.39 Stereo image pair from which ranged contours are computed.

holes, slots, and pockets may be produced (see Fig. 6.38), based on a machining/inspection time/cost analysis, which will include such factors as time needed to select and load tools (operator), change tools, etc. This problem has some characteristics that may be best solved through artificial intelligence or optimization techniques.

Results

Although the model at this intermediate stage is still crude, it was considered a useful test to have a part manufactured from it. This intermediate stage model will later be updated with CMM data as described in the section on Integration Efforts.

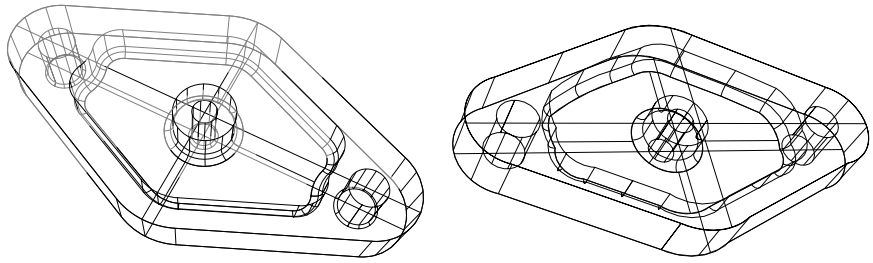


FIGURE 6.40 Original and reverse-engineered part models.

The right portion of Fig. 6.40 shows the result of applying this method to an actual part. The original part model is shown in the left portion of that figure and the stereo image pair used in image processing is shown in Fig. 6.39. Note that although the original CAD model was available in this case, it is used only for demonstration purposes, not as part of the reverse-engineering process. The reverse-engineered model was used to create a process plan and machine a part on the Monarch VMC-45 milling machine. The original part and reproduction can be seen in Fig. 6.41.

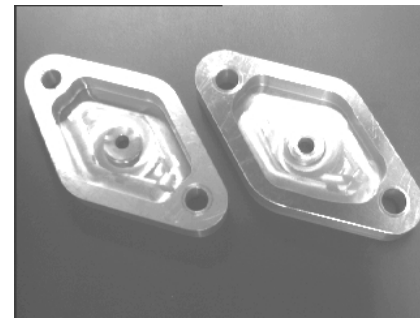


FIGURE 6.41 Original and reproduction.

6.6 System Integration

The Initiation Experiment

This experiment was performed to integrate the visual system with the state machine. An appropriate DRFSM was generated by observing the part and generating the feature information. A mechanical part was put on a black velvet background on top of the coordinate measuring machine table to simplify the vision algorithms. The camera was placed on a stationary tripod at the base of the table so that the part was always in view. The probe could then extend into the field of view and come into contact with the part, as shown in Fig. 6.42.

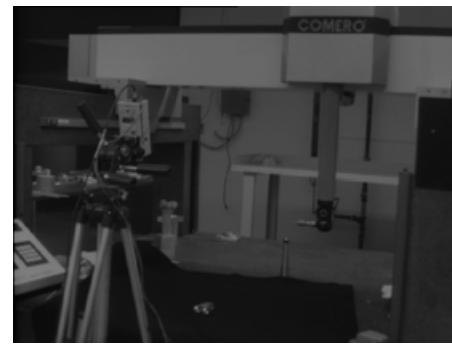


FIGURE 6.42 Experimental setup.

Once the first level of the DRFSM was created, the experiment proceeded as follows. First, an image was captured from the camera. Next, the appropriate image processing takes place to find the position of the part, the number of features observed (and the recursive string), and the location of the probe. A program using this information produces a state signal that is appropriate for the scene. The signal is read by the state machine and the next state is produced and reported. Each closed feature is treated as a recursive problem; as the probe enters a closed region, a new level of the DRFSM is generated with a new transition vector. This new level then drives the inspection for the current closed region.

The specific dynamic recursive DEDS automaton generated for the test was a state machine G (shown in Fig. 6.43.); where the set of states $X = \{\text{Initial, EOF, Error, A, B, C, D}\}$ and the set of transitional events $\Sigma = \{1, 2, 3, 4, 5, 6, 7, 8, 9, \text{eof}\}$. The state transitions were controlled by the input signals supplied by intermediate vision programs. There are four stable states A, B, C, and D that describe the state of the

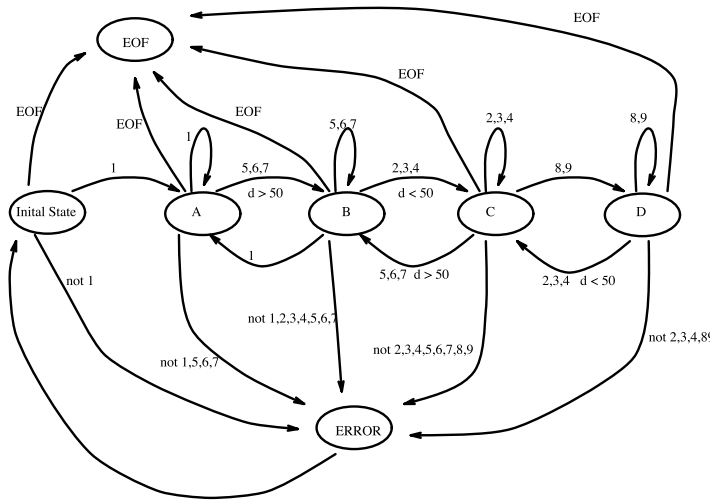


FIGURE 6.43 State machine used in test.

probe and part in the scene. The three other states—Initial, Error, and EOF—specify the actual state of the system in special cases. The states can be interpreted as:

Initial State: waiting for first input signal

A: part alone in scene

B: probe and part in scene, probe is far from part

C: probe and part in scene, probe is close to part

D: probe touching or overlapping part (recursive state)

Error: an invalid signal was received

EOF: the end of file signal was received

The Second Experiment

In the second experiment, we use a robot arm (a PUMA 560), a vision sensor (B/W CCD camera) mounted on the end effector, and a probe to simulate the coordinate measuring machine (CMM) probe, until the necessary software interface for the CMM is developed. There are several software interfaces on a Sun Sparcstation, for controlling all these devices (see Fig. 6.44.)

A DRFSM DEDS algorithm is used to coordinate the movement of the robot sensor and the probe. Feedback is provided to the robot arm, based on visual observations, so that the object under consideration can be explored. This DRFSM was generated by GIJoe, as shown in Fig. 6.6. The DEDS control algorithm will also guide the probe to the relevant parts of the objects that need to be explored in more detail (curves, holes, complex structures, etc.) Thus, the DEDS controller will be able to *model, report,* and *guide* the robot and the probe to reposition *intelligently* in order to recover the structure and shape parameters. The data and parameters derived from the sensing agent are fed into the CAD system for designing the geometry of the part under inspection. We used the α_1 design environment for that purpose. Using the automatic programming interface we have developed for α_1, we generate the required code to reconstruct the object using the data obtained by the sensing module.

Running the Experiment

The first step in running this experiment is setting the lighting conditions as desired (same conditions when constructing the reflectance map table), then initializing the robot and the camera and set them to initial positions. The experiment starts by taking images for the object from two positions, to

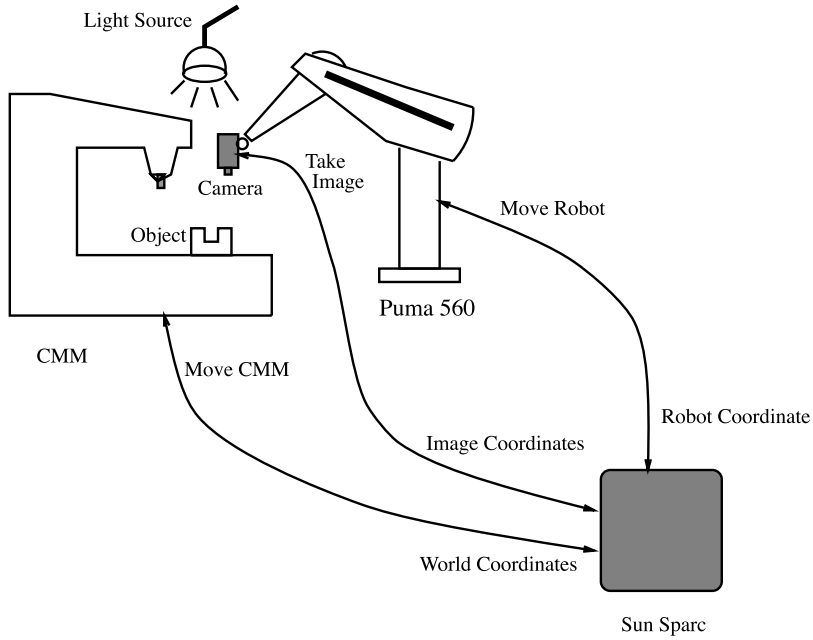


FIGURE 6.44 An experiment for inspection and reverse engineering.

generate two sets of contours to be fed into the stereo module for depth estimation. Using the stereo module with the assistance of the reflectance map table and the camera calibration module, an initial set of world coordinates for these contours is generated. Next, the DRFSM DEDS machine drives the probe and the robot arm holding the camera to inspect the object using the information generated from the stereo module and the relation between the object's contours. Figure 6.45 shows the DRFSM for this experiment.

This machine has the following states:

- **A:** the initial state, waiting for the probe to appear.
- **B:** the probe appears, and waiting for it to close. Here, close is a relative measure of the distance between the probe and the current feature, since it depends on the level of the recursive structure. For example, the distance at the first level, which represents the outer contours or features, is larger than that of the lower levels.
- **C:** probe is close, but not on feature.
- **D:** the probe appears to be on feature in the image, and waiting for physical touch indicated from the CMM machine.
- **E:** (the recursive state) physical touch has happened. If the current feature represents a closed region, the machine goes one level deeper to get the inner features by a recursive call to the initial state after changing the variable transition parameters. If the current feature was an open region, then the machine finds any other features in the same level.
- **F:** this state is to solve any vision problem happens during the experiment. For example, if the probe is occluding one of the features, then the camera position can be changed to solve this problem.
- **ERROR 1:** usually, there is time limit for each part of this experiment to be done. If for any reason, one of the modules does not finish in time, the machine will go to this state, which will report the error and terminate the experiment.

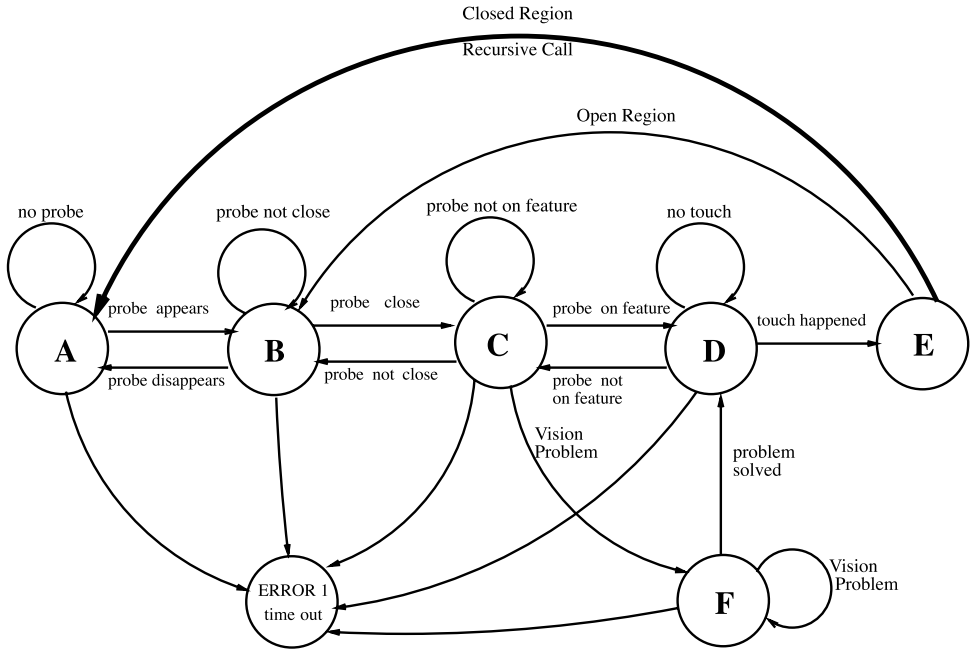


FIGURE 6.45 The DRFSM used in the second experiment.

A set of final world coordinates for the contours is obtained and fed to the α_1 interface, which in turn generates the required code for generating an α_1 model for the the object. Figure 6.46 shows a block diagram for this experiment with the results after each step.

Experimental Results, Automated Inspection

Two machine parts were used in the experiment to test the inspection automaton. The pieces were placed on the inspection table within view of the camera. Lighting in the room was adjusted so as to eliminate reflection and shadows on the part to be inspected.

Control signals that were generated by the DRFSM were converted to simple English commands and displayed to a human operator so that the simulated probe could be moved.

The machine was brought online and execution begun in State A, the start state. The camera moved to capture both 2D and 3D stereo vision information and a rough α_1 model was constructed to describe the surface, as shown in Fig. 6.47. The reconstruction takes place in state A of the machine. The constructed model is used by the machine in subsequent states. For example, the distance between the probe and the part is computed using this model and the observed probe location.

After initiating the inspection process, the DRFSM transitioned through states until the probe reached the feature boundary. The state machine then called for the closed region to be recursively inspected until finally, the closed regions are explored and the machine exits cleanly. Figures 6.48, 6.49, and 6.50 depict some exploration sequences.

6.7 Summary of Current Developments

This summary concludes the chapter by outlining some of the goals and progress within the project. We first describe some goals and methodology, and then outline current and past activities.

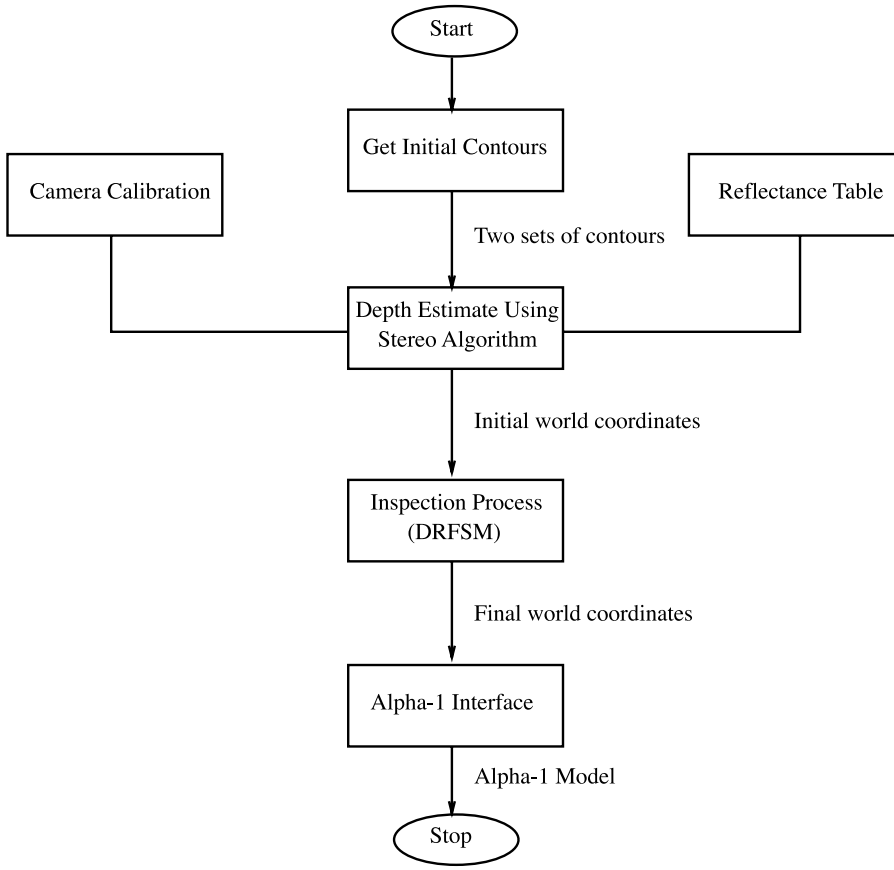


FIGURE 6.46 Block diagram for the experiment.

FIGURE 6.47 The two stereo images and the final α_1 model that was found in the experiment.

Goals and Methodology

We use an observer agent with some sensing capabilities (vision and touch) to actively gather data (measurements) of mechanical parts. Geometric descriptions of the objects under analysis are generated and expressed in terms of a CAD System. The geometric design is then used to construct a prototype of the object. The manufactured prototypes are then to be inspected and compared with the original object using the sensing interface and refinements made as necessary.

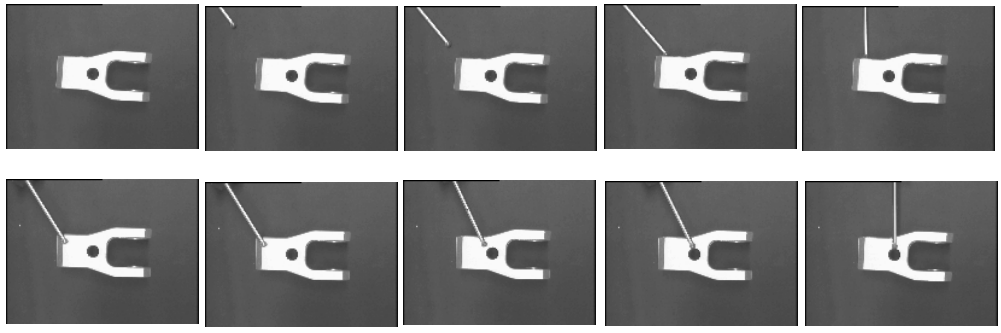


FIGURE 6.48 Test sequence (1).

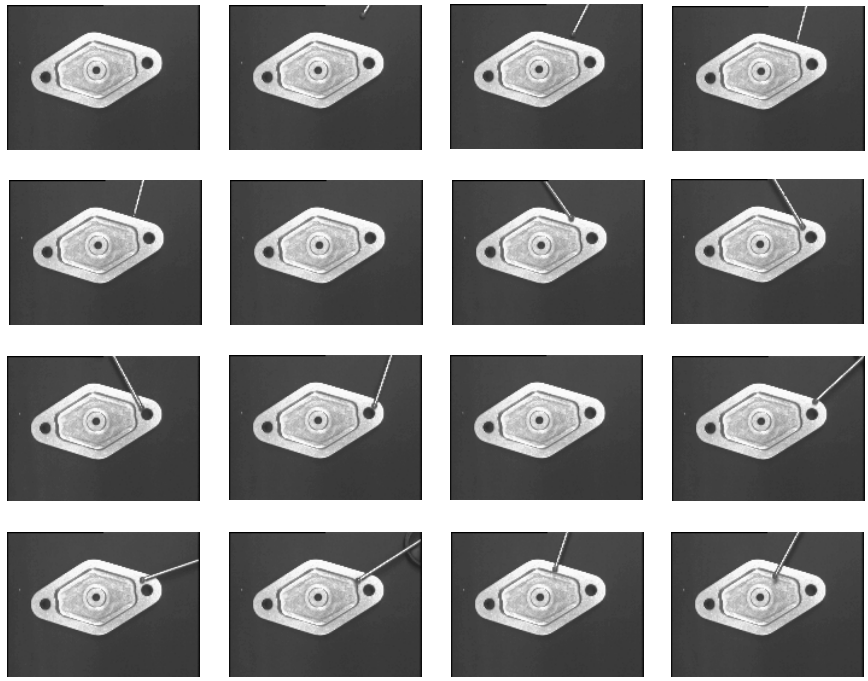


FIGURE 6.49 Test sequence (2).

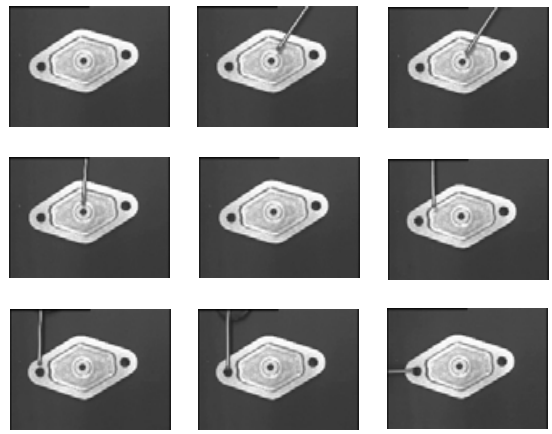


FIGURE 6.50 Test sequence (2) (contd.).

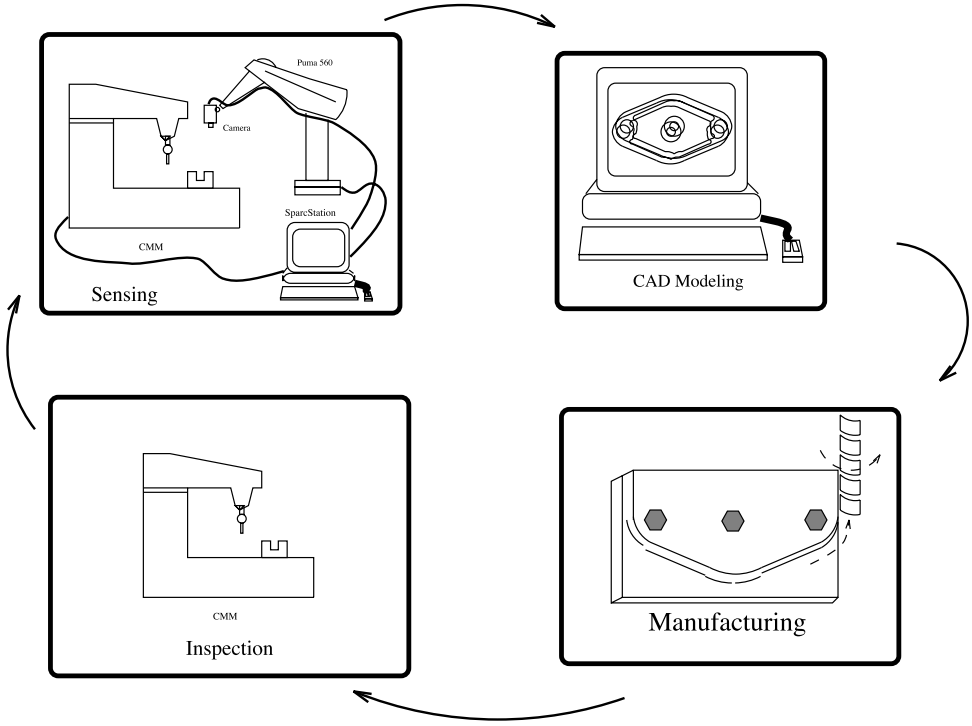


FIGURE 6.51 Closed-loop reverse engineering.

The application environment we are developing consists of three major working elements: the sensing, design, and manufacturing modules. The ultimate goal is to establish a computational framework that is capable of deriving designs for machine parts or objects, inspect and refine them, while creating a flexible and consistent engineering environment that is extensible. The control flow is from the sensing module to the design module, and then to the manufacturing component. Feedback can be re-supplied to the sensing agent to inspect manufactured parts, compare them to the originals, and continue the flow in the loop until a certain tolerance is met (see Fig. 6.51). The system is intended to be ultimately as autonomous as possible. We study what parts of the system can be implemented in hardware. Some parts seem to be inherently suited to hardware, while other parts of the system it may be possible to put in hardware; but experimentation will provide the basis for making that decision. Providing language interfaces between the different components in the inspection and reverse-engineering control loop is an integral part of the project.

Current Developments

We use a robot arm (a PUMA 560), a vision sensor (B/W CCD camera) mounted on the end effector, and will be using the coordinate measuring machine (CMM) with the necessary software interfaces to a Sun SparcStation as the sensing devices. A DRFSM DEDS algorithm is used to coordinate the movement of the robot sensor and the CMM. Feedback is provided to the robot arm, based on visual observations, so that the object(s) under consideration can be explored. The DEDS control algorithm will also guide the CMM to the relevant parts of the objects that need to be explored in more detail (curves, holes, complex structures, etc). Thus, the DEDS controller will be able to *model, report,* and *guide* the robot and the CMM to reposition *intelligently* in order to recover the structure and shape parameters.

The data and parameters derived from the sensing agent are then fed into the CAD system for designing the geometry of the part(s) under inspection. We use the α_1 design environment for that purpose. The goal is to provide automatic programming interfaces from the data obtained in the sensing module to the α_1 programming environment. The parametric and 3-D point descriptions are to be integrated to provide consistent and efficient surface descriptions for the CAD tool. For pure inspection purposes, the computer aided geometric description of parts could be used as a *driver* for guiding both the robotic manipulator and the coordinate measuring machine for exploring the object and recognizing discrepancies between the real part and the model. The computer aided design parameters will then to be used for manufacturing the prototypes.

The software and hardware requirements of the environment are the backbone for this project. We selected parts of the system for possible hardware implementation. The DEDS model, as an automaton controller, is very suitable for Path Programmable Logic (PPL) implementation. A number of the visual sensing algorithms could be successfully implemented in PPL, saving considerable computing time. There is a lot of interfacing involved in constructing the inspection and reverse-engineering environments under consideration. Using multi-language object-based communication and control methodology between the three major components (sensing, CAD, and CAM) is essential.

Past, Current, and Future Activities

Completed Activities

- Designed the DRFSM DEDS framework for recursive inspection
- Implemented image processing modules for recognizing features and probe position on the parts
- Designed and implemented visual structure recovery techniques for machine parts (using stereo, contour, and illumination map data) and implemented calibration routines
- Designed and implemented a sensing to CAD interface for generating α_1 code for bodies from depth, contour (and data reduction) illumination map, and the recursive feature relationships
- Implemented the DRFSM DEDS automata for recursive inspection (using robot-held camera, probe, and actual parts)
- Designed sensor and strategy-based uncertainty modeling techniques for the robot-held camera, for recovering the DEDS transitional "events" with uncertainty
- Designed and implemented a modification to an existing reactive behavior design tool (GIJoe) to accommodate "dumping" the code of DRFSM DEDS from a graphical interface (used to draw the inspection control automaton)
- Implemented feature identification for subsequent manufacturing (from sensed data, i.e., what does set(s) of sensed data points "mean" in terms of manufacturing features)
- Manufactured parts from camera reconstructed α_1 surfaces

Current Activities

- Designing the DEDS to VLSI design language interface (a graphical interface)
- Designing and implementing the software "uncertainty" module for subsequent hard-wiring into a chip
- Using focusing, motion, moments, shading, and more accurate robot and camera calibration techniques to enhance the visual processing
- Feature *interaction* identification for manufacturing (i.e., how can sensed features best be represented for manufacturing)

- Modifying the sensing to CAD interface for allowing CMM sensed data, in addition to visual data
- Implementing the DRFSM DEDS automata for recursive inspection and reverse engineering (using moving camera, CMM and actual parts)
- Implementing "safety" recursive DEDS for checking the sensing activities, for example, positions of probe, part, and camera

Future Activities

- Implement the VLSI modules for the DRFSM DEDS controller
- Implement the "uncertainty" chip
- Manufacture parts from camera and CMM reconstructed α_1 surfaces (with feature interaction identification built in)
- Writing and using a common shared database for storing data about the geometric models and the rules specifying the communication between the different phases
- Implement sensor-based noise modeling modules for the robot-held camera and the CMM (hardware and software)

6.8 Integration Efforts

The following explains some of the integration efforts within the different areas of the project.

Robotics and Sensing

We intend to develop a software interface for the CMM machine, and a discrete event dynamic system (DEDS) algorithm will be used to coordinate the movement of the robot sensor and the CMM. The DEDS control algorithm will also guide the CMM to the relevant parts of the objects that need to be explored in more detail (curves, holes, complex structures, etc.).

As a starting point to develop this interface, we will work with the Automated Part Inspection (API) package. API is a semi-automatic feature-based part inspector that is fully integrated with the α_1 system. This package, some of which can be seen in Fig. 6.52, enables a user with an α_1 model composed of machined features to simulate and/or drive the CMM to inspect the machined part. Using our intermediate feature-based model to guide the inspection as if it were the original, we will be able to incorporate the sense of touch into our knowledge base. With a new, more accurate model, we can loop back to the beginning of the inspection process until we have captured every aspect of the parts we inspect to the tolerances we desire.

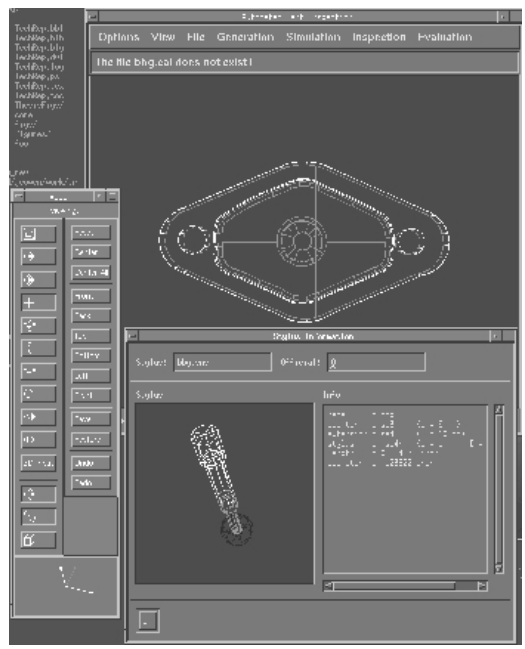


FIGURE 6.52 The API user interface.

Computer Aided Design and Manufacturing

We intend to develop the CAD interface to be more accurate and to accept more complicated models. The goal is to enhance the automatic

programming interface between the data obtained in the sensing module to the α_1 programming environment. The parametric and 3-D point descriptions are to be integrated to provide consistent and efficient surface descriptions for the CAD tool. For pure inspection purposes, the computer aided geometric description of parts could be used as a *driver* for guiding both the robotic manipulator and the coordinate measuring machine for exploring the object and recognizing discrepancies between the real part and the model.

The computer aided design parameters are then to be used for manufacturing the prototypes. Considerable effort has been made for automatically moving from a computer aided geometric model to a process plan for making the parts on the appropriate NC machines and then to automatically generate the appropriate machine instructions [6]. We use the Monarch VMC-45 milling machine as the manufacturing host. The α_1 system produces the NC code for manufacturing the parts.

VLSI, Uncertainty Modeling, and Languages

The software and hardware requirements of the environment are the backbone for this project. We intend to select parts of the system implementation and study the possibility of hard-wiring them. There has been considerable effort and experience in VLSI chip design [5, 8] and one of the subproblems would be to study the need and efficiency of making customized chips in the environment. The DEDS model, as an automaton, is very suitable for path programmable logic (PPL) implementation. A number of the visual sensing algorithms could be successfully implemented in PPL, saving considerable computing time. Integrated circuits for CAGD surface manipulation is an effort that is already underway. We intend to investigate a new area: the possibility of implementing the DEDS part of the system in integrated circuitry.

Another important part to be implemented in hardware, is the "uncertainty" chip, which will provide fast decisions about the accuracy of our measurements. This is important for deciding whether the part needs more inspection steps or not. The uncertainty model depends on the nature of the part being inspected, the sensor, the strategy being used to sense the part, and the required accuracy.

There is a lot of interfacing involved in constructing the inspection and reverse-engineering environments under consideration. The use of multi-language object-based communication and control methodology between the three major components (sensing, CAD, and CAM) is essential. We intend to use a common shared database for storing data about the geometric model and the rules governing the interaction of the different phases in the reproduction and inspection paradigms [10, 17]. We have already used a graphical behavior design tool [4] for the automatic production of the sensing DEDS automata code, from a given control language description. A sensing $\rightarrow$ CAD interface has been developed as well.

6.9 Conclusions

We propose a new strategy for inspection and/or reverse engineering of machine parts and describe a framework for constructing a full environment for generic inspection and reverse engineering. The problem is divided into *sensing, design,* and *manufacturing* components, with the underlying software interfaces and hardware backbone. We use a recursive DEDS DRFSM framework to construct an intelligent sensing module. This project aims to develop sensing and control strategies for inspection and reverse engineering, and also to coordinate the different activities between the phases. The developed framework utilizes existing knowledge to formulate an adaptive and goal-directed strategy for exploring, inspecting, and manufacturing mechanical parts.

References

1. Aloimonos, J. and Shulman, D. *Integration of Visual Modules an Extension of the Marr Paradigm.* Academic Press, 1989.
2. α_1. α_1 User's Manual. University of Utah, 1993.

3. Benveniste, A. and Guernic, P. L. Hybrid dynamical systems theory and the signal language. *IEEE Transactions on Automatic Control,* 35, 5, 1990.
4. Bradakis, M. J. Reactive behavior design tool. Master's thesis, Computer Science Department, University of Utah, January 1992.
5. Carter, T. M., Smith, K. F., Jacobs, S. R., and Neff, R. M. Cell matrix methodologies for integrated circuit design. *Integration, The VLSI Journal,* 9, 1, 1990.
6. Drake, S. and Sela, S. A foundation for features. *Mechanical Engineering,* 111, 1, 1989.
7. Ens, J. and Lawrence, P. An investigation of methods for determining depth from focus. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 15, 2, 1993.
8. Gu, J. and Smith, K. A structured approach for VLSI circuit design. *IEEE Computer,* 22, 11, 1989.
9. Hsieh, Y. C. Reconstruction of sculptured surfaces using coordinate measuring machines. Master's thesis, Mechanical Engineering Department, University of Utah, June 1993.
10. Lindstrom, G., Maluszynski, J., and Ogi, T. Using types to interface functional and logic programming. In *1991 SIGPLAN Symposium on Principles of Programming Languages* (July 1990), Vol. 10. technical summary (submitted).
11. Marr, D. and Hildreth, E. Theory of edge detection. In *Proceedings Royal Society of London Bulletin,* 204, 301–328, 1979.
12. Nerode, A. and Remmel, J. B. A model for hybrid systems. In *Proc. Hybrid Systems Workshop, Mathematical Sciences Institute* (May 1991). Cornell University.
13. Özveren, C. M. *Analysis and Control of Discrete Event Dynamic Systems: A State Space Approach.* PhD thesis, Massachusetts Institute of Technology, August 1989.
14. Pratt, V. Direct least-squares fitting of algebraic surfaces. In *SIGGRAPH '87* (1987), 145–152.
15. Schalkoff, R. J. *Digital Image Processing and Computer Vision.* John Wiley & Sons, 1989.
16. Sobh, T. M. and Bajcsy, R. A model for observing a moving agent. In *Proceedings of the Fourth International Workshop on Intelligent Robots and Systems (IROS '91)* (November 1991). Osaka, Japan.
17. Swanson, M. and Kessler, R. Domains: efficient mechanisms for specifying mutual exclusion and disciplined data sharing in concurrent scheme. In *First U.S./Japan Workshop on Parallel* (August 1989).
18. Tsai, R. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. In *Radiometry - (Physics-Based Vision),* G. H. L. Wolff, S. Shafer, Ed. Jones and Bartlett, 1992.
19. van Thiel, M. T. *Feature Based Automated Part Inspection.* Ph.D. thesis, University of Utah, June 1993.
20. Willson, R. Personal communication, 1993.

Appendix A: Sample GI Joe Output

```
int State_B(VTV_ptr)
vtype *VTV_ptr;
{
    int DoneFlag;
    EventType Event;
    vtype *newVTV_ptr;
    int EventMask=0;

#ifdef VERBOSE

    printf("in state B\n");

#endif
    if (VTV_ptr == NULL) {

#ifdef VERBOSE

    fprintf(stderr, "*** ERROR: null vtv in state B\n");
```

```
#endif

            exit (4);
        };
        EventMask |= TimeOutMask;
        EventMask |= NoProbeMask;
        EventMask |= ProbeCloseMask;
        EventMask |= ProbeFarMask;
        DoneFlag 5 FALSE;
        while (!DoneFlag) {
            Event = Get_DRFSM_Event(EventMask, VTV_ptr);
            if (Event . type == TimeOut) {
                DoneFlag = TRUE;
                if (Event . fn != NULL) DoneFlag = (*(Event . fn))();
                State_ERROR(VTV_ptr);
            }
            else if (Event . type == NoProbe) {
                DoneFlag = TRUE;
                if (Event . fn != NULL) DoneFlag = (*(Event . fn))();
                State_A(VTV_ptr);
            }
            else if (Event . type == ProbeClose) {
                DoneFlag = TRUE;
                if (Event . fn != NULL) DoneFlag = (*(Event . fn))();
                State_C(VTV_ptr);
            }
            else if (Event . type == ProbeFar) {
            }
        }
}
```

Appendix B: Sample Calibration Code Output

Coplanar calibration (full optimization)

data file: a . pts

f = 8.802424 [mm]
kappa1 = 0.003570 [1/mm^2]

Tx = −25.792328, Ty = 77.376778, Tz = 150.727371 [mm]

Rx = −134.988935, Ry = −0.127692, Rz = −0.068045 [deg]

R

0.999997	0.000737	0.002416
−0.001188	−0.706972	0.707241
0.002229	−0.707242	−0.706968

sx = 1.000000
Cx = 276.849304, Cy = 252.638885 [pixels]

Tz/f = 17.123394

calibration error: mean = 0.331365,
standard deviation = 0.158494 [pixels]

Appendix C: Comparison Between Hough Transform and Curvature Technique

The curvature technique as implemented for this application is described in flowchart form in Fig. C.1. Using a similar analysis for a basic Hough transform implementation (just to detect circles) shows that it would require:

- *MMM* assignments (to initialize memory)
- *NMM*
  - 5 addition operations
  - 3 multiplication operations
  - 1 sqrt operations
  - 6 assignments
  - 3 comparisons
- *MMM* integers for memory

where *M* is the precision.

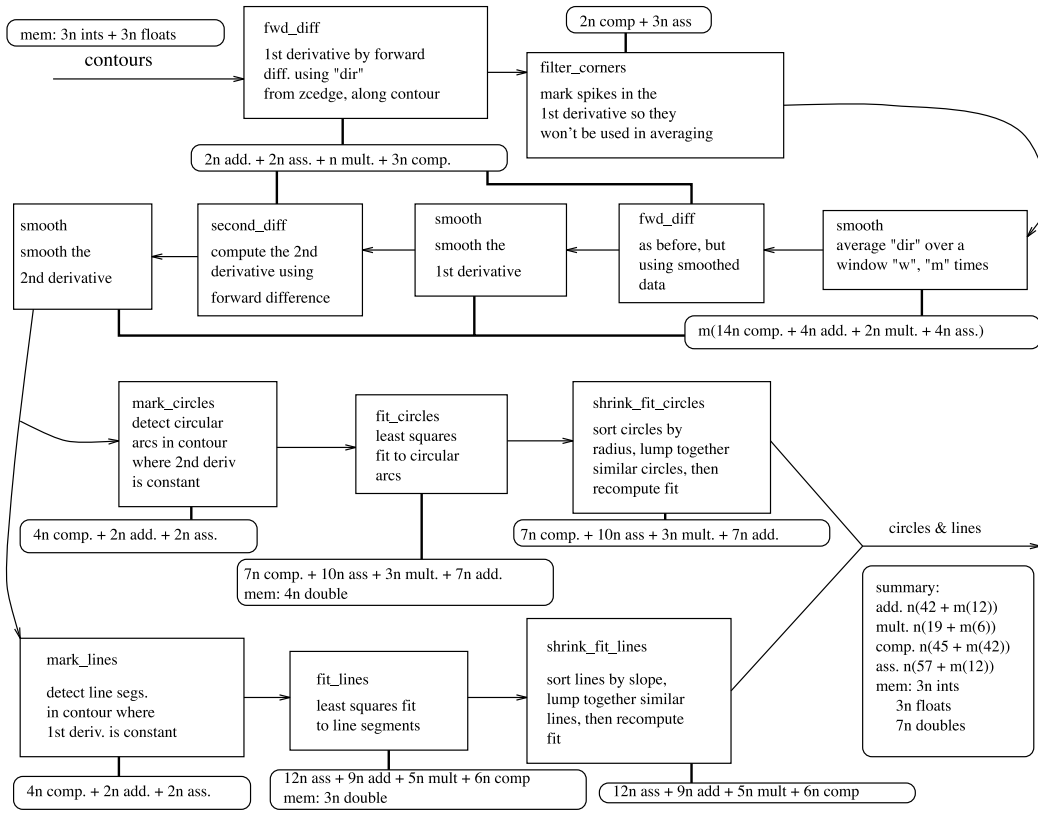


FIGURE C.1 Flowchart of the implemented curvature technique.

Assuming pixel accuracy, M is approximately N/π. N, for this application, can be taken to be contour length, bounded by $\sqrt{N_{lines}N_{samples}}$. Thus, the Hough tranform may be considered of order N^3 while the curvature technique used is at most order N^2. Not included in the Hough evaluation is that it would be necessary to do some sort of mode detection to determine the number of circles found.

It is anticipated that the fitting algorithm may be extended to include other conic sections than circles, and additionally that it may be extended to use three dimensional coordinates. While the Hough transform is a very useful technique, we anticipate that its memory and order requirements will grow too rapidly to meet our future needs.

7

Application of Reference Architectures for Enterprise Integration

7.1 Introduction
7.2 Enterprise Reference Architectures
7.3 CIMOSA
Introduction • CIMOSA Approach • CIMOSA Modeling Framework • CIMOSA Integrating Infrastructure
7.4 Application of CIMOSA at Traub
Introduction • Reorganization Objective • Requirements Definition • Architectural Design • Detailed Design • Implementation • The Role of CIMOSA in Traub's Reorganization Process
7.5 The Practical Value of CIMOSA
7.6 Two Other Enterprise Reference Architectures
GRAI/GIM • PERA
7.7 Baan's Enterprise Modeler
7.8 Summary

Arian Zwegers
Baan Development

Henk Jan Pels
Eindhoven University of Technology

7.1 Introduction

Enterprises face integration problems due to necessary internal adaptations to cope with severe competition in the global marketplace. To survive at the global market, efficient operation and innovative management of change are essential. Enterprises need to evolve and be reactive, so that change and adaptation should be a natural dynamic state rather than something occasionally forced onto the enterprise. Enterprise engineering is the discipline that identifies the need for change in enterprises and carries out that change expediently and professionally. Enterprise engineering intends to cater for continuous evolution and to achieve a certain level of enterprise integration. Enterprise integration propagates the integration of all enterprise resources to optimize business operations. That is, enterprise integration involves the collection, reduction, storage, and use of information, the coordination of product flows, the organization of machines, the integration of human actions, etc.

Recently, some reference architectures have been defined that aim to provide the necessary frameworks for enterprise integration. They are based on the idea that large parts of enterprise integration projects are similar in every type of enterprise. These parts could be standardized and supported by methodologies, tools, and other products that facilitate integration projects (Bernus et al., 1996).

The objective of this chapter is to investigate the practical value of enterprise reference architectures concerning the integration of production systems. For this, three enterprise reference architectures have

been studied: namely CIMOSA, GRAI/GIM, and PERA. In addition, the CIMOSA reference architecture has been applied during an industrial reorganization project.

The outline of this chapter is as follows. Section 7.2 outlines the objectives and ideas behind reference architectures for enterprise integration. An essential point is that enterprise reference architectures aim to support the whole life cycle of an enterprise integration project rather than just the (architectural) design activities.

CIMOSA is the best-known and most studied of the three enterprise reference architectures. In addition, it is also the one with the most impact at the standardization committees. CIMOSA is examined more closely in order to obtain a good understanding of the concepts of reference architectures for enterprise integration. Section 7.3 gives an explanation of CIMOSA's view on enterprise integration and how to accomplish it. Furthermore, CIMOSA's two most important elements, the modeling framework and the integrating infrastructure, are shortly discussed.

Section 7.4 presents an application of CIMOSA during a reorganization project at a machine tool manufacturer. The role of the CIMOSA modeling framework in the design of a functional control architecture is illustrated. Due to immature specifications, the assistance of the CIMOSA integrating infrastructure was rather limited.

Subsequently, the practical value of CIMOSA is discussed. Although the discussion focuses on CIMOSA, most statements apply to most enterprise reference architectures. Section 7.5 shows that the realization of CIMOSA's true objective, namely a dynamic, flexible, adaptive enterprise by execution of models, is far away.

The two other major reference architectures for enterprise integration, GRAI/GIM and PERA, are presented in Section 7.6. Whereas CIMOSA lacks a true engineering methodology, these two reference architectures are accompanied by well-defined methodologies.

Finally, Section 7.7 shows an alternative approach in the field of enterprise resource planning (ERP) that does realize some of the targets of enterprise integration thinking. The Dynamic Enterprise Modeler of Baan Company allows users to make changes in models that are directly reflected in changes in the underlying Baan ERP applications.

7.2 Enterprise Reference Architectures

Several problems occur during the (re-)design of an integrated manufacturing system. Because such a system cannot be bought off-the-shelf, each company has to develop its own. Williams et al. (1994a) notice that designing an integrated manufacturing system meets with a number of difficulties. Several viewpoints must be taken into account; not only the technical point, but also the economic, social, and human points of view have to be considered. By definition, CIM systems are very complex, and the development of such systems is often quite expensive and risky. Most systems are not designed from scratch; the existing systems have to be considered in the development process, so that newly developed systems are integrated with the old ones. In addition, Aguiar and Weston (1995) claim that the activities performed during each phase of the life cycle are essentially derived from ad hoc procedures, so that the quality of the resultant system will depend considerably on the experience of the persons involved. The problem is accentuated due to poor formalism with which those activities are usually carried out. This often leads to solutions that do not adequately address business requirements, experience low repeatability of successful results, and reveal a lack of traceability of design decisions, etc.

The objective of enterprise reference architectures is to offer the framework that solves the problems mentioned above, and with which an enterprise might develop its integrated manufacturing system. The idea behind enterprise reference architectures is that a large part of integration projects is in fact similar and common in every type of enterprise (Williams et al., 1994a). This part could be standardized and utilized instead of re-developing it from scratch. Once standardized, generally accepted reference architectures can be supported by tools, methodologies, and a range of compatible products, thus making the entire integration project more time- and cost-efficient.

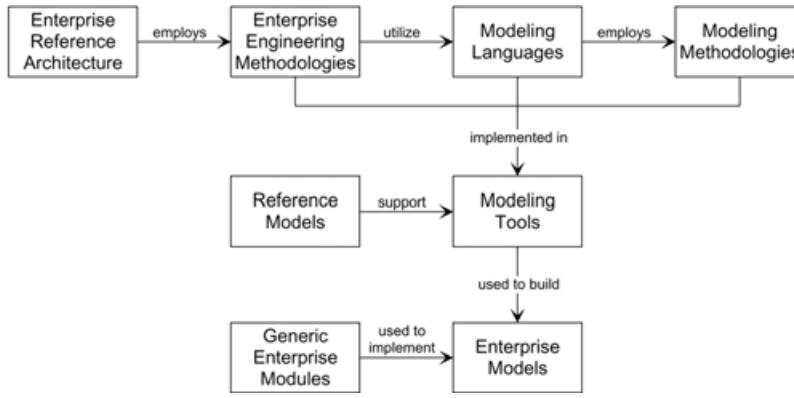


FIGURE 7.1 Components of the GERAM framework.

Wyns et al. (1996) state that the benefits of reference architectures lie, among others, in unified and unambiguous terminology, in envisaged simplicity of designing specific systems, and in high quality by relying on proven concepts. In addition, a reference architecture models the whole life history of an enterprise integration project. It indicates and justifies how and at what stage in the development process external constraints and engineering design decisions are introduced, thereby providing traceability between solution-independent requirements and final realizations. It provides the framework in which enterprise integration methodologies work.

The IFAC/IFIP Task Force on Architectures for Enterprise Integration has developed an overall frame work to organize existing enterprise integration knowledge. The proposed framework was entitled GERAM—Generalized (or Generic) Enterprise Reference Architecture and Methodology. GERAM is about those methods, models, and tools that are needed to build and maintain the integrated enterprise. GERAM defines a toolkit of concepts for designing and maintaining enterprises for their entire life cycle. The scope of GERAM encompasses all the knowledge needed for enterprise engineering. Thus, GERAM provides a generalized framework for describing the components needed in all types of enterprise engineering processes. GERAM provides a description of all elements recommended in enterprise engineering and integration (IFAC/IFIP, 1997).

Fig. 7.1 shows the components of the GERAM framework. The most important component is the Enterprise Reference Architecture, which defines enterprise-related concepts recommended for use in enterprise engineering projects. The reference architecture should point toward purposeful organization of enterprise concepts (Nell, 1996); it should identify and structure concepts for enterprise integration, most notably life-cycle activities and architectural concepts such as abstraction, hierarchy, and views. GERAM distinguishes between the methodologies for enterprise engineering and the modeling languages that are used by the methodologies to describe and specify the structure, content, and behavior of the enterprise. These languages will enable the modeling of the human part in the enterprise operation as well as the part of business processes and supporting technologies. The modeling process is supported by guidelines, and the results of the process are enterprise models that represent all or part of the enterprise operations, including its manufacturing or service tasks, its organization and management, and its control and information systems. These models should be used to improve the ability of the enterprise to evaluate operational or organizational alternatives (e.g., by simulation), and thereby improve its current and future performance.

The methodologies and the languages used for enterprise modeling are supported by enterprise modeling tools. The modeling process is enhanced by using reference models that provide reusable models of human roles, processes, and technologies. The operational use of enterprise models is supported by specific modules that provide prefabricated products such as human skill profiles for specific professions, common business procedures (e.g., banking and tax rules), or IT infrastructure services.

By means of the GERAM framework, the overlaps and differences of enterprise reference architectures can be identified. After all, the ways in which enterprise reference architectures try to achieve their objectives differ from one reference architecture to the other. The following components are the minimal set of elements a reference architecture should include:

1. Enterprise engineering methodologies, which can be thought of as roadmaps and instructions of how to go about an enterprise integration project or program
2. Modeling languages, which are needed to support enterprise integration, and should be placed in relation to each other by means of the reference architecture
3. A modeling methodology, which comprises a set of guidelines that define the steps to be followed during a modeling activity

In addition, the following components are elements that should preferably accompany an enterprise reference architecture:

4. Modeling tools, which are computer programs that help the construction, analysis, and, if applicable, the execution of enterprise models as expressed in enterprise modeling languages
5. Reference models, which contain a formal description of a type (or part of an) enterprise
6. Generic enterprise modules, which are products that implement (parts of) a reference model; for example, an integrating infrastructure, or components thereof

7.3 CIMOSA

Introduction

The goal of the ESPRIT project AMICE (reverse acronym of European Computer Integrated Manufacturing Architecture) was to develop an Open System Architecture for CIM (CIMOSA). The project started in the mid-1980s and finished after some extensions in the mid-1990s. CIMOSA should facilitate continuous enterprise evolution and make necessary operational improvements manageable. At the same time, CIMOSA should provide a strategy to deal with legacy systems to protect current and planned investment in an enterprise's production process. CIMOSA aims to offer support for enterprise integration, more precisely for "business integration" and "application integration" (AMICE, 1993a).

Business integration is concerned with the coordination of operational and control processes to satisfy business objectives. In every enterprise, processes are present that provide supervisory control of the operational processes and coordinate the every-day execution of activities. CIMOSA aims to integrate these processes by process-oriented modeling of enterprises. Modeling these processes and their interrelations can be used in decisions regarding the requested level of business integration.

Application integration, which affects the control of applications, is concerned with the usage of information technology to provide interoperation between manufacturing resources. Cooperation between humans, machines, and software programs must be established by the supply of information through inter- and intra-system communication. CIMOSA tries to support integration at this level by defining a sufficient infrastructure to permit system-wide access to all relevant information regardless of where the data reside.

In addition to business and application integration, a third level of integration is discerned by the AMICE project, namely physical system integration. This level is concerned with the interconnection of physical systems and has led to a number of standards, such as OSI and MAP. CIMOSA supports this type of integration by adherence to the standards.

CIMOSA Approach

CIMOSA provides a framework that guides designers in the design and implementation of CIM systems. In addition, it aims to guide vendors in the development of CIM system components, so that these

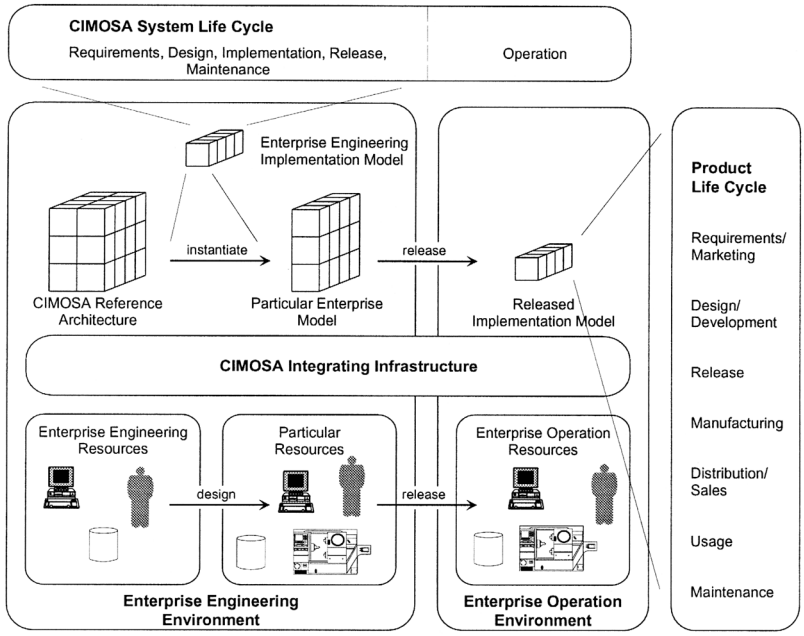


FIGURE 7.2 Overview of the CIMOSA concepts (AMICE, 1993a.)

components can be added and removed at will. It does not provide a standard architecture to be used by the entire manufacturing industry, but rather a reference architecture from which a particular enterprise can derive particular architectures that fulfill its needs. As such, CIMOSA adheres to a descriptive, rather than a prescriptive methodology (AMICE, 1993a). Fig. 7.2 gives an overview of the CIMOSA concepts.

The CIMOSA modeling framework enables one to describe a particular enterprise. Accompanying methods, called "model creation processes," guide engineers in the creation and maintenance process to obtain and maintain a consistent system description. CIMOSA supports the explicit description of enterprise processes at different levels of abstraction for strategic, tactical, and operational decision-making. Simulation of alternatives and evaluation of design decisions enhances the decision support. CIMOSA supports incremental modeling of the enterprise rather than following an overall top-down approach. In short, CIMOSA allows the end user to define, prototype, design, implement, and execute its business processes according to his/her needs.

CIMOSA facilitates a system life cycle that guides the user through model engineering and model execution. The life cycle starts with the collection of business requirements in a requirements definition model, and goes, through the translation of the requirements into a system design model, to the description of the implemented system. These phases are followed by a model release for operation and model execution for operational control and monitoring. However, various methodologies consisting of various system life-cycle phases are possible to instantiate particular models from the reference architecture. These methodologies are supported by tool sets, which are defined by enterprise engineering implementation models.

Model engineering and model execution are supported by the CIMOSA integrating infrastructure. This infrastructure provides a set of generic services that process the released implementation model, provide access to information, and connect to resources. In addition, the integrating infrastructure hides the heterogeneity of the underlying manufacturing and information technology.

In the next two subsections, the two most important parts of CIMOSA—namely, the CIMOSA modeling framework and the CIMOSA integrating infrastructure—are explained in more detail.

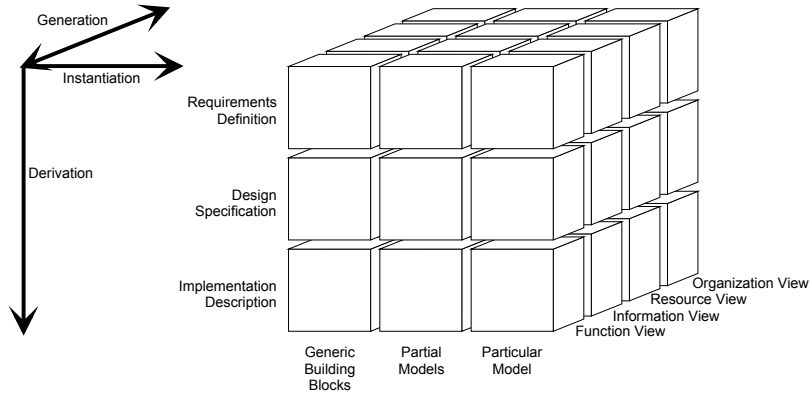


FIGURE 7.3 CIMOSA modeling framework.

CIMOSA Modeling Framework

In Fig. 7.3, the modeling framework is represented by the CIMOSA cube. The cube offers the ability to model different aspects and views of an enterprise (AMICE, 1993a,b). This three-dimensional framework has a dimension of genericity, a dimension of enterprise models, and a dimension of views:

- The dimension of genericity (and stepwise instantiation) is concerned with the degree of particularization. It goes from generic building blocks to their aggregation into a model of a specific enterprise domain.
- The dimension of modeling (and stepwise derivation) provides the modeling support for the system life cycle, starting from statements of requirements to a description of the system implementation.
- The dimension of views (and stepwise generation) offers the possibility to work with sub-models representing different aspects of the enterprise.

CIMOSA Integrating Infrastructure

The CIMOSA integrating infrastructure enables CIMOSA models to be executed, and it allows the control and monitoring of enterprise operations as described in the models. Furthermore, it provides a unifying software platform to achieve integration of heterogeneous hardware and software components of the CIM system. Applications use the generic services of the integrating infrastructure, so that they need no longer contain the specifics of the data-processing environment. This provides increased portability and flexibility of the applications and reduces the maintenance tasks considerably.

The integrating infrastructure consists of a number of system-wide, generic services. The business services control the enterprise operations according to the model. The information services provide for data access, data integration, and data manipulation. The presentation services act as a standardized interface to humans, machines, and applications. A product that is connected to the presentation services can be attached and removed without changing any other part of the information technology environment. Figure 7.4 displays the integrating infrastructure and the above-mentioned services. Other services are the common services and the system management services that

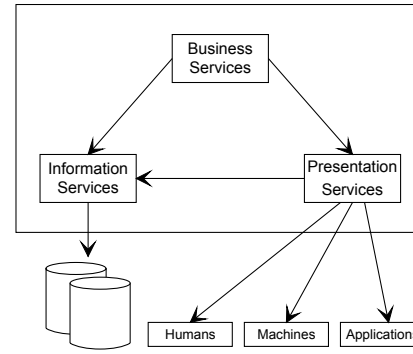


FIGURE 7.4 CIMOSA integrating infrastructure.

provide for system-wide data communication and facilities to set up, maintain, and monitor the components of the integrating infrastructure.

7.4 Application of CIMOSA at Traub

Introduction

The ESPRIT project VOICE (Validating OSA in Industrial CIM Environments) validated the CIMOSA results by the development of manufacturing control and monitoring solutions according to the CIMOSA concepts in three types of industries. The VOICE project consisted of vendors, system integrators, research institutes, and industrial partners. In each of the three industrial pilots, a different aspect of the manufacturing process was addressed. This section focuses on one of these pilots, namely the Traub pilot. Traub's motivation regarding the validation of CIMOSA was to examine whether CIMOSA is a useful tool to support the restructuring process of the existing manufacturing organization and to support the new organization with implemented information technology (Gransier and Schönewolf, 1995). Besides some mentioned references, the main sources of this section are VOICE meetings and reports.

Traub AG is a German manufacturer of machine tools. The products cover conventional machines as well as NC machines for turning and milling. The economic downturn in the market and the world-wide recession have forced all machine tool manufacturers to improve their cost situation and shorten the terms of delivery. Customer demands are characterized by an increased diversification of products, resulting in decreasing numbers of series production. Enormous sales efforts and the introduction of new products are not the only guarantee for survival. Enhanced demands to become more flexible and efficient forced Traub to reorganize its production department (Schlotz and Röck, 1995).

Figure 7.5 shows Traub's production control system before the reorganization. At that time, it consisted of a mainframe with the Production Planning System, an IBM RS/6000 for area control functionalities, and several cell controllers. Traub undertook global production planning for a year and a half in advance. In this planning, machine types, options, and number of products were incorporated. Every 10 days, a partial planning was made, which consisted of timed orders for design, purchase, and production control. It was possible to control the order flow from the area controller by releasing orders, standing in a "10 days order pool." Dedicated applications established the transmission of NC-programs between the area controller and the cell controllers. The worker at the NC-machine obtained a list of order data, and could transfer and edit NC-programs from and to the machines, and transmit the optimized programs to the area controller. Order progress and other monitoring data could be automatically sent from the machines to the area controller or by entering a message on a terminal. Monitoring information was sent to the mainframe at the production planning level from terminals on the shop floor located at several places near the machines.

One of the problems associated with the old situation at Traub was its flexibility to respond to changes in customer needs. Shorter delivery times with simultaneous reduction of stocks were already forcing Traub to shorten machining times and to link all processes in design, planning, and shop floor more closely. Even more, it frequently became necessary to rearrange plans at short notice because of changes in demand, machine downtimes, missing production facilities, urgent contracts with large customers, absence of staff, or rejects. Rescheduling of manufacturing jobs became difficult due to the limited feedback from the shop floor, and expenditures for short-notice re-work and machine modifications increased sharply.

In particular, the preparatory sectors were the areas where increased deadline pressure and the tendency toward smaller and smaller batch sizes added to planning complexity and expenditure. Most notably, tool management became problematic because the lack of up-to-date information concerning the availability of machine tools and their components also reduced the production capacity while operators waited for resources. Furthermore, production management had to cope with a growing amount of specialized tools necessary to produce individually designed machine parts. There were nine machine centers, that each

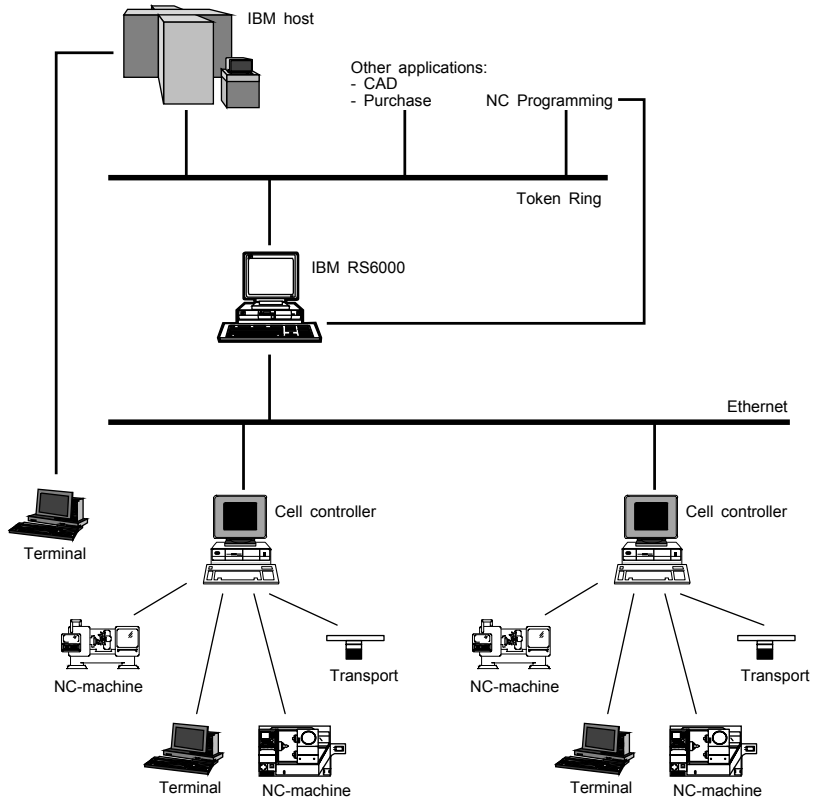


FIGURE 7.5 Traub's production control system before the reorganization.

having a stock of approximately 120 tools. Each tool consists of eight to ten components. At that time, there was no registration of which tool or component was in which machine (Schönewolf et al., 1992).

Reorganization Objective

The most important target of the reorganization was to optimize the order throughput time. To achieve this, a number of changes had to be implemented related to the cooperation on optimized order scheduling between the area controller and the cell controllers. Other changes concern tool management, monitoring data acquisition, etc. In this sub-section, only a few major changes are discussed.

The old, process-oriented manufacturing departments such as turning, milling, and grinding were changed for a more flexible organization. The new organization is based on the principles of cellular manufacturing, which involves the creation of clusters of machines that are designed and arranged to produce a specific group of component parts (Black, 1983). In particular, Traub expected the cellular manufacturing principles to result in a decrease in setup times.

An area control system was required that had to perform the fine planning of manufacturing device utilization under rescheduling conditions with the help of a production scheduling simulation tool. This area control system had to be linked enterprisewide with existing production planning, CAD, and CAM applications. It had to control not only the manufacturing processes, but also the delivery of material, tools, and equipment. Furthermore, it had to take into account the actual status of each NC-machine. This information had to be transferred online—without any influence of the working people—directly from the machine controller to the planning application of the area controller.

Fine-planning had to be supported by efficient tool management to get a tool-optimized order sequence to decrease the setup time for each machine. This system would provide information on the location

and states of tools by means of a tool identification system and a central database with tool information. It was necessary to integrate the tool logistics application with the entire existing infrastructure (Schlotz and Röck, 1995).

Requirements Definition

Traub defined requirements in a functional sense for the application to be developed, and in a technological/physical sense for the application's underlying infrastructure. In other words, requirements were defined for both the functional and technology aspects of the various components in the manufacturing system. In addition, requirements were imposed on the reengineering process, influenced by financial and time aspects.

Traub described its requirements from a functional point of view in terms of a scenario. In this scenario, the needed functions and information flows, and their places in the total manufacturing system, were outlined. For example, part of the scenario for the fine-planning activity (precision planning) was described as follows.

"The work sequences released are assigned to the individual machines during the automatic precision planning. During a planning sequence, only the machine capacity is taken into account. Starting from the terminating time given by the order pool system, timing is done backwards. If the calculated starting time lies in the past, the timing is done forwards while considering the transition time. If the given terminating time is not observed during this forward scheduling, this is explicitly displayed in the system. No alternative machine groups are considered in the automatic precision planning" (Schönewolf et al., 1992).

Requirements for the technology to be used were given as well. Besides Traub's current needs, requirements took into consideration the existing manufacturing system, strategic aspects such as standards, the factory environment, and production process constraints. Traub mainly defined its requirements of the infrastructure in terms of the CIMOSA integrating infrastructure; for example:

- "Multiple machines must be connected to a homogeneous system (presentation services).
- Connectivity to multiple databases on heterogeneous networks must be achieved from different kinds of computer systems (information services).
- The network must be transparent (common services)" (Schönewolf et al., 1992).

Architectural Design

After requirements were defined, the design of the CIM system commenced. The design activities of Traub's reorganization project could be distributed over two phases: architectural design and detailed design. In the architectural design phase, the system's functional and technology architectures were defined, supported by the CIMOSA framework. In the detailed design phase, the system was worked out in more detail, based on the defined architectures.

Area control was positioned between the production planning level (not in the scope of the reorganization project) and the shop floor. As an autonomous decision center in a distributed order planning structure, it processes the order pool coming from the production planning system, and performs scheduling and management tasks for machine-level planning. The incoming orders are scheduled for a 10-day period on the various machine groups. For each machine group, the area controller performs a daily planning to achieve an optimized order schedule with time slices of 2 days. The new order sequence on a machine is calculated on the basis of the available tools and the list of needed tools, which is extracted from the NC programs for the new orders. Tools available at NC machines or in the tool store can be requested. Tool handling is supported by the tool management process.

Being intermediate between production planning and shop floor, area control not only has to provide the shop floor with orders, but also allows feedback of information from the shop floor to the production planning system. Then, this system can use data that reflects the actual situation in the shop floor to optimize the planning process. Based on online messages from the cell control level, the area controller also supports processing and visualization of data such as the actual status of orders.

The scheduled orders in the time frame of 2 days are sent to the cell controller for execution. Tool management has allocated the tools required for the orders, and the NC programs are downloaded from the NC program server. Subsequently, the cell controller acts as an autonomous decision center, responsible for the execution of the orders, the collection of machine and operational data, and monitoring the shop-floor processes.

Traub defined its production control and tool management system by means of modeling this system and its environment with the CIMOSA requirements level. First, Traub built a user model with the Systems Analysis and Design Technique, because a tool and knowledge covering this modeling method was available. Later, Traub converted the user model to a CIMOSA model at the requirements definition level. For modeling at the requirements level, Traub used the modeling tool "GtVOICE," which is described by Didic et al. (1995). Traub specified the functions and their interrelations for both the tool logistic system and its direct neighbors. By modeling, Traub structured its system component functions, defined the components' allowed inputs and outputs, and specified the relations between these components. That is, by modeling at requirements definition level, a functional architecture was designed.

Figure 7.6 shows a CIMOSA model that presents a global view of Traub's production control functions. The principal control functions are captured in CIMOSA *domains*, the constructs that embrace the main enterprise processes. Figure 7.6 gives the identified domains and their relations in the form of *domain relationships*. Non-CIMOSA domains are parts of the enterprise that have not been considered for the moment and that could be detailed in the future, such as "Purchase" and "DNC-Node," or that are closed applications that cannot be described, such as the "Area Control" application. Usually, these applications are legacy systems.

Traub also made some models that showed more details than Fig. 7.6. Figure 7.7 is a partial refinement of the previous figure, showing domains and *domain processes*. CIMOSA represents main enterprise functionalities as *domain processes*. CIMOSA offers a modular modeling approach; the system can be extended with new domain processes, and system modifications can be limited to few domain processes. Enterprise operation is structured into a set of interoperating domain processes exchanging results and requests. They encapsulate a well-defined set of enterprise functionality and behavior to realize certain business objectives under given constraints. In the Traub case, examples of concurrent processes are the

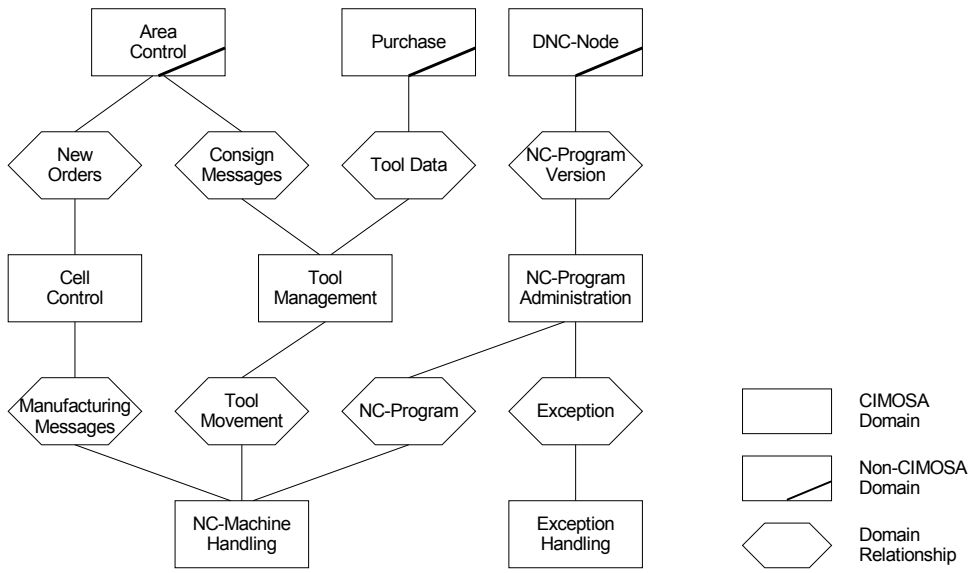


FIGURE 7.6 Overview of Traub's production control functions. (Reprinted from Zwegers et al., *Evaluation of Architecture Design with CIMOSA,* Figure 2, Elsevier Science, 1997. With permission.)

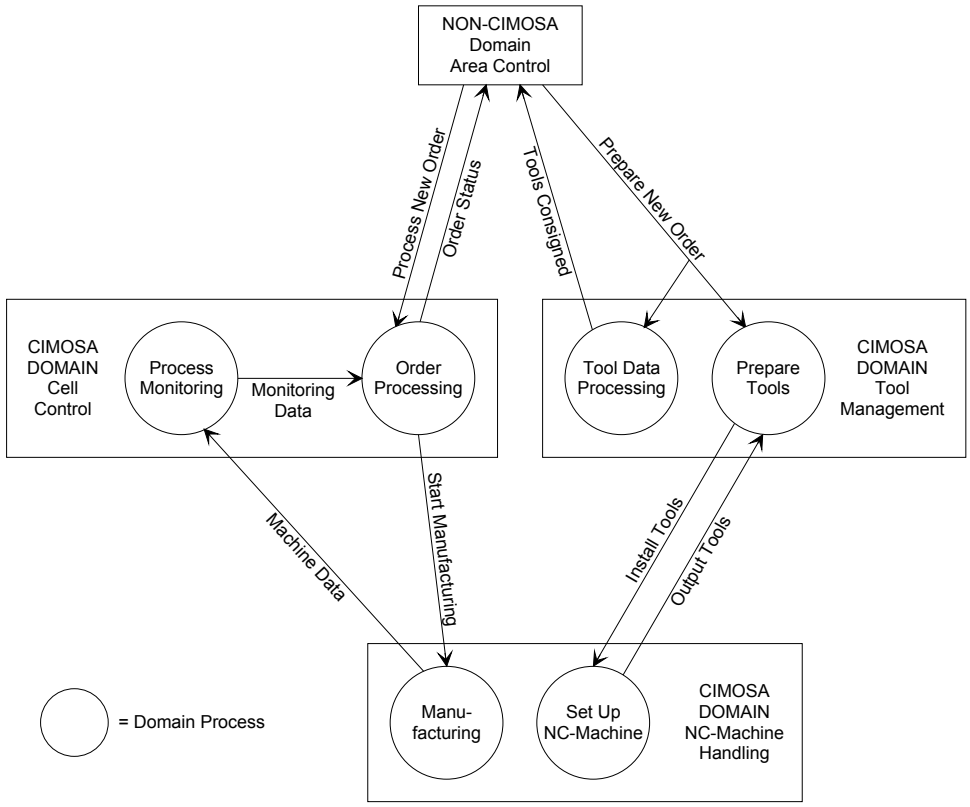


FIGURE 7.7 Partial refinement of Traub's domains. (*Source*: From Zwegers et al., *Evaluation of Architecture Design with CIMOSA*, Figure 3, Elsevier Science, 1997. With permission.)

processing of tool data and the preparation of tools. These processes are represented in domain "Tool Management" by two independent domain processes, namely "Tool Data Processing" and "Prepare Tools." Note that the domain processes in Fig. 7.7 influence each other's behavior.

Domain processes are the root of the decomposition tree. They employ *business processes* that are in the middle of the tree. The leaves are called *enterprise activities* and are employed by business processes or, if there are no business processes, by domain processes. The behavior of a certain business or domain process is defined by rules, according to which enterprise activities belonging to this process are carried out. Enterprise activities represent the enterprise functionality as elementary tasks, and they are defined by their inputs, outputs, function, and required capabilities. Fig. 7.8 presents a part of the behavior of domain process "Order Processing." For clarity, some relationships have been deleted. Events, which are either received from or sent to other domain processes, are represented by a Z-shape; enterprise activities are shown as boxes labeled "EA." Note that there are no business processes specified for this domain process. The large triangles indicate behavioral rules according to which enterprise activities or business processes are carried out. Fig. 7.8 was constructed using the GtVOICE tool.

Note that the requirements definition level of the CIMOSA modeling framework is used at architectural design activities and not during the requirements definition process. The reason is that CIMOSA does not support a "true" definition of requirements in the sense as described in the previous subsection. Instead, the CIMOSA requirements definition level offers support in structuring a manufacturing system, that is, in defining a functional architecture.

Along with defining a functional architecture, Traub also outlined a technology architecture that was influenced by the existing infrastructure. When defining functional components, one immediately

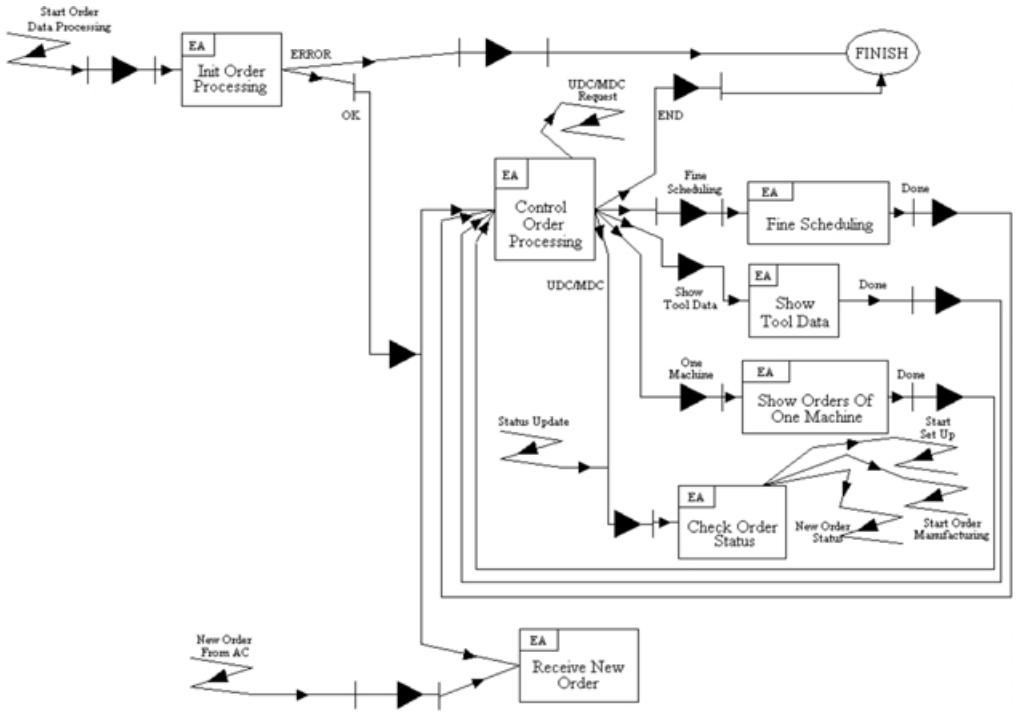


FIGURE 7.8 Process behavior of domain process "Order Processing" (partly). (*Source*: From Zwegers et al., *Evaluation of Architecture Design with CIMOSA*, Figure 4, Elsevier Science, 1997. With permission.)

maps these components on technological ones; the functional architecture is depicted on the technology architecture. For example, when a designer defines a control function, he/she decides to execute this function by a certain type of software module. In addition, the designer determines the interaction of this component with other technology modules.

Together with its partner, FhG-IPK (Fraunhofer-Institut für Produktionsanlagen und Konstruktionstechnik) in Berlin, Traub built a testbed for demonstrating the viability of the new production concepts. Traub initially defined three production management domains that it considered in its manufacturing system (or rather in its testbed), namely tool management, order planning ("cell control" in Fig. 7.6), and machine handling. Subsequently, Traub distributed these three functions over an area controller, a cell controller, and attached machines.

The technology architecture of the testbed is shown in Fig. 7.9. The area control level comprised a DecStation 5100 for long-term order planning and tool management. The area controller's user interface was implemented via the X.11 protocol on a terminal that was connected to the area controller via TCP/IP on Ethernet. Communication with the cell controller was established via MMS. The cell controller was a 486 PC running OS/2, enabling cell controller applications to run in a multi-tasking environment. A MAP network connected the cell controller with shop-floor devices. The cell controller received orders from the area controller, after which it processed the orders, controlling the shop-floor devices via MMS. These devices consisted of a robot controller connected to an industrial robot, a Traub NC-machine, and a PLC that controlled a conveyor and a round-table. Whereas the PLC had a MAP interface and connected directly to the MAP network, the NC-machine and the robot controller communicated with the network via a "protocol converter." The converter presented functionalities of both the NC-machine and the robot controller as MMS virtual machine devices to the MAP network. The machine and the robot controller connected to the protocol converter via V.24/LSV2.

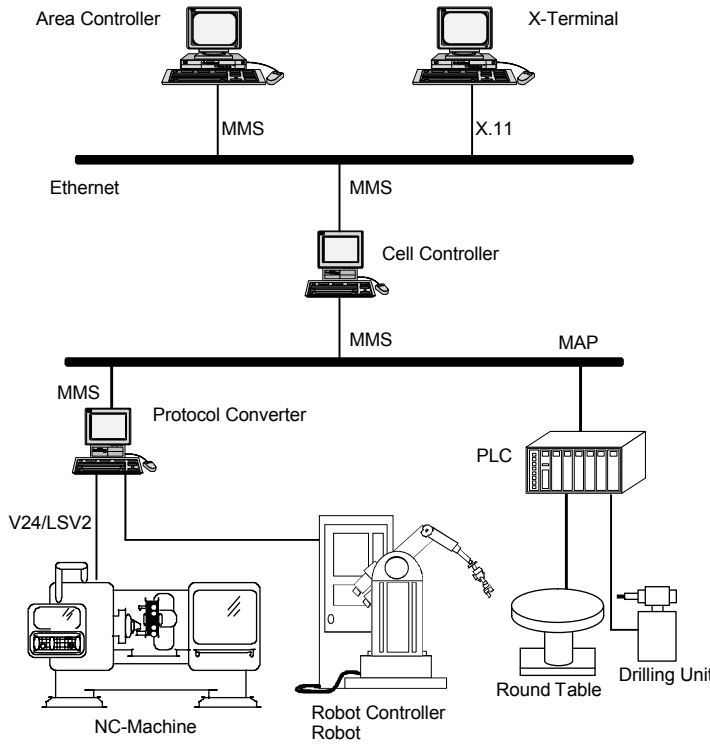


FIGURE 7.9 Technology architecture of Traub's testbed.

Detailed Design

In the second design phase, Traub specified its production control system in more detail, elaborating on the architectures that were defined in the architectural design phase. Both architecture definitions were further decomposed and worked out into detailed system specifications.

By means of the CIMOSA design specification level, a designer is able to detail a manufacturing system's functionality, taking the model at requirements level as the starting point. The requirements definition modeling level supports a user in the definition of his/her system's functional architecture, whereas the role of the design level is to restructure, detail, and optimize the required functionality in a consistent model. System optimization can be supported by simulation, taking all business and technical constraints into account. By specifying a model at the design level, a designer describes the full functionality of a CIM system, while staying within the definition of the functional architecture. If the model at the design specification level reveals inconsistencies, or the model lacks optimization, it might be necessary to adjust the model at the requirements definition level.

In addition to detailing the functionality of the system, the designer specifies the technology to be employed to achieve the required system functionality. Simply stated, CIMOSA prescribes that for each of the most detailed specified functions, called *functional operations*, a specified resource is assigned that provides the required capabilities. The prime task of the design specification modeling level is to establish a set of (logical) resources that together provide the total set of required capabilities. Some of these required capabilities are offered by the generic services of the integrating infrastructure.

For example, Traub made a model at the design specification level, elaborating on the previously made model at the requirements definition level. Traub specified the required information structure, it defined its most elementary functions, and it assigned resources to these functions. During the creation of the models, the function and information views were used extensively, whereas the resource and organization views

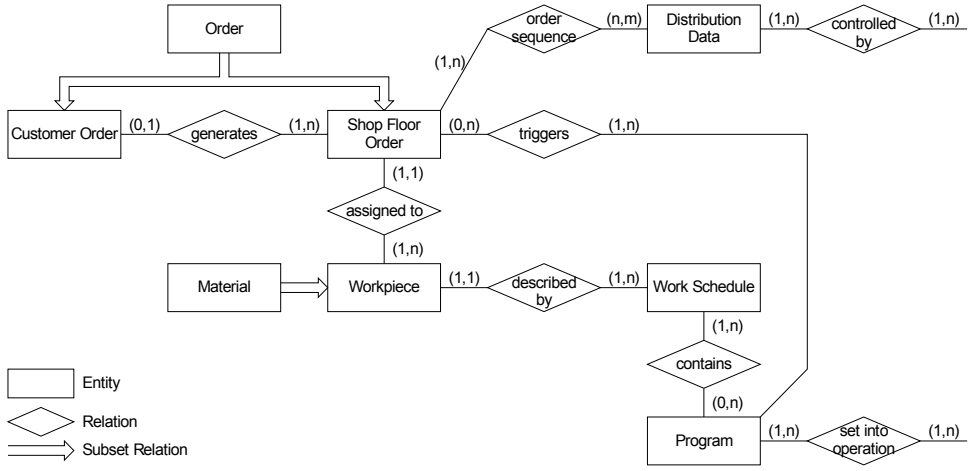


FIGURE 7.10 Model at particular design specification level, information view (partial). (*Source*: From Zwegers et al., *Evaluation of Architecture Design with CIMOSA*, Figure 5, Elsevier Science, 1997. With Permission.)

were barely addressed. It was not sensible to consider other factors such as responsibility (specified by constructs of the organization view) before Traub was satisfied with the new control structure regarding functionality and information flow. Part of the specified information structure is depicted in Fig. 7.10.

A refinement of specified functions is accompanied by a refinement of the technology that provides the desired functions. An enterprise looks for adequate products, either commercial ones or user developed. The technology components, which are defined in the architectural design phase, are specified completely. The definitions of the CIMOSA integrating infrastructure support this specification. Then, the products that fulfill the design specifications are selected, considering enterprise policies and constraints. CIMOSA states that the final build/buy decisions should result in a model that describes the implemented system. Appropriately, CIMOSA calls this model the implementation description model. However, because the AMICE consortium defined this part of the modeling framework after Traub reorganized its production department, Traub does not have any experience with it.

To implement a prototype, Traub identified candidates that might help to implement software modules or that might be used as complete software modules within the testbed. From the system specification and the analysis of possible candidates, Traub defined products, toolkits, and tools to implement the functionalities of the area controller and the cell controller. Furthermore, network interfaces were adopted and products fulfilling integrating infrastructure services were chosen. For example, Oracle 6 was selected as the product that had to provide the information services. It was connected by an SQL gateway to EasyMAP, which was used to provide the testbed's communication services. The communication services are part of the common services. In a later stage, FhG-IPK's communication platform was chosen in order to offer the desired communication services to the operational system.

Implementation

The final phase contains the implementation and release for operation of the specified CIM system. Implementation is based on the results and decisions of the previous phases.

Implementation activities comprise those tasks needed to bring the system into operation. During the implementation phase, Traub procured and built the necessary new physical components. Note that the word "physical" should not be taken too literally; in software engineering, for example, there are no such things as "physical components." The components were tested on the testbed and the correctness of the underlying logical model was verified. Traub decided to implement its physical system in an evolutionary

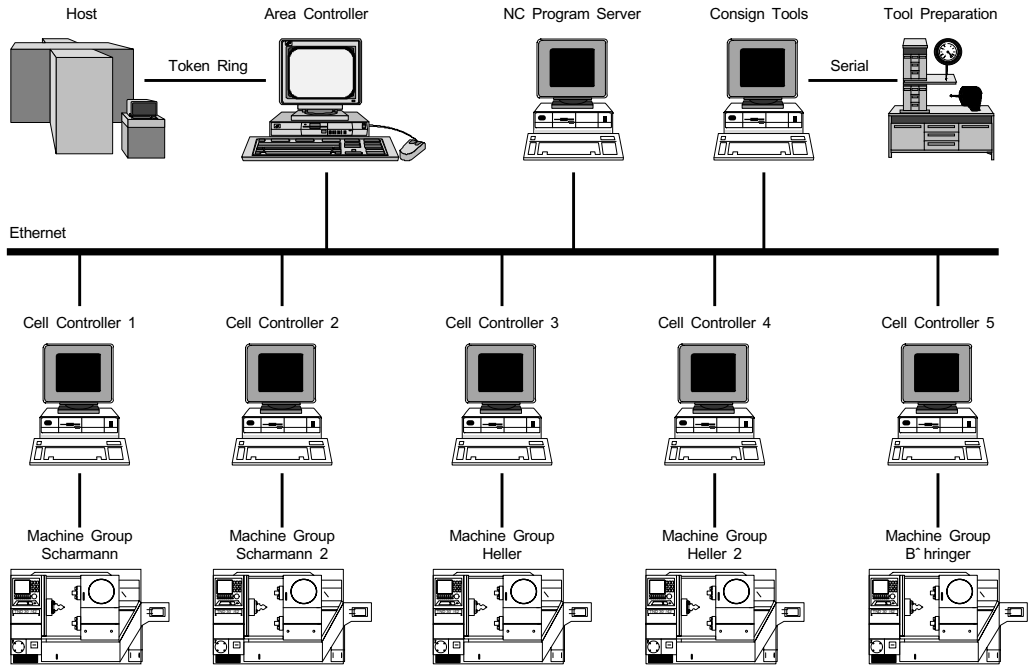


FIGURE 7.11 Technology architecture of Traub's implemented production control system. (*Source*: From Schlotz, C., and Röck, M., *Reorganization of a Production Department According to the CIMOSA Concepts*, Figure 7, Elsevier Science, 1995. With permission.)

way, with stable intermediate forms. In this way, Traub hoped to achieve a stable implementation of all products and their interactions. When the system passed the tests with satisfactory results, the system was prepared for transfer to production.

Traub felt that user acceptance was a major concern. It realized that the complex changes of the manufacturing system had to be put carefully to the operators in the production environment. Therefore, after testing single components as prototypes in the test environment, training sessions for the users of the area controller and the cell controller were held. To make the workers familiar with the new system, parts were transferred to the production without being integrated in the existing system. In addition, the graphical user interface was adapted according to users' needs, which was done to get a greater acceptance. When Traub believed that the operators were ready to use the new technologies, the new components were integrated in the existing system and released for operation.

Finally, the accepted production control system was released for operation. Figure 7.11 shows the technology architecture of Traub's production control system as implemented during the reorganization project.

The Role of CIMOSA in Traub's Reorganization Process

During the reorganization of Traub's production department, the role of CIMOSA was twofold: the CIMOSA modeling framework assisted the definition of a functional architecture, whereas the CIMOSA integrating infrastructure supported the design of the technology architecture of the control system's infrastructure.

The CIMOSA models allowed Traub to analyze and redesign its functional control architecture. During architectural design, Traub was able to acquire knowledge of its current manufacturing control system by means of modeling at the requirements definition level. This knowledge was needed to analyze the existing system. Then, the model was changed to take required modifications into account. By means of the specification of operational and control functions, their inputs and outputs, and interfaces between functions, a functional architecture was defined.

The definition of a sound functional architecture, upon which the further design of the system is based, is one of the keys to business integration. CIMOSA takes a modular approach to integration; that is, enterprise operation is modeled as a set of cooperating processes that exchange results and requests. Only this exchanged data needs to have a representation that is common between the cooperating processes. By employing models for identification, analysis, and coordination of existing and new functions, required functional architectures can be designed that define modular components and their interfaces. When the detailed design and implementation of the CIM system follow the architectural specifications, the requested integration of business processes will be obtained. This way, the required level of business integration is achieved, which is reflected by the coordination of operational and control processes.

CIMOSA aims to support application integration by its integrating infrastructure. Cooperation between humans, software programs, and machines must be established. Whereas business integration can be regarded as the integration of functional components, application integration affects the integration of technology components. The integrating infrastructure aims to provide services that integrate the enterprise's heterogeneous manufacturing and information systems to satisfy the business needs as identified with the help of the modeling framework. The CIMOSA specification of the integrating infrastructure can be seen as a reference model. Components designed and implemented according to this reference model should provide desired features such as interoperability, portability, connectivity, and transparency.

Due to the immature specification of most integrating infrastructure services, Traub only considered the communication services. An example of an implementation that fulfills the CIMOSA specifications is the communication platform developed by the FhG-IPK. This platform has been developed according to the needs of Traub and the other two industrial VOICE partners, and was based on the communication services of the CIMOSA integrating infrastructure. Lapalus et al. (1995) provide additional information about the communication platform and application integration.

7.5 The Practical Value of CIMOSA

In this section, the practical value of CIMOSA is central. Its theoretic value is undoubtedly large, because many ideas, developments, and products are inspired by it. In fact, many contemporary products seem to be largely influenced by the CIMOSA concepts. Its value for industry is evaluated by means of the sets of essential and desirable elements that reference architectures should be accompanied by. Section 7.2 states that a reference architecture should be accompanied by modeling languages, a modeling methodology, and enterprise engineering methodologies. Furthermore, it is desirable that a reference architecture be supported by reference models, modeling tools, and generic enterprise modules. A special feature of CIMOSA, as compared to other enterprise reference architectures, is that it aims to execute models; that is, to control and monitor enterprise operations as described in the models.

On the Eligibility of the Reference Architecture

Architectural concepts such as hierarchy, views, and abstraction are represented in CIMOSA's modeling framework. The functional specifications are hierarchically decomposed by means of the domain process, business process, and enterprise activity constructs. The modeling framework provides an open set of (at the moment) four views to focus on different enterprise aspects. As for abstraction, going from the requirements definition level, via the design specification level, to the implementation description level, can be considered as moving from a focus on the functional, via the technology, to the physical domain. However, the level of detail increases as well. That is, CIMOSA relates a low level in the design process to implementation details. It does not support the specification of a true technology architecture, but only the detailed specification of technology modules.

The CIMOSA modeling framework supports designers during the specification and analysis of functional architectures of production control systems. A reference architecture should allow the specification of functional control architectures, either by refinement of reference models or by design from scratch. Reference models were not used during the Traub reorganization project; instead, models were made

from scratch. These models proved their usefulness by revealing some bottlenecks and inconsistencies in the organization (Schlotz and Röck, 1995).

Zwegers et al. (1997) show that most characteristics of functional control architectures can be specified by means of the CIMOSA modeling framework. Domain processes, events, and object views are adequate constructs to specify concurrent processes and the exchange of information and products between these processes. By defining domain processes and establishing relations between them, any type of control architecture can be modeled. In addition, the modular nature of domain processes makes CIMOSA models extensible and modifiable.

On the Complexity of the Modeling Framework

Industry—almost unanimously—regards the modeling framework as too complex. In Traub's view, the high complexity of the CIMOSA modeling framework requires well-trained engineers (Schlotz and Röck, 1995). Traub practically used only the framework's function and information view; the resource view was barely addressed and the organization view was not used at all. The organization view, for example, was seen as a view to manage systems, not to design them. In addition, the great number of constructs and their sometimes-ambiguous definitions hamper a practical application. A tool was needed to unambiguously define the meaning of constructs.

On the Novelty of the Model Creation Processes

A modeling framework should be accompanied by guidelines, called "model creation processes" by CIMOSA. A designer must be guided to navigate through the modeling framework in a consistent and optimized path, to ensure complete, consistent, and optimal models. During the modeling process for the Traub application, no such modeling guidelines were available. Recently, some guidelines have been defined, but their practical value for industry has not yet been established.

At the moment, CIMOSA provides a methodology to guide users in the application of its modeling framework. Zelm et al. (1995) describe this so-called CIMOSA Business Modeling Process. As for modeling at the requirements definition level, for example, the top-down modeling process begins with the identification of domains, the definition of domain objectives and constraints, and the identification of relationships between the domains. Afterward, the domains are decomposed into domain processes, which are further decomposed in business processes and enterprise activities. The modeling process at the requirements definition level ends with some analyses and consistency checking.

More recently, new methodologies have been proposed that aim to be improvements or alternatives to the CIMOSA Business Modeling Process. For example Reithofer (1996) proposes a bottom-up modeling methodology for the design of CIMOSA models. He claims that the CIMOSA Business Modeling Process can hardly be focused on processes and activities that are relevant to solve a concrete problem. His bottom-up modeling approach should not have this disadvantage. In addition, Janusz (1996) asserts that existing CIMOSA models of particular enterprises are not complete, not consistent, and not optimal. Also, these models often describe only functions or sub-processes limited by department borders of an enterprise. Therefore, Janusz developed an algorithm that filters out process chains of an existing CIMOSA model. Using the algorithm, the completeness and the consistency of the considered process chains can be checked.

On the Inadequacy of the Enterprise Engineering Methodology

CIMOSA lacks a "true" engineering methodology, which provides instructions of how to go about an enterprise integration project or program. Williams et al. (1994a) notice that CIMOSA does have a "life history" for CIM systems (namely, the CIMOSA system life cycle) but that this description has not been extended into a "true" methodology. Zwegers and Gransier (1995) give a description of the engineering approaches adopted by the three industrial partners of the VOICE project, which used CIMOSA during their reengineering trajectories. However, these engineering approaches have not been extended into a methodology either. Possible users are not supported by knowledge on how to apply CIMOSA to carry out integration projects. This point cannot be over emphasized; an enterprise reference architecture without matching methodology defeats its own purpose.

On the Availability of the Modeling Tools

A modeling language should be supported by a software tool. During the creation of models, problems might occur regarding the size and complexity of the architectural models. Models become too big to be overlooked by the user—and they become inconsistent. Furthermore, the modeling process without tool support is tardy, and maintainability and extendibility of the models are jeopardized.

Some tools have been developed for modeling with CIMOSA, such as GtVOICE (Didic et al., 1995) and SEW-OSA (Aguiar et al., 1994). GtVOICE was developed by the VOICE project. This tool ensures model consistency and reduces modeling time by easy model modification and extension. In addition, it provides non-ambiguous interpretation of CIMOSA constructs and a uniform way to present models among CIMOSA users. Finally, GtVOICE makes communication with other tools feasible, such as an SQL data server and a rapid prototyping toolkit.

SEW-OSA (System Engineering Workbench for CIMOSA) was developed at the Loughborough University of Technology in England. It combines the CIMOSA concepts with Petri net theories, object-oriented design, and the services of an infrastructure called CIM-BIOSYS (Aguiar et al., 1994). Both SEW-OSA and GtVOICE support the design of CIMOSA models according to the CIMOSA Business Modeling Process as described by Zelm et al. (1995).

On the Absence of Reference Models

Perhaps most important for industry are guidelines that support designers to make the right architectural choices; that is, choices that result in flexible, effective systems. Industry needs guidelines for the architectural design of systems, to discard inadequate options early in the design process and to enable the selection of the best options. Clearly, such guidelines are not present at present. However, it is not an objective of CIMOSA to provide such a prescriptive methodology. The CIMOSA modeling framework aims to support system designers with descriptive modeling of enterprise operation; it is a descriptive rather than prescriptive framework.

Nevertheless, the CIMOSA modeling framework offers the ability to develop reference models with its partial modeling level. Best-practice reference models are the encapsulations of the prescriptive guidelines mentioned above. As such, they are recognized as major tools to support CIM system development projects (Faure et al., 1995). They should be based on architectural principles, such as modularity, structural stability, and layers. Aguiar and Weston (1995) signal the need to populate workbenches with a library of reference models. However, virtually no CIMOSA-compliant reference models are currently available.

CIMOSA lacks the guidelines and reference models needed to make a transition from requirements to specification. After all, it prescribes how to make a specification of a system, but is does not prescribe how to design the system. It offers the architecting concepts, but not the architecting principles nor the business knowledge. It gives the ruler and compass to draw a house, but it does not give the construction theory.

On the Promises of the Integrating Infrastructure

The promises of the CIMOSA integrating infrastructure appear too good to be true. Integration of hardware and software components, model execution, vendor independence, reduced maintenance, and increased application portability and flexibility appeal to companies facing a heterogeneous world. However, the CIMOSA specifications of the services as used by Traub (AMICE, 1993b) reveal many gaps. In addition, no commercial products supporting the CIMOSA specifications are currently available. Nevertheless, enterprises appear to be more attracted by application integration promised by the integrating infrastructure than by business integration as actually supported by the modeling framework.

On the Illusion of Model Execution

After time, models produced during an enterprise integration project are almost certain to lose their validity. Enterprise integration is an ongoing process rather than a one-off effort. However, there is little chance for an enterprise to maintain an up-to-date picture of itself as time goes by. Much of the effort initially invested in modeling an enterprise's processes is lost as reality diverges from those models (Bernus et al., 1996).

Traub, for example, foresees a considerable effort in keeping its models consistent with the implementation, also due to the complexity of the CIMOSA modeling framework (Schlotz and Röck, 1995). Note that the consistency aspect is not a CIMOSA-related issue, but rather a common problem for all enterprise reference architectures.

To remedy this obstacle, enterprise models must actually be used to drive operation control and monitoring rather than being kept on the shelf. The transition from specification to an operational system requires an infrastructure that supports "model execution" and therefore must consist of what CIMOSA calls "business services." The realization of the aims behind the business services as originally conceived might just as well prove to be an illusion for a long time. Bernus et al. (1996), for example, regard the achievement of business services as a goal for the future.

CIMOSA fails to fulfill the objectives of model execution. The result of the specification phase (i.e., documentation in the form of models) is useful for (one-time) business integration. Without a real integrating infrastructure with services such as the business services, however, it is not sufficient to fulfill the objectives of the enterprise integration philosophy. Applying CIMOSA does not result in operational systems, let alone in flexible, adaptive, efficient systems; the translation from the models into a real system must be made. Given the fact that it does not offer guidelines and reference models that support designers in the transition from requirements to specification, CIMOSA should merely be seen as a framework for the generation of documentation.

7.6 Two Other Enterprise Reference Architectures

Williams et al. (1994b) claimed that there were only three major enterprise reference architectures known at that time. This section discusses the other two enterprise reference architectures—in addition to CIMOSA—that Williams et al. referred to: namely, the GRAI Integrated Methodology and the Purdue Enterprise Reference Architecture.

GRAI/GIM

The GRAI laboratory of the University of Bordeaux, France, has been rather active in the field of enterprise reference architectures. Along with developing its own ideas, the GRAI laboratory has contributed to the ESPRIT projects IMPACS and AMICE. Here, the main elements of what has become known as the GRAI Integrated Methodology (GIM) are described: namely, a "global model," a modeling framework, and a structured approach to guide the application of the methodology.

The global model describes the invariant parts of a CIM system: the subsystems, their relationships, and their behavior. The global model (sometimes called the "macro reference model" or "reference model") is based on the concepts of three activity types and their corresponding executional subsystems. These three subsystems are:

1. Physical subsystem, which performs the activities of product transformation using human and technical resources
2. Decisional subsystem, which guides production toward its goals
3. Informational subsystem, which feeds the other subsystems with information

Sometimes, a fourth subsystem is distinguished, namely the functional subsystem (Doumeingts et al., 1987; 1992).

The modeling framework uses $IDEF_0$ formalisms to model the physical and functional subsystems, GRAI formalisms for the decisional subsystem, and MERISE formalisms for the informational subsystem. The GRAI formalisms are described below; for the other formalisms, the reader is referred to (Doumeingts et al., 1995). The GRAI formalisms consist of the GRAI grid and the GRAI nets. The GRAI grid allows one to model a decision system. It is displayed as a table-like frame, and it uses a functional criterion to identify production management functions and a decision-cycle criterion to identify decisional levels.

Each function is decomposed into several levels according to the decision horizon *H* and revision period *P*. A decision period is a time interval through which decisions are valid; a revision period is a time interval at the end of which decisions are revised. The building block of a grid is a decision center that is the intersection of a production management function and a decisional level. Decision centers are mutually connected by decision links and information links. The GRAI nets allow one to model the various activities of each decision center identified in the GRAI grid. The results of one discrete activity can be connected with the support of another discrete activity. Because this is done for each decision center, the links between decision centers are shown (Chen et al., 1990).

GIM's structured approach aims to cover the entire life cycle of the manufacturing system. Figure 7.12 shows that the approach consists of four phases: initialization, analysis, design, and implementation.

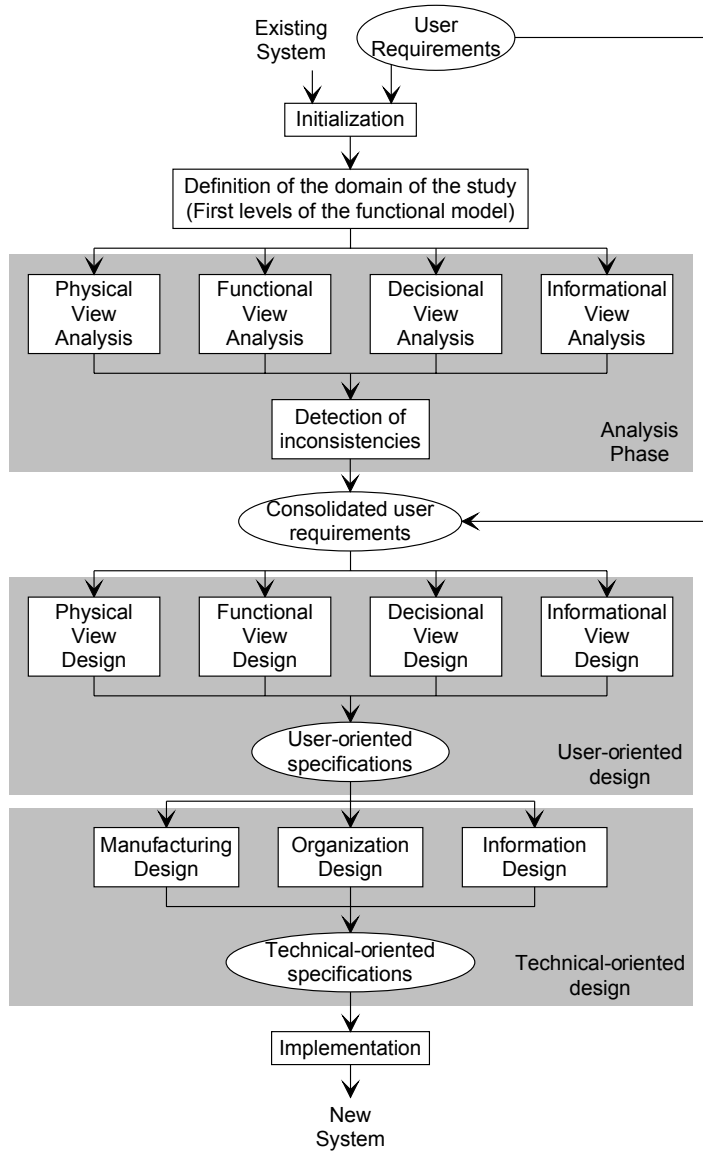


FIGURE 7.12 GIM structured approach. (*Source*: From Doumeingts et al., *Methodologies for Designing CIM Systems: A Survey*, Figure 20, Elsevier Science, 1995. With permission.)

Initialization consists of defining company objectives, the domain of the study, the personnel involved, etc. The analysis phase results in the definition of the characteristics of the existing system in terms of four user-oriented views. The design phase is performed in two stages: user-oriented design and technical-oriented design. User-oriented design employs the results of the analysis phase to establish requirements for the new system, again in terms of the four user-oriented views. Technical-oriented design consists of transforming the user-oriented models of the new system design to technical-oriented models. These models express the system requirements in terms of the required organization, information technology, and manufacturing technology. Finally, the new system is implemented (Doumeingts et al., 1995).

GRAI/GIM covers the whole life cycle of a manufacturing system, except the operation and decommission phases. Its four views differ from CIMOSA's; it introduces a decisional and physical view. However, the models of the physical view do not describe physical attributes; they describe functional attributes of physical elements. The four model content views used during the analysis and user-oriented design phases are translated to three implementation views during the technical-oriented design phase. Concerning modeling languages, GRAI/GIM is less formal than CIMOSA. After all, CIMOSA aims to achieve model execution, and that requires formal modeling languages. GRAI/GIM uses several known modeling techniques such as $IDEF_0$ and MERISE, and it developed the GRAI grids and nets. Although the GRAI laboratory has completed many projects with its modeling framework, there is no modeling methodology described in the literature. Obviously, this does not imply that there is no such methodology. GIM's structured approach offers an enterprise engineering methodology. However, this structured approach is focused on the initial phases of a system life cycle; GRAI/GIM mainly supports designers during the analysis and design phases.

As for modeling tools, there is no tool known in the literature that supports modeling with the GRAI/GIM modeling framework. The same applies to reference models. GRAI/GIM is based on a "global model," a kind of generic model of an enterprise. However, no reference models that encapsulate industry-type or area-specific knowledge are known. GRAI/GIM does not aim to provide generic enterprise modules such as parts of an integrating infrastructure.

PERA

The Purdue Enterprise Reference Architecture (PERA) and its accompanying Purdue Methodology were developed at Purdue University, (Nest Lafayette, IN). This university has also taken a leading role in the definition of reference models for computer integrated manufacturing.

The Purdue Methodology is based on an extensive Instructional Manual that guides the preparation of Master Plans. According to the methodology, an overall Master Plan is necessary before attempting to implement any CIM program. A Master Plan includes a CIM Program Proposal (i.e., a set of integrated projects whose completion will ensure the success of the desired integration of the enterprise). The Purdue Enterprise Reference Architecture provides the framework for the development and use of the Instructional Manual, the resulting Master Plan, and the ultimately implemented CIM Program Proposal. PERA is the glue that holds together all aspects of the CIM program (Williams, 1994).

Figure 7.13 shows that the Purdue Enterprise Reference Architecture is characterized by the layered structure of its life-cycle diagram. Starting with the Enterprise Business Entity, it leads to a description of management's mission, vision, and values for the entity under consideration. From these, the operational policies are derived for all elements of concern. Two—and only two—kinds of requirements are derived from the policies, namely, those defining information-type tasks and physical manufacturing tasks. Tasks become collected into modules or functions, which at their turn are connected into networks of information or of material and energy flow. Then, technology choices are made; the role of the human in the information architecture and in the manufacturing architecture is defined. The result of the technology choices are three implementation architectures: namely, the information systems architecture, the human and organizational architecture, and the manufacturing equipment architecture. After functional and detailed design, the life cycle goes through the construction and operation phases, after which it is disposed of (Williams, 1994).

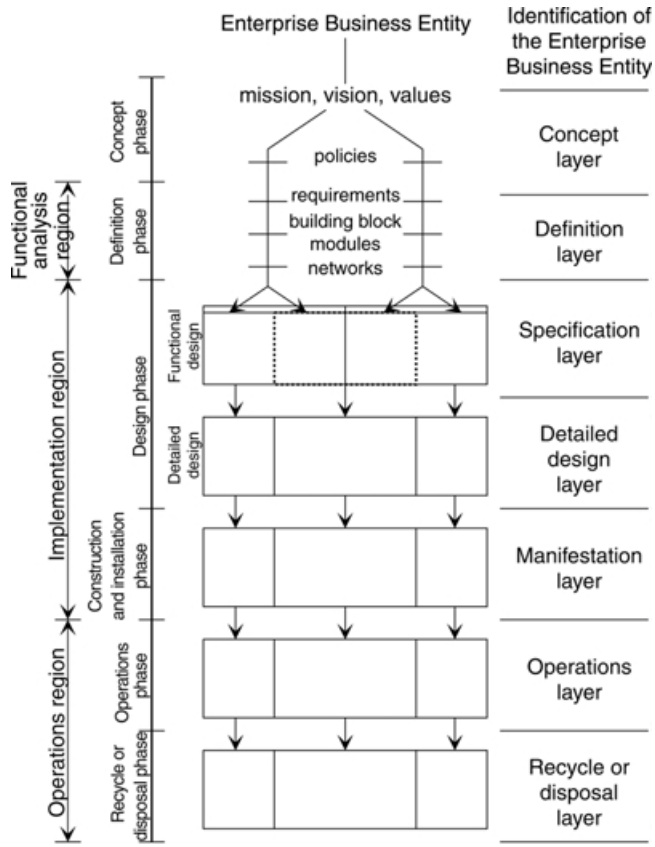


FIGURE 7.13 Phases and layers of the Purdue Enterprise Reference Architecture. (*Source*: From Williams, T., *The Purdue Enterprise Reference Architecture*, Figure 1, Elsevier Science, 1994. With permission.)

A specific feature of PERA is the emphasis it puts on the role of humans. By defining the functions that will be performed by humans, the information and manufacturing architectures are converted into the information systems architecture, the human and organizational architecture, and the manufacturing equipment architecture. Figure 7.14 illustrates that this definition involves three "lines." The automatability line shows the absolute extent of pure technological capability to automate tasks and functions, and is limited by the fact that many tasks and functions require human innovation and cannot be automated with presently available technology. The humanizability line shows the maximum extent to which humans can be used to implement tasks and functions, and is limited by human abilities in speed of response, physical strength, etc. The extent of automation line shows the actual degree of automation carried out or planned in the system. This third line defines the boundary between the human and organizational architecture and the information systems architecture on the one hand, and the boundary between the human and organizational architecture and the manufacturing equipment architecture on the other side. Its location is influenced by economic, political, social, and technological factors (Rathwell and Williams, 1996).

PERA explicitly takes into account the role of the human in a manufacturing system. In addition, it does not distinguish between model content views such as function, information, and resource, but rather between purpose views (information and manufacturing architecture) and implementation views (human and organizational architecture, information systems architecture, and manufacturing equipment architecture). The Purdue Methodology offers an enterprise engineering methodology that covers all phases of a system life cycle. Of the three enterprise reference architectures described in this chapter, PERA is clearly accompanied by the most extensive methodology. As for modeling languages, PERA offers only the task module construct. Information system tasks, manufacturing tasks, and human-based tasks

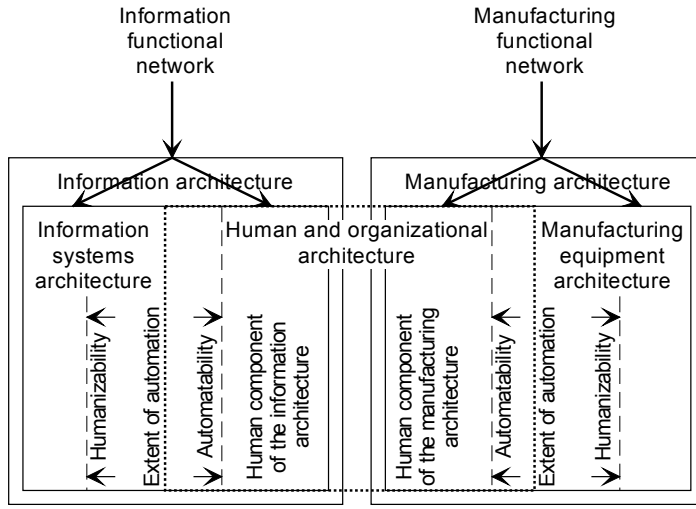


FIGURE 7.14 Humanizability, automatability, and extent of automation to define the three implementation architectures. (*Source*: From Rathwell, G.A. and Williams, T.J., Use of the Purdue Enterprise reference architecture and methodology in industry (the Fluor Daniel example), in: *Proceedings of the IFIP TC5 Working Conference on Models and Methodologies for Enterprise Integration*, Bernus, P. and Nemes, L., Eds., pp. 12–44, 1996. With permission from Kluwer Academic/Plenum Publishers.)

are modeled by means of this construct. By combining the various task modules into networks, a type of data-flow or material and energy flow diagram results. There are no modeling methodologies known.

In addition, no modeling tool is known in literature. Although the only construct is quite simple, models might become large and, without a tool, their accuracy over time is jeopardized. Earlier, the developers of PERA had defined a reference model for computer-integrated manufacturing (Williams, 1989). This reference model is restricted to the automatable functions of an enterprise, and all functions that might require human innovation are considered as external entities. PERA does not provide components of an integrating infrastructure or any other generic enterprise modules.

GRAI/GIM and PERA provide some solutions to some of CIMOSA's shortcomings. In particular, they provide well-defined engineering methodologies: GRAI/GIM offers its structured approach, and PERA is accompanied by the Purdue Methodology. For enterprise integration, however, it is necessary to keep an up-to-date picture of an enterprise as time goes by. None of the three reference architectures can guarantee that.

7.7 Baan's Enterprise Modeler

Recently, some of the concepts of enterprise integration have been implemented in tools for the configuration and implementation of enterprise resource planning (ERP) solutions. For example, the market leader in ERP applications, the German company SAP AG, has developed a tool called "Business Engineer." This configuration and implementation tool is based on the R/3 reference model. The R/3 reference model describes business processes that are most commonly needed in practice and that can actually be implemented with SAP's R/3 system (Keller and Detering, 1996). Business Engineer aims to streamline implementation of R/3, and to adapt an existing configuration to new needs or changed circumstances. It allows users to configure their own enterprise model, and to automatically tie the functionality of R/3 into the enterprise model (SAP, 1997).

In this section, the Enterprise Modeler, which is part of Baan Company's Orgware™ tool set, is described in detail. Baan Company is currently one of the world's largest suppliers of ERP solutions. It recognized that enterprises attempting to implement ERP systems went through a serial process of first modeling the enterprise's processes and organization, then implementation, and finally manual configuration of the actual system. The usual result of this static process is that customer investment in

modeling activities has not returned the expected value in the actually implemented system. This is because the statically defined system is no longer in line with the new needs of the business. Therefore, Baan Company developed Orgware, a set of tools and concepts that allows an enterprise to adapt its Baan ERP application real-time to changing market needs. Even more, it supports a fast implementation and optimization of a system (Huizinga, 1995).

Dynamic Enterprise Modeling (DEM) concepts comprise the foundation of Orgware. Basically, DEM comes down to developing a model of an enterprise and using this model to adapt the enterprise's processes. As such, it does not differ from the objectives of enterprise reference architectures. However, because Orgware is integrated with Baan's ERP product, there is a direct link between the models and their realization in the Baan software.

The Enterprise Modeler is one of Orgware's main components. Examples of other components are implementation methods and assisting IT services. A user makes four types of models with the Enterprise Modeler; namely, business control models, business function models, business process models, and organization models. In a business function model, a functional decomposition of business functions is presented. It is the starting point for process selection and configuration. A business process model describes formal processes and procedures, and is the basis for configuration of Baan software. An organization model is a description of the organizational structure of an enterprise. The business control model links the other three models, and it provides modeling teams as a starting point for modeling at a high level (Boudens, 1997). Figure 7.15 presents an overview of Dynamic Enterprise Modeling with the Enterprise Modeler.

A central feature of the Enterprise Modeler is its use of reference models. In a repository, functions, procedures, rules, and roles are defined. The rules connect functions and procedures to each other. From the components in the repository, reference models are assembled that are specifically designed by industry consultants for a particular industry sector. For example, in the reference organization model for engineer-to-order enterprises, an organizational structure is outlined that is based on the best practices of that type of industry. In the reference business function and process models, optimization relationships and roles are added respectively (Huizinga, 1995).

Models for a specific enterprise called "project models" can be instantiated from the reference models. Phases are added to the project function model that provide for a phased implementation of the Baan system; some functions are, for example, only wanted after optimization of the procedures. Employees

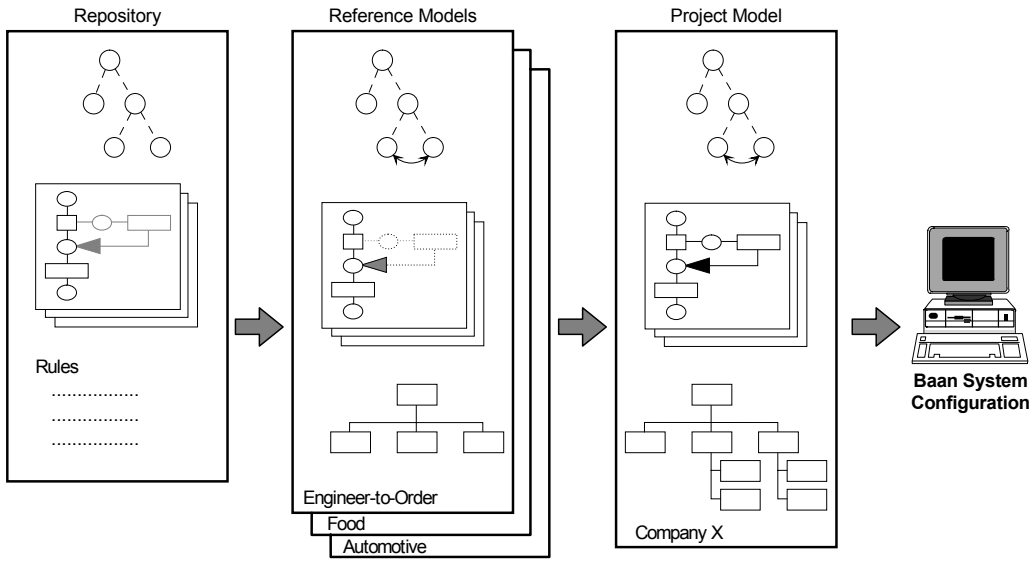


FIGURE 7.15 Dynamic Enterprise Modeling with the Enterprise Modeler.

are added in the project organization model. Subsequently, employees are linked to roles, thereby providing the connection between the process and organization models. Finally, a configurator generates user-specific menus and authorizations from the project models, and automatically configures the Baan system by setting parameters.

A clear advantage of Baan's Orgware product compared to enterprise reference architectures is its linkage with the underlying Baan ERP applications. Models made by the Enterprise Modeler mirror the implementation and configuration of the Baan system. Moreover, changes in the models are directly reflected in changes in the Baan system. Enterprise reference architectures, on the other hand, lack an integrating infrastructure with business services. Models made by these reference architectures might reflect the enterprise's current systems and processes, but changes in these models must be "manually translated" to changes of the modeled processes and applications.

Orgware's advantage of being linked with the underlying Baan ERP applications is also its disadvantage: it covers only Baan products. The Enterprise Modeler allows one to model manual procedures and applications by vendors other than Baan sessions. Clearly, it cannot automatically configure these other applications. Although ERP products tend to cover more and more, they do not comprehend the processes of an entire enterprise. Shop-floor control processes, for example, are not covered. In addition, the Enterprise Modeler does not model a company's IT infrastructure. For example, it cannot support distributed Baan applications. A product such as Orgware is a major improvement in the implementation of ERP applications and, as such, it provides a contribution to the realization of the enterprise integration philosophy. For the objectives of enterprise integration to become true, however, more is needed than that: model execution by an infrastructure with real business and presentation services.

7.8 Summary

Reference architectures for enterprise integration aim to provide the necessary frameworks with which companies might adapt their operation due to changing internal and external conditions. This chapter has investigate the practical value of these enterprise reference architectures for the integration of production systems. Three reference architectures have been studied; namely, CIMOSA, GRAI/GIM, and PERA. Their overlaps and differences can be identified by means of the GERAM framework. Reference architectures should at least be accompanied by engineering methodologies, modeling languages, and modeling methodologies. In addition, modeling tools, reference models, and some generic enterprise modules are desirable components.

CIMOSA strives for the facilitation of continuous enterprise evolution. It intends to offer support for business integration by means of its modeling framework, and for application integration by means of its integrating infrastructure. Starting with the reference architecture and possible reference models, a designer can make a CIMOSA model of its enterprise operations. The CIMOSA integrating infrastructure processes the released model, so that operations are executed according to the model.

CIMOSA was applied during a reorganization project at Traub AG, a German tool manufacturer. The modeling framework assisted Traub in the definition of a functional control architecture. Due to immature specification of the integrating infrastructure, only the specification of the communication services was helpful to Traub.

The CIMOSA reference architecture does not cover a full life cycle and lacks an accompanying engineering methodology. However, CIMOSA does provide an eligible, although quite complex, framework for the specification and analysis of functional architectures for production control systems. It prescribes how to make a specification of the system, but it does not prescribe how to design the system. In addition, a designer must personally make the translation from models to a real system. Therefore, CIMOSA should be merely seen as a framework for the generation of documentation.

GRAI/GIM and PERA provide some solutions to some of CIMOSA's shortcomings. In particular, they provide well-defined engineering methodologies. For enterprise integration, however, it is necessary to keep an up-to-date picture of an enterprise as time goes by, which is something the three reference architectures cannot guarantee.

A product that ensures up-to-date models of an enterprise is Baan Company's Enterprise Modeler. Based on repository and reference models, a designer makes models of its enterprise. Because the Enterprise Modeler is integrated with the Baan applications, it automatically configures these applications based on the models. Changes in the models are immediately reflected by changes in the configuration of the applications.

References

Aguiar, Marcos Wilson C., Ian Coutts, and Richard H. Weston. (1994). Model enactment as a basis for rapid prototyping of manufacturing systems. In *Proceedings of the European Workshop on Integrated Manufacturing Systems Engineering (IMSE '94),* Grenoble, Dec. 1994, 86–96.

Aguiar, Marcos Wilson C., and Richard H. Weston. (1995). A model-driven approach to enterprise integration. *International Journal of Computer Integrated Manufacturing,* Vol. 8, No. 3, pp. 210–224.

AMICE Consortium. (1993a). *CIMOSA: Open System Architecture for CIM.* Springer, Berlin.

AMICE Consortium. (1993b). *CIMOSA: Open System Architecture for CIM—Technical Base Line,* Version 2.0.

Bernus, Peter, Laszlo Nemes, and Theodore J. Williams. (1996). *Architectures for Enterprise Integration.* Chapman & Hall, London.

Black, J.T. (1983). Cellular manufacturing systems reduce setup time, make small lot production economical. *Industrial Engineering,* 15(11), 36–48.

Boudens, Gijs. (1997). Eenduidig opstellen van het business control model in de dynamic enterprise modeler. M.Sc. thesis, Eindhoven University of Technology, Eindhoven (in Dutch).

Chen, D.G. Doumeingts, and L. Pun. (1990). An integrated inventory model based upon GRAI tools. *Engineering Costs and Production Economics,* 19, 313–318.

Didic, M.M., F. Couffin, E. Holler, S. Lampérière, F. Neuscheler, J. Rogier, and M. de Vries. (1995). Open engineering and operational environment for CIMOSA. *Computers in Industry; Special Issue on Validation of CIMOSA,* 27(2), 167–178.

Doumeingts, Guy, Bruno Vallespir, Didier Darricau, and Michel Roboam. (1987). Design methodology for advanced manufacturing systems. *Computers in Industry,* 9(4), 271–296.

Doumeingts, Guy, David Chen, and François Marcotte. (1992). Concepts, models and methods for the design of production management systems. *Computers in Industry,* 19(1), 89–111.

Doumeingts, G., B. Vallespir, and D. Chen. (1995). Methodologies for designing CIM systems: a survey. *Computers in Industry; Special Issue on CIM in the Extended Enterprise,* 25, 263–280.

Faure, J.M., A. Bassand, F. Couffin, and S. Lampérière. (1995). Business process engineering with partial models. *Computers in Industry; Special Issue on Validation of CIMOSA,* 27(2), 111–122.

Gransier, Theo, and Werner Schönewolf. (1995). Editorial: Validation of CIMOSA. *Computers in Industry; Special Issue on Validation of CIMOSA,* 27(2), 95–100.

Huizinga, Jan. (1995). Een gereedschapkist vol software-en businesskennis. *Software Magazine,* 12(10), 68–71 (in Dutch).

IFAC/IFIP Task Force on Architectures for Enterprise Integration. (1997). *GERAM: Generalised Enterprise Reference Architecture and Methodology.* Available to download from http://www.cit.gu.edu.au/~bernus/taskforce/geram/V1_2.html.

Janusz, Barbara. (1996). Modeling and reorganizing of process chains using CIMOSA. In *Proceedings of the IFIP TC5/WG5.3/WG5.7 International Conference on the Design of Information Infrastructure Systems for Manufacturing (DIISM '96),* Jan Goossenaerts, Fumihiko Kimura, and Hans Wortmann, Eds., Chapman & Hall, London, 140–151.

Keller, Gerhard, and Sören Detering. (1996). Process-oriented modeling and analysis of business processes using the R/3 Reference Model. In *Proceedings of the IFIP TC5 Working Conference on Models and Methodologies for Enterprise Integration,* Peter Bernus and Laszlo Nemes, Eds., Chapman & Hall, London, 69–87.

Lapalus, Etienne, Shu Guei Fang, Christian Rang, and Ronald J. van Gerwen. (1995). Manufacturing integration. *Computers in Industry; Special Issue on Validation of CIMOSA,* 27(2), 155–165.

Nell, J.G. (1996). Enterprise representation: an analysis of standards issues. In *Proceedings of the IFIP TC5 Working Conference on Models and Methodologies for Enterprise Integration*, Peter Bernus and Laszlo Nemes, Eds., Chapman & Hall, London, 56–68.

Rathwell, Gary A., and Theodore J. Williams. (1996). Use of the Purdue Enterprise Reference Architecture and Methodology in industry (the Fluor Daniel example). In *Proceedings of the IFIP TC5 Working Conference on Models and Methodologies for Enterprise Integration,* Peter Bernus and Laszlo Nemes, Eds., Chapman & Hall, London, 12–44.

Reithofer, W. (1996). Bottom-up modelling with CIMOSA. In *Proceedings of the IFIP TC5/WG5.3/WG5.7 International Conference on the Design of Information Infrastructure Systems for Manufacturing (DIISM '96),* Jan Goossenaerts, Fumihiko Kimura, and Hans Wortmann, Eds., Chapman & Hall, London, 128–139.

SAP AG. (1997). R/3 *Business Engineer: Knowledge-based, Interactive R/3 Configuration and Continuous Change Management.* Available to download from http://www.sap.com/products/imple/media/pdf/50014850.pdf.

Schlotz, Claus, and Marcus Röck. (1995). Reorganization of a production department according to the CIMOSA concepts. *Computers in Industry; Special Issue on Validation of CIMOSA,* 27(2), 179–189.

Schönewolf, W., C. Rang, W. Schebesta, and M. Röck. (1992). *TRAUB/IPK Pilot/Testbed Volume—Specification Phase.* VOICE Report R92073.

Williams, T.J. (on behalf of the CIM Reference Model Committee, Purdue University). (1989). A reference model for computer integrated manufacturing from the viewpoint of industrial automation. *International Journal of Computer Integrated Manufacturing; special issue on CIM Architecture,* 2(2), 114–127.

Williams, Theodore J. (1994). The Purdue Enterprise Reference Architecture. *Computers in Industry; Special Issue on CIM Architectures,* 24(2–3), 141–158.

Williams, T.J., P. Bernus, J. Brosvic, D. Chen, G. Doumeingts, L. Nemes, J.L. Nevins, B. Vallespir, J. Vlietstra, and D. Zoetekouw. (1994a). Architectures for integrating manufacturing activities and enterprises. *Computers in Industry; Special Issue on CIM Architectures,* 24(2–3), 111–139.

Williams, Theodore J., John P. Shewchuk, and Colin L. Moodie. (1994b). The role of CIM architectures in flexible manufacturing systems. In *Computer Control of Flexible Manufacturing Systems*, Sanjay B. Joshi and Jeffrey S. Smith, Eds., Chapman & Hall, London, 1–30.

Wyns, Jo, Hendrik van Brussel, Paul Valckenaers, and Luc Bongaerts. (1996). Workstation architecture in holonic manufacturing systems. In *Proceedings of the 28th CIRP International Seminar on Manufacturing Systems,* Johannesburg, South Africa.

Zelm, Martin, François B. Vernadat, and Kurt Kosanke. (1995). The CIMOSA business modelling process. *Computers in Industry; Special Issue on Validation of CIMOSA,* 27(2), 123–142.

Zwegers, A.J.R., S.G. Fang, and H.J. Pels. (1997). Evaluation of Architecture Design with CIMOSA. *Computers in Industry,* (to be published).

Zwegers, A.J.R., and T.A.G. Gransier. (1995). Managing re-engineering with the CIMOSA architectural framework. *Computers in Industry,* 27(2), 143–153.

8

Engineering Monitoring and Diagnosis Using Wavelet Transforms

8.1 Introduction

8.2 Wavelet Transforms
The Principles of Wavelet Transforms • The Properties of Wavelet Transforms • Discrete Binary Orthogonal Wavelet Transform • Wavelet Packet

8.3 Use Wavelet Transforms for Feature Extraction
The Feature Extraction Method • The Feature Assessment Criteria • The Procedure for Automatic Feature Extraction

8.4 Use Wavelet Transform for Monitoring

8.5 Use Wavelet Transform for Diagnosis

8.6 Application Examples
Chatter Monitoring in Turning • Tool Wear Diagnosis in Drilling

8.7 Conclusions

Ruxu Du
University of Miami

In Computer Integrated Manufacturing (CIM) systems, machines and processes must be continuously monitored and controlled to ensure proper operation. Because of the complexity of the machines and processes, monitoring and diagnosis are usually difficult, involving the use of many techniques, among which signal processing is very important. Through signal processing, the critical information from the signal is captured and correlated to the health conditions of the machines and processes. There are many signal processing methods available, and their effectiveness is dependent on the characteristics of the signals.

Wavelet transform is a recently developed signal processing method. It has been successfully applied in many areas, for example; image processing, speech recognition, and data compression. This chapter introduces the basic principles of wavelet transforms and its applications for engineering monitoring and diagnosis. The chapter contains seven sections. Section 8.1 briefly introduces the history of wavelet transform along with its basic idea. Section 8.2 describes the mathematical backgrounds of wavelet transforms and several commonly used wavelet transforms, followed by a demonstration example. Section 8.3 describes the critical step of engineering monitoring and diagnosis: feature extraction. Through feature extraction, the characteristics of sensor signals are captured and related to the system conditions. Sections 8.4 and 8.5 present the methods for engineering monitoring and diagnosis, respectively. Section 8.6 presents two application examples of wavelet transform in engineering monitoring and diagnosis: one is chatter monitoring in a turning and the other is tool condition in drilling. Finally, Section 8.7 contains the conclusion.

8.1 Introduction

According to *Webster's New World Dictionary of the American Language*, monitoring, among several other meanings, means checking or regulating the performance of a machine, a process, or a system. Diagnosis, on the other hand, means deciding the nature and the cause of a diseased condition of a machine, a process, or a system by examining the symptoms. In other words, monitoring is detecting suspicious symptoms, while diagnosis is determining the cause of the symptoms. There are several other words and phrases that have similar or slightly different meanings, such as fault detection, fault prediction, in-process evaluation, online inspection, identification, and estimation.

Monitoring and diagnosis is an integrated part of Computer Integrated Manufacturing (CIM) systems. To ensure proper operation, machines and processes in CIM systems must be continuously monitored and controlled. For example, in an automated machining cell, tool condition monitoring is very important because broken tools or worn-out tools will inevitably produce scratch parts.

Due to the complexity of the machines and processes, monitoring and diagnosis are usually difficult, involving the use of many techniques from sensing to signal processing, decision-making, and cost-effective implementation. In general, and despite the differences among machines, processes, and systems, engineering monitoring and diagnosis follow a similar structure as shown in Fig. 8.1. As shown in the figure, the health condition of a system (referred to as the system condition) may be considered as the input of the system and the sensor signals as the outputs of the system, which are also affected by noise. Through signal processing, the features of the signals (called feature signals) are captured. Finally, based on the feature signals, the system conditions are estimated. Clearly, signal processing is very important, without which the critical information (the feature signals) could not be captured.

Depending on the applications, various sensor signals can be used; for example, force, pressure, vibration (displacement and acceleration), temperature, voltage, current, acoustics, acoustics emission, optic image, etc. The variation of a signal predicts changes in the health conditions of a machine or a process. For example, the excessive temperature increase in an electrical motor usually correlates to either electrical problems such as a short-circuit or mechanical problems such as scratched bearing. Unfortunately, the signals are also affected by the process working conditions and various noises. In the above example, the variations in the process working condition may include rotating speed and the load of the motor; and the noise disturbances may include power supply fluctuation and sampling noise. The effects of working conditions and noise disturbance can be minimized by means of signal processing. In general, for engineering monitoring, diagnosis, and control, the sensor signals appear as functions of time. All of the information contained in the signal is hidden in the complicated arabesques appearing in its graphical representation. The objective of signal processing is to capture the features signal that characterize the system conditions.

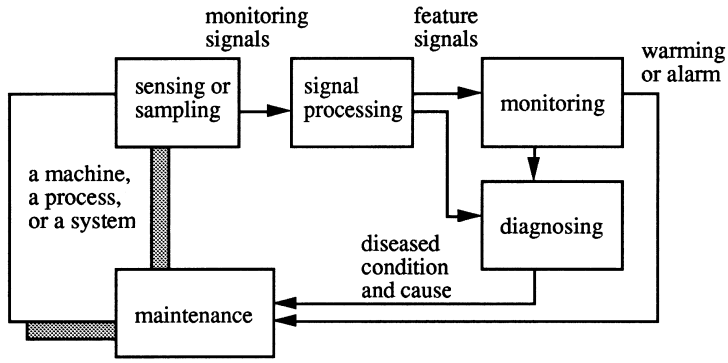


FIGURE 8.1 A unified model for engineering monitoring and diagnosis.

Depending on the characteristics of the signals, different signal processing methods should be used. One of the most commonly used signal processing methods is the Fourier transform, which effectively captures the characteristics of linear time-invariant periodic signals (called stationary signals). The characteristic features of the stationary signals include the peak frequencies, as well as the energy and the damping at each peak frequency. Because of the development of Fast Fourier Transform (FFT), Fourier analysis has become an industrial standard in many engineering fields such as the vibration analysis of rotating machinery and the regulation of electrical circuits. However, there are nonlinear and/or time-variant signals (non-stationary signals) that contain much more information than can be characterized by the peak frequencies, energy, and damping. In this case, it is necessary to use other methods, such as wavelet transform.

Wavelet transform is a relatively new method of signal processing that has recently been applied to various science and engineering fields with great success. Its development was nursed by a number of researchers such as Daubechies, Meyer and Mallat in late 1980s and early 1990s (Daubechies, 1988; 1992; Meyer, 1989; Mallat, 1989a, b). They proposed the mathematical definition of wavelet transform, discovered the basic features of wavelets, developed discrete wavelet transform algorithms, and solved several practical problems in speech recognition and computer vision. Since then, a great deal of research has been carried out and the successful applications have been reported in the literature.

According to the literature, most wavelet transform applications are associated with computer vision (image processing) and speech recognition. It has been shown that, using wavelet transform, one can extract the characteristic features from a given image and compress the image into a feature representation containing much less data. For example, DeVore, Jawerth, and Lucier (1992) studied the relationship between the original image and the image reconstructed from the wavelet feature representation and showed that the wavelet transform is near-optimal for a large class of images. Coifman and Wickerhanster (1992) proposed an entropy-based algorithm to extract features from sound waves based on wavelet packet transform. They showed an example of feature extraction for the word "armadillo," although they did not show the efficiency of the extraction and the compression ratio. Interestingly, the reconstruction of the signal from wavelet features may be unstable, as shown in Hummel's simulation (1989). Another example is the Ranganath's enhancement filter used in processing chest radiograph (Ranganath, 1991). However, the applications of wavelet transform in engineering monitoring and diagnosis have been limited. In fact, little research has been found. One of them was the work by Tansel and McLauglin (1993), who used wavelet transform and artificial neural network for tool condition monitoring in deep-hole drilling. They showed that the wavelet transform can be used to extract useful features from vibration signals but did not show the effectiveness of the selected features. In Wu and Du (1996), wavelet packet transform was used for tool condition monitoring in turning and drilling.

In general, the use of wavelet transform for engineering monitoring and diagnosis consists of two steps: feature extraction and decision-making. Feature extraction can be viewed as signal compression, which transforms the original signal (usually a large set of data) into a number of features (a much smaller set). This will inevitably result in the loss of information from the original signal. Due to the nature of the problems in computer vision and speech recognition, most studies on wavelet transform have concentrated on extracting features that represent the original signal near 100%. However, for engineering monitoring and diagnosis, the sensor signals are typically contaminated by noise. Therefore, it is desirable to extract the features that represent the characterisstics of the process (information) and to separate these features from the noise. Evaluating the effectiveness of the extracted features is referred to as feature assessment. Feature assessment has been discussed in several articles. For example, Ornstein and Weiss (1993) proposed a method relating the features to the entropy of a system. Rioul and Flandrin (1992) developed a feature assessment method based on signal energy representations as a function of time and scale. Cho and et al. (1992) proposed the use of coherence spectra, and Chen and et al. (1991) suggested the use of the autocorrelation function of residues. All these techniques have their advantages and limitations.

Decision-making can be viewed as a process that correlates the feature signals to the system conditions. It is usually done through supervised learning: operators instruct the computer of the possible patterns in the feature signals and relate them to the system conditions; then, the computer fine-tunes the patterns

to best-fit the data. Typical computer decision-making methods include pattern recognition, fuzzy logic, decision tree, and artificial neural network. Ideally, there exists a one-to-one correlation between feature signals and system conditions. When such a correlation is found, engineering monitoring and diagnosis will be most effective and efficient (Du, Elbewatwi, and Wu, 1995). In fact, this is the motivation for studying the wavelet transforms: use wavelet transform to capture the feature signals that are closely correlated with the system conditions, and hence to make intelligent monitoring and diagnosis decisions.

8.2 Wavelet Transforms

The Principles of Wavelet Transforms

It is well known that an energy-limited signal (i.e., a square integratable signal), $f(t)$, can be represented by its Fourier transform $F(\omega)$ as:

$$f(t) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} F(\omega)e^{i\omega t}\,d\omega \tag{8.1}$$

where

$$F(\omega) = \int_{-\infty}^{+\infty} f(t)e^{-i\omega t}\,dt \tag{8.2}$$

Note that $F(\omega)$ and $f(t)$ constitute a pair of Fourier transforms. Equation (8.2) is called the Fourier transform of $f(t)$ and Equation (8.1) is called the inverse Fourier transform. From a mathematical point of view, Eq. (8.1) implies that the signal $f(t)$ can be decomposed into a family of harmonics $e^{i\omega t}$ and the weighting coefficients $F(\omega)$ represent the amplitudes of the harmonics in $f(t)$.

Wavelet transform is defined in a similar manner. However, instead of using the harmonics $e^{i\omega t}$, the wavelet transform uses a series of wavelet bases:

$$\psi_{s\tau}(t) = \frac{1}{\sqrt{s}}\psi\left(\frac{t-\tau}{s}\right) \tag{8.3}$$

where $s > 0$ represents the scale, $\tau \in \mathbf{R}^n$ represents the translation, and $\psi(o)$ is called mother wavelet. Mother wavelet can be in various forms, such as Morlet's function, Mexican hat function, piecewise constant wavelet function (Rioul, 1993), as well as Lemarie and Battle's function (Mallat, 1989a). Nevertheless, the integrals of $\psi(o)$ and its moments (1st, 2nd,…,$(m-1)$th) must be zero; that is:

$$\int_{-\infty}^{+\infty} \psi(t)\,dt = \cdots = \int_{-\infty}^{+\infty} t^{m-1}\psi(t)\,dt = 0 \tag{8.4}$$

Following the wavelet bases, the signal $f(t)$ can be represented as follows (Daubechies, 1990; Mallat 1989a):

$$f(t) = \frac{1}{C_\psi}\int_{-\infty}^{+\infty}\int_{0}^{+\infty} W_s[f(\tau)]\psi_{s\tau}(t)\,ds\,d\tau \tag{8.5}$$

where, c_ψ is a normalization constant depending on the mother wavelet, and $W_s[f(\tau)]$ is the wavelet transform defined as:

$$W_s[f(\tau)] = \int_{-\infty}^{+\infty} f(t)\psi_{s\tau}(t)\,dt \tag{8.6}$$

Similar to Fourier transform, $W_s[f(\tau)]$ and $f(t)$ constitute a pair of wavelet transforms: Eq. (8.6) is called the wavelet transform and Eq. (8.5) is referred to as the inverse wavelet transform (also called reconstruction formula). Analogy to Fourier transform, Eq. (8.6) implies that the wavelet transform can be considered as signal decomposition. It decomposes the signal $f(t)$ onto a family of wavelet bases; and the weighting coefficients, $W_s[f(\tau)]$, represent the amplitudes at given time τ with scale s. This is the so-called Grossmann-Morlet wavelet.

From a physical point of view, Fourier transform is well suited for stationary signals. In Fourier transform, stationary signals are decomposed canonically into linear combinations of waves (sines and cosines). However, Fourier cannot describe the transient events in the nonstationary signals because the transient events can hardly be approximated by sines and cosines. This necessitates the use of wavelet transform. In wavelet transforms, nonstationary signals can be decomposed into linear combinations of wavelets. To show this, let us first examine how the wavelets are formed. The wavelets, $\psi_{s\tau}(t)$, are constructed from the mother wavelet, $\psi(o)$, through the process of translation and dilation. Fig. 8.2 shows a typical example of mother wavelet shaped as a Mexican hat. The translations of the mother wavelet are shown in Fig. 8.3. They form a series of time windows, $\psi_{0i}(t)$, $i = 1, 2, \ldots$, which can capture the signal features at specific times. Note that $\psi_{00}(t) = \psi(t)$. Fig. 8.4 illustrates the dilation of the mother wavelet, $\psi_{j0}(t)$, $j = 1, 2, \ldots$, and their corresponding spectra. It is seen that the dilation changes the frequency window so that specific frequency components of the signal can be captured. With a combination of translation and dilation, the wavelets form a series of time windows and frequency bands from which all kinds of feature signals can be captured. Hence, wavelet transforms effectively describe the nonstationary signals.

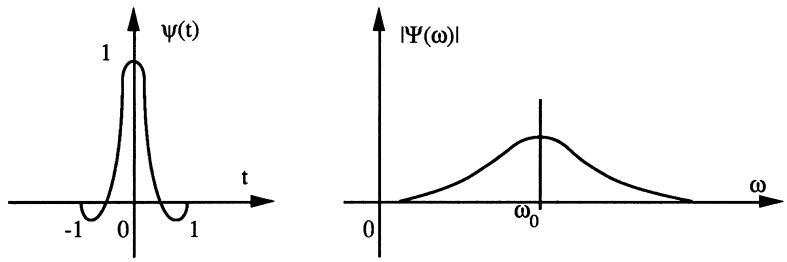


FIGURE 8.2 The mother wavelet shaped as a Mexican hat.

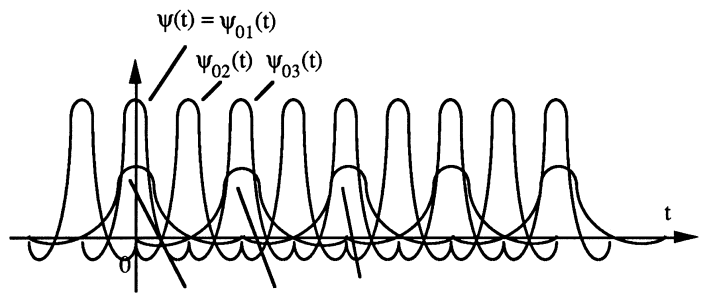


FIGURE 8.3 The translations of the mother wavelet.

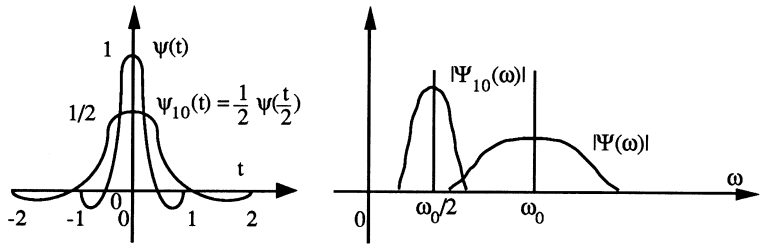


FIGURE 8.4 The dilations of the mother wavelet and their corresponding spectra.

Because of the variability in nonstationarity, the wavelet techniques, which are specific to the nonstationarity of the signal, include wavelets of the "time-frequency" and "time-scale" types. "Time-frequency" wavelets are suited, most specifically, to the analysis of quasi-stationary signals, while "time-scale" wavelets are adequate for signals having a fractal structure. In the following discussion, we focus on the time-scale type of wavelet transform as defined in Eq. (8.5) for two reasons. First, in most engineering monitoring and diagnosis applications, the sensor signals have fractal structures (i.e., the signals have transient, trend, and/or multiple and varying frequencies and amplitudes). For example, in a drilling operation, a drill gradually engages and penetrates a workpiece and then withdraws from the workpiece. Consequently, the corresponding spindle motor current varies following a pattern of going up and down. Such a nonlinear signal can be best characterized by time-scale wavelet transforms. Second, time-scale wavelet transforms can be easily and efficiently implemented. This is discussed further in a later section.

The Properties of Wavelet Transforms

The properties of wavelet transforms may vary, depending on the mother wavelet. However, it can be shown that all wavelet transforms possess four important properties (Meyer, 1989):

1. *Multi-resolution*: Wavelet transform decomposes a signal into various components at different time windows and frequency bands, all of which form a surface in the time-frequency plane. The size of the time window is controlled by the translation, while the length of the frequency band is controlled by the dilation. Hence, one can examine the signal at different time windows and frequency bands by controlling the translation and the dilation. This is called multi-resolution. It is interesting to note that multi-resolution is a unique feature of wavelet transform. Those who are familiar with time-frequency distributions may recall that time-frequency distributions can also describe the time-frequency information of the signal. However, in time-frequency distributions, both the time window and frequency band are fixed.
2. *Localization*: As shown in Fig. 8.4, the dilation changes the shapes of the wavelet bases. The smaller the dilation j, the sharper the shape. On the other hand, as shown in Fig. 8.3, the translation shifts the wavelet bases with time. By controlling the dilation and the translation, specific features of a signal at any specific time-frequency window can be explicitly obtained. This is called localization, which magnifies specific features of the signal. For example, if we want to capture a wide frequency band signal feature that occurs with a time delay, then we could use $\psi_{12}(t), \psi_{13}(t), \ldots$, or $\psi_{22}(t), \ldots$, etc. as filters. These filters can filter out the other components of the signal and hence locate the interested features of the signal. In comparison, in time-frequency distributions, the information in every time-frequency window can only be equally weighted.
3. *Zoom-in and zoom-out*: As shown in Fig. 8.4, the time window and the frequency band of the wavelet bases change correspondingly through the dilation. The wider the time window, the narrower the frequency band, and vice versa. For example, when the time window changes

from $\psi_{00}(t)$ to $\psi_{10}(t)$, the frequency band changes from $\psi_{00}(\omega) = \psi(\omega)$ to $\psi_{10}(\omega)$. This is called zoom-in and zoom-out. It implies that wavelet transforms are capable of capturing both the short-time, high-frequency information and the long-time, low-frequency information of the signal. This is desirable because, with a large time window, we can still capture specific frequency components. In comparison, in Fourier transform and time-frequency distributions, the increase of time window results in an increase of frequency band, and hence may result in the loss of accuracy.

4. *Reconstruction*: A signal $f(t)$ can be reconstructed from its wavelet transform at any resolution without information loss. This will be discussed later.

In fact, it can be shown that Fourier transform and time-frequency distributions are special cases of wavelet transforms (Daubechies, 1990). With their capability of multi-resolution, localization, zoom-in and zoom-out, and reconstruction, the wavelet transforms allow us to see both the "forest" and the "tree" and, hence, are much more effective for signal processing for nonstationary signal processing.

Discrete Binary Orthogonal Wavelet Transform

Binary orthogonal wavelet transform is one of the earliest developed and commonly used wavelet transforms. It is a discrete wavelet transform designed for digital signals (Meyer, 1989; Rioul, 1993). In discrete wavelet transforms, the scale parameter s is taken as an integer power of two, that is, $s = 2^j$, $j = 1, 2, \ldots$; and the time parameter t is taken as a series of integers k (i.e., $t \rightarrow k = 1, 2\ldots$); that is:

$$\psi_{jk}(t) = \frac{1}{2^j}\psi\left(\frac{t}{2^j} - k\right) \tag{8.7}$$

where, $j, k = 1, 2, \ldots$.This is the Daubechies wavelet.

One of the most commonly used discrete wavelet transforms is the binary orthogonal wavelet transform. Let $A_j[o]$ and $D_j[o]$ be a pair of operators. At the jth resolution, $A_j[f(t)]$ represents an approximation of the signal $f(t)$, and $D_j[f(t)]$ represents the information loss, or the so-called the detail signal (Mallat, 1989). Then, it can be shown that (Meyer, 1989):

$$A_j[f(t)] = f(t) * \phi_j(t) \tag{8.8}$$

$$D_j[f(t)] = f(t) * \psi(t) \tag{8.9}$$

where, $\phi_j(t)$ is a smooth scaling orthogonal bases, $\psi_j(t)$ is an orthogonal wavelet bases, and "$*$" denotes convolution. Furthermore, $\phi_j(t)$ and $\psi_j(t)$ are correlated through a pair of conjugate quadrature mirror filters $h(t)$ and $g(t)$ defined below:

$$\phi_j(t) = h(t) * \phi_{j-1}(t) \tag{8.10}$$

$$\psi_j(t) = g(t) * \psi_{j-1}(t) \tag{8.11}$$

Following the reference (Meyer, 1989), we can use the filters defined below:

$$h(t) = [0.48296, 0.83652, 0.22414, -0.12941]$$

$$g(t) = [0.12941, -0.22414, -0.83652, -0.48296]$$

It should be pointed out that using different filters will result in different wavelet results. It is often necessary to combine Eq. (8.8) to (8.11); thus, we have the binary wavelet transform:

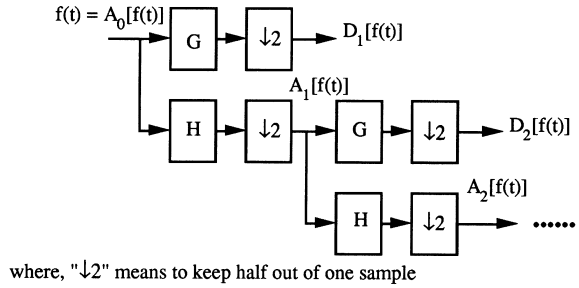


FIGURE 8.5 The pyramid algorithm of the binary wavelet transform.

$$A_j[f(t)] = h(t) * A_{j-1}[f(t)] \tag{8.12}$$

$$D_j[f(t)] = g(t) * A_{j-1}[f(t)] \tag{8.13}$$

or,

$$\begin{aligned} A_0[f(t)] &= f(t) \\ A_j[f(t)] &= \sum_k h(k-2t) * A_{j-1}[f(t)] \\ D_j[f(t)] &= \sum_k g(k-2t) * A_{j-1}[f(t)] \end{aligned} \tag{8.14}$$

where, $t = 1, 2, \ldots, N$; $j = 1, 2, \ldots, J$; and $J = \log_2 N$.

The binary wavelet transform has the following features:

1. At the jth resolution, a signal $f(t)$ is decomposed into two parts: an approximation $A_j[f(t)]$ and a detail signal $D_j[f(t)]$. Both can be calculated by the convolution of the previous approximation $A_{j-1}[f(t)]$. Therefore, the discrete binary orthogonal wavelet transform can be computed recursively through a "pyramid algorithm" as illustrated in Fig. 8.5.
2. From a mathematical point of view, the filters $g(t)$ and $h(t)$ are a pair of quadrature mirror filters: $h(t)$ is a low-pass filter and $g(t)$ is a high-pass filter. At each recursion step in Eq. (8.14) the approximation $A_j[f(t)]$ is obtained from the previous approximation passing through the low-pass filter, while the detail signal $D_j[f(t)]$ is obtained from the previous approximation passing through the high-pass filter. Hence, $A_j[f(t)]$ represents the low-frequency components of original signal $f(t)$, and $D_j[f(t)]$ represents the high-frequency components of the original signal $f(t)$. Moreover, with the increase of resolution j, the center frequencies of the filters decrease by a ratio of $(1/2)^j$.
3. Because of the orthogonality of $\phi(t)$ and $\psi(t)$, the lengths of $A_j[f(t)]$ and $D_j[f(t)]$ are half of the length of $A_{j-1}[f(t)]$. That is, at each recursion, the signal is compressed to half its size in its approximation. And because the signal $f(t)$ has a finite number of samples N, the binary wavelet transform is limited in resolution. The maximum resolution on which the wavelet transform can be performed is:

$$J = \log_2 N \tag{8.15}$$

At each resolution, however, the wavelet representation, $\{A_J[f(t)], D_j[f(t)]|_{j=1,2,\ldots,J}\}$ always has the same number of data as the signal $f(t)$. Hence, the wavelet representation contains the same amount of information as the original signal $f(t)$, although half the information is compressed in the approximation, while the other half is in the detail signal.

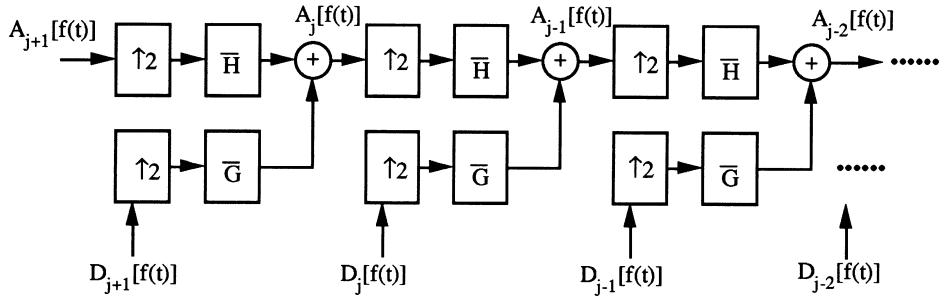


FIGURE 8.6 The reconstruction algorithm of the binary wavelet transform.

4. Because the wavelet transform is a complete representation of the signal, the original signal $f(t)$ can be reconstructed by means of an inverse wavelet transform or the reconstruction formula below:

$$A_j[f(t)] = 2\left\{\sum_k h(k-2t) * A_{j+1}[f(t)] + \sum_k g(k-2t) * D_{j+1}[f(t)]\right\} \tag{8.16}$$

where, $j = J-1, J-2, \ldots, 1, 0$. This reconstruction algorithm is illustrated in Fig. 8.6.

Wavelet Packet

Let operators H and G be the convolution sum defined below:

$$H\{0\} = \sum_k h(k-2t) \tag{8.17}$$

$$G\{0\} = \sum_k g(k-2t) \tag{8.18}$$

Then, Eq. (8.14) can be rewritten as follows:

$$A_j[f(t)] = H\{A_{j-1}[f(t)]\} \tag{8.19}$$

$$D_j[f(t)] = G\{A_{j-1}[f(t)]\} \tag{8.20}$$

It is seen that the binary wavelet transforms use the operators H and G for the approximation $A_{j-1}[f(t)]$ only, but not the detail signal $D_{j-1}[f(t)]$. If the operators H and G are applied to both $A_{j-1}[f(t)]$ and $D_{j-1}[f(t)]$, then it results in the wavelet packet transform [Coifman and Wickerhanster, 1992]:

$$A_j[f(t)] = H\{A_{j-1}[f(t)]\} + G\{D_{j-1}[f(t)]\} \tag{8.21}$$

$$D_j[f(t)] = G\{A_{j-1}[f(t)]\} + H\{D_{j-1}[f(t)]\} \tag{8.22}$$

Let $P_j^i(t)$ be the ith packet on the jth resolution; then, the wavelet packet transform can be computed by the recursive algorithm below:

$$\begin{aligned} P_0^1(t) &= f(t) \\ P_j^{2i-1}(t) &= H[P_{j-1}^i(t)] \\ P_j^{2i}(t) &= G[P_{j-1}^i(t)] \end{aligned} \tag{8.23}$$

Resolution	Original Signal f(t), t = 1, 2, …, N								Number of Packets	Data in Each Packet
1st	$P_1^1(t) = Hf(t)$				$P_1^2(t) = Gf(t)$				2^1	$2^{J-1} = N/2$
2nd	$P_2^1 = HP_1^1$		$P_2^2 = GP_1^1$		$P_2^3 = HP_1^2$		$P_2^4 = GP_1^2$		2^2	$2^{J-2} = N/4$
3rd	…	…	…	…	…	…	…	…	2^3	$2^{J-3} = N/8$
…	……								…	…
jth	$P_i^j(t)$, $i = 1, 2, \ldots, 2j$; $t = 1, 2, \ldots, 2^{J-j}$								2^j	2^{J-j}

FIGURE 8.7 The pyramid algorithm of the wavelet packet transform.

where $t = 1, 2, \ldots, 2^{J-j}$, $i = 1, 2, \ldots, 2^j$, $j = 1, 2, \ldots, J$, and $J = \log_2 N$. The wavelet packet transform algorithm can be represented by a binary tree as shown in Fig. 8.7. From the figure, it is seen that the signal $f(t)$ can be decomposed into a small number of large packets at lower resolutions ($j = 1$ or 2) or a large number of small packets at higher resolutions ($j = 5$ or 6). On the jth resolution, there are a total of 2^j packets and each packet contains 2^{J-j} data points.

As a modified version of binary wavelet transform, the wavelet packet transform has the same properties as the former. Also, there exists the inverse wavelet packet transform, or the signal reconstruction:

$$P_j^i(t) = 2[\bar{H}P_{j+1}^{2i-1}(t) + \bar{G}P_{j+1}^{2i}(t)] \tag{8.24}$$

where $j = J - 1, J - 2, \ldots, 1, 0$; $i = 2^j, 2^{j-1}, \ldots, 2, 1$; and the operators $\bar{H}$ and $\bar{G}$ are the conjugate of H and G:

$$\bar{H} = \sum_k h(t - 2k) \tag{8.25}$$

$$\bar{G} = \sum_k g(t - 2k) \tag{8.26}$$

This is the same as for the binary orthogonal wavelet transform. It is interesting to note that the signal reconstruction can be done either by using the packets with the same size on the same resolution, or by using the packets with different sizes on different resolutions.

It is interesting to note that the wavelet packets can be computed individually. For example, if only two packets $P_5^1(t)$ and $P_1^{11}(t)$ need to be computed, then instead of computing the entire wavelet packet transform resolution by resolution, one can use the following equations:

$$P_5^1(t) = H^5[f(t)] \tag{8.27}$$

$$P_5^{11}(t) = (HG)^2 H[f(t)] \tag{8.28}$$

This reduces the computing load to 28.75% compared to computing the entire wavelet packet transform.

In addition to the nice features inherited from the wavelet transforms, the wavelet packet transform has a couple of additional advantages, including allowing less signal details and easy digital implementation. In the discussions that follow, we will use wavelet packet transform. As an example, Fig. 8.8 shows the wavelet packet transform of a nonstationary signal. The original signal $f(t)$ is shown in Fig. 8.8a. In Fig. 8.8b, the signal is decomposed onto various number of packets at different resolutions. The reconstructed signal using all the packets at resolution 5 is shown in Fig. 8.8c. It is seen that Figs. 8.8a and 8.8c are essentially the same.

To conclude this section, it is worthwhile to point out that there are several wavelet software packages available on the market, such as the wavelet toolbox in MATLAB. The reader may find them useful and rather easy to use. Dozens of wavelet Web sites around the world can be found on the Internet, including www.amara.com. These Web sites provide updated information on wavelet research and applications, as well as free software.

8.3 Use Wavelet Transforms for Feature Extraction

The Feature Extraction Method

For engineering monitoring and diagnosis, features extraction is very important. This is due to the fact that many sensor signals are nonstationary signals. To ensure the accuracy and reliability of the monitoring, it is very important to extract the characteristic features of the signals that describe the defects of the system but are not affected by the system working conditions and noise (the system noise and the sampling noise). Using the wavelet packet transform, the signal features can be effectively extracted. From a mathematical point of view, the feature extraction can be considered as signal compression. Recalling the fact that wavelet packets represent the compressed signals, it is desirable to use the wavelet packets as the extracted features. As pointed out earlier, each wavelet packet represents a certain part of the signal in a specific time-frequency window. Apparently, not all the wavelet packets contain useful information, especially at higher resolutions. Continuing the example in Fig. 8.8, it is seen that at higher resolution, only a few packets contain large amounts of information. For example, at resolution 6, out of 64 packets, only 6 packets, $P_6^1(t)$, $P_6^2(t)$, $P_6^{43}(t)$, $P_6^{44}(t)$, $P_6^{47}(t)$, and $P_6^{59}(t)$, contain relatively large amounts of information, while the other packets are essentially zeros. Therefore, we can use the wavelet packets that contain large amounts of information as the features, which are called feature packets. From a mathematical point of view, the feature packets contain the principle components of the signal. Hence, using wavelet feature packets, we are able to extract the principle components of the signal. The feature extraction process is called compression because the selected wavelet packets contain less data than the original signal. Specifically, at the resolution j, there are 2^j packets; choosing M feature packets, $P_j^i(t)$, $i = i_1, i_2, \ldots, i_M$, will preserve $(M/2^j)$ portions of the data. Hence, the ratio $(M/2^j)$ is called the compression ratio.

Using the M feature packets $P_j^i(t)$, $i = i_1, i_2, \ldots, i_M$, and setting the other packets in the same resolution to zero, we then obtain a reconstructed signal.

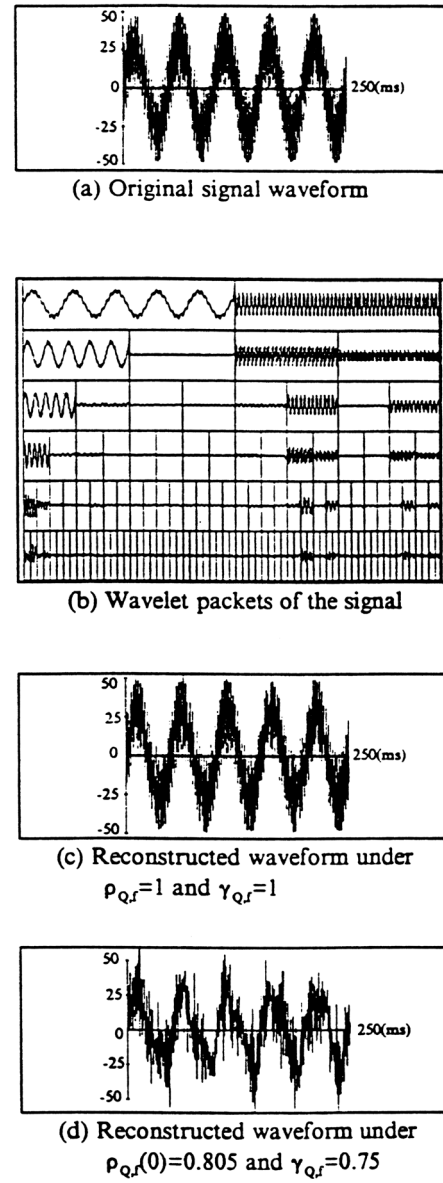


FIGURE 8.8 An example of wavelet packet transform.

Continuing the example above, using the feature packets and setting the other packets to zero, the reconstructed signal is shown in Fig. 8.8d. It is seen that the principle components of the signal are indeed preserved.

Finally, to compute the feature packets, one does not need to compute the entire wavelet packet transform. This is due to the fact that if a packet at lower resolution does not contain useful information (principle components of the signal), then its succeeding decomposition in the higher resolutions will not contain useful information and, hence, can be discarded. For the above example, the computations include only the following:

At the 1st resolution: $P_1^1(t),\ P_1^2(t)$

At the 2nd resolution: $P_2^1(t),\ P_2^3(t),\ P_2^4(t)$

At the 3rd resolution: $P_3^1(t),\ P_3^6(t),\ P_3^8(t)$

At the 4th resolution: $P_4^1(t),\ P_4^{11}(t),\ P_4^{12}(t),\ P_4^{15}(t)$

At the 5th resolution: $P_5^1(t),\ P_5^{22}(t),\ P_5^{24}(t),\ P_5^{30}(t)$

At the 6th resolution: $P_6^1(t),\ P_6^2(t),\ P_6^{43}(t),\ P_6^{44}(t),\ P_6^{47}(t),\ P_6^{59}(t)$

This can be recognized from Fig. 8.8b. It should also be pointed out that each step of the wavelet transform computation involves only simple addition and multiplication (to carry out the convolution). Therefore, the feature packets can be computed fast and hence be used for online applications.

The Feature Assessment Criteria

The effectiveness of the feature packets can be evaluated based on the reconstructed signal. Let $Q(t) = \hat{P}_0^1(t)$, the estimate of $P_0^1(t)$, be a reconstructed signal using M feature packets $P_j^i(t)$, $i = i_1, i_2, \ldots, i_M$, while the other packets in the same resolution are set to zero. Then, the effectiveness of the selected feature packets can be assessed by examining the difference between $Q(t)$ and $f(t)$. Intuitively, if the selected features are appropriate, then $Q(t)$ will be close to $f(t)$. Ideally, the difference should be a white noise, which implies that the selected features completely capture the information of the signal. Quantitatively, the feature packets can be assessed by the two criteria described below.

1. The cross-correlation between $Q(t)$ and $f(t)$. The cross-correlation between $Q(t)$ and $f(t)$ is defined by:

$$P_{Qf}(k) = \frac{R_{Qf}(k)}{R_f(k)}, \quad k = 0, 1, \ldots, N-1 \tag{8.29}$$

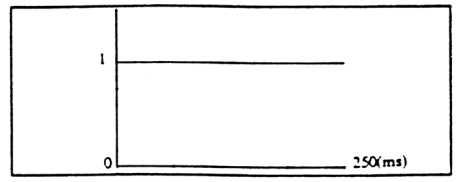
where $R_{Qf}(k) = E[Q(t)f(t-k)]$ is the cross-covariance between $Q(t)$ and $f(t)$, and $R_f(k) = E[f(t)f(t-k)]$ is the variance of $f(t)$. This definition is slightly different to the definition of cross-correlation in statistics where both the variances of $f(t)$ and $Q(t)$ would be used in the denominator. It is designed to stress how close the original signal $f(t)$ and its reconstruction $Q(t)$ are in time domain. If $Q(t)$ is equal to $f(t)$, then $\rho_{Qf}(k) = 1$, for $k = 0, 1, \ldots, N-1$. On the other hand, if there is no correlation between $Q(t)$ and $f(t)$, then $\rho_{Qf}(k) = 0$. However, $\rho_{Qf}(k)$ may be greater than 1 or smaller than -1, since the denominator $R_f(k)$ may not always be larger than $R_{Qf}(k)$. Assuming that the selected feature packets are $\{P_j^i(t), i = i_1, i_2, \ldots, i_M\}$, then they are effective if $\rho_{Qf}(k)$ approach 1, for the first few ks, ($k = 0, 1, \ldots, K$), since it implies the principle components of $Q(t)$ and $f(t)$ are the same. Note that $\rho_{Qf}(k) = 1$ is also undesirable because this would mean the original signal was being completed reproduced by the extracted features, which includes not only the useful information but also the noises. As an example,

Fig. 8.9a shows the cross-correlation corresponding to Fig. 8.8c and Fig. 8.9b shows the cross-correlation corresponding to Fig. 8.9d.

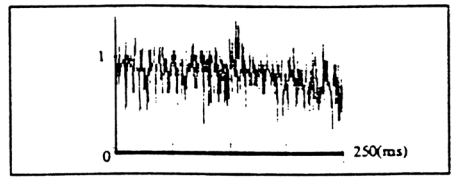
2. The cross-coherence between $Q(t)$ and $f(t)$. The cross-coherence between $Q(t)$ and $f(t)$ is defined as follows:

$$\gamma_{Qf}(\omega) = \frac{S_{Qf}(\omega)}{S_f(\omega)} \qquad (8.30)$$

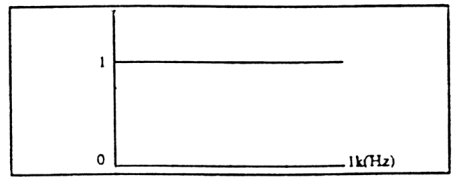
where $S_{Qf}(\omega)$ is the cross-power-spectrum between $Q(t)$ and $f(t)$, and $S_f(\omega)$ is the power spectrum of $f(t)$. Similar to the cross-correlation function defined above, the cross-coherence defined in Eq. (8.30) is slightly different to that in statistics. It stresses the difference between $f(t)$ and its reconstruction $Q(t)$. If the reconstruction $Q(t)$ has the same characteristics in frequency domain as that of $f(t)$, then $\gamma_{Qf}(\omega) = 1$ for all ω. On the other hand, if there are no common frequency components between $Q(t)$ and $f(t)$, then $r_{Qf}(W) = 0$. Note that $\gamma_{Qf}(\omega)$ may be greater than one. Similar to that of the cross-correlation, the cross-coherence measures how the selected features resemble the original signal in frequency domain. If the cross-coherence is very small (say 0.3), then the selected feature packets do not contain enough frequency information. Note that $\gamma_{Qf}(\omega) = 1$ is undesirable as well because it means that the selected features completely reconstruct the original signal, including noises. The best selected features contain the information of the signal but not the noise. Consequently, the cross-coherence will be close to one but not equal to one. As an example, Fig. 8.9c shows the cross-coherence function corresponding to Fig. 8.8c, and Fig. 8.9d shows the cross-coherence function corresponding to Fig. 8.9d.



(a) Cross-correlation of signals in Figs. 8a and 8c.



(a) Cross-correlation of signals in Figs. 8a and 8d.



(a) Cross-coherence of signals in Figs. 8a and 8c.

(a) Cross-coherence of signals in Figs. 8a and 8d.

FIGURE 8.9 The cross-correlation and the cross-coherence functions for the example in Fig. 8.8.

The use of a particular feature assessment criterion is dependent on the application. If the time-domain information in the application is the focus of the investigation, then $\rho_{Qf}(k)$ should be used. On the other hand, if the frequency-domain information is more important, then $\gamma_{Qf}(\omega)$ should be used.

The Procedure for Automatic Feature Extraction

In summary, as shown in Fig. 8.10, the procedure for automatic feature extraction consists of four steps:

Step 1: Calculate the wavelet packet transform of the training samples. Suppose that K set of sensor signals, $\{f_k(t), t = 1, 2, \ldots, N; k = 1, 2, \ldots, K\}$, is obtained from a system condition. The first step is to apply wavelet packet transform to the signal $f_k(t)$, which generates wavelet packets: $\{P^i_{kj}(t), t = 1, 2, \ldots, 2^{J-j}; k = 1, 2, \ldots, K, i = 1, 2, \ldots, 2^j\}$ at the jth resolution.

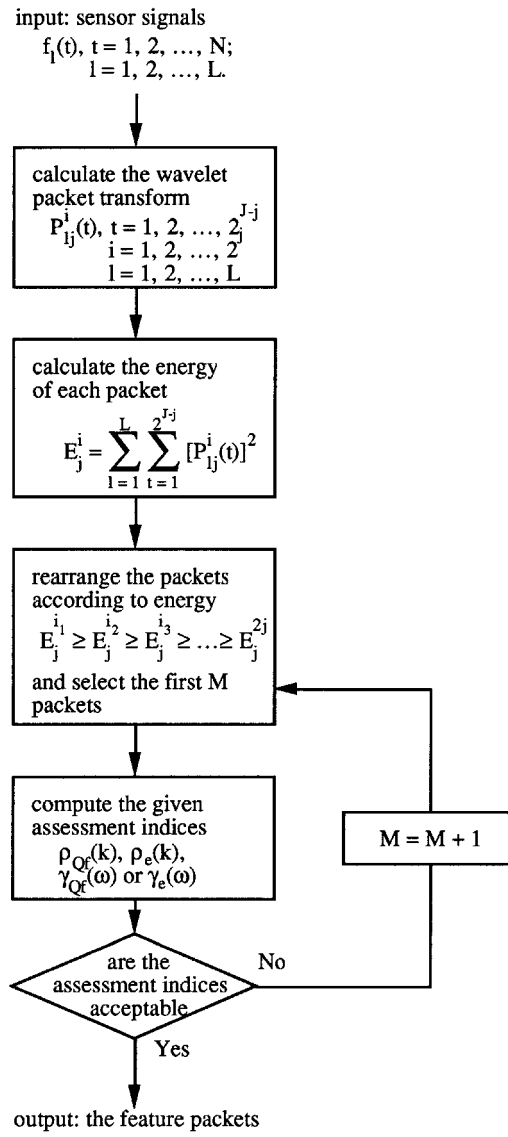


FIGURE 8.10 The procedure for feature extraction.

Step 2: Calculate the energy of the wavelet packets. The energy of a wavelet packet is defined by [15]:

$$E_j^i = \sum_{k=1}^{K} \sum_{t=1}^{2^{j-j}} [P_k^i(t)]^2 \tag{8.31}$$

It describes the amount of information that the wavelet packet contains.

Step 3: Select the feature wavelet packets according to energy. Rearrange the packets according to the descendent order of energy:

$$E_j^{i1} \geq E_j^{i2} \geq E_j^{i3} \geq \ldots \geq E_j^{2^j}$$

Then select the first M packets as the feature packets, which are denoted as $\{P_j^i(t), i = i_1, i_2, \ldots, i_M\}$. As a rule of thumb, one can start with $M = 2$.

Step 4: Evaluate the selected feature packets. Using the feature packets and setting the other packets to zero, a reconstruction $Q(t)$ can be obtained by using Eq. (8.24). Next, apply the feature assessment criteria above; if the given assessment criteria are satisfied, then the feature extraction is completed. Otherwise, add the next feature packet into the feature packets (i.e., $M = M + 1$) and repeat the above steps until a satisfactory result is obtained.

It should be noted that each feature packet contains a set of data. Hence, it is often useful to use certain indices to describe the characteristic of the data. These indices include the peak-to-valley value:

$$T_j^i = \max_t\{P_j^i(t)\} - \min_t\{P_j^i(t)\} \tag{8.32}$$

and the crest factor:

$$C_j^i = \frac{T_j^i}{\bar{P}_j^i} \tag{8.33}$$

where

$$\bar{P}_j^i = \frac{1}{2^{J-j}}\sum_{t=1}^{2^{J-j}} P_J^i(t)$$

The other indices, such as mean, variance, kurtosis, can also be used.

8.4 Use Wavelet Transform for Monitoring

As pointed out in Section 8.1, monitoring requires accuracy (sensitive to the system malfunctions under various system working conditions and disturbances) and speed (fast response to the system malfunctions). These can be achieved using wavelet feature packets.

In general, engineering monitoring consists of two phases: learning and real-time monitoring. The goal of learning is to set up the alarm thresholds for real-time monitoring. Assuming that K sets of training signals, $\{f_k(t); t = 1, 2, \ldots, N; k = 1, 2, \ldots, K\}$, are obtained under *normal* system condition (note that in monitoring, we only care if the system is normal or abnormal), then the learning can be conducted as follows:

1. Determine the wavelet feature packets. This can be done using the procedure given in Section 8.3. It should be pointed out that the energy of the wavelet packets is a very important index for monitoring, but can only be obtained from a complete signal. Hence, it cannot be directly used for real-time monitoring although it can be used for learning.
2. Reconstruct the signals using the feature packets. Using the feature packets, $\{P_j^i(t), i = i_1, i_2, \ldots, i_M\}$, and setting the other packets to zero, the reconstructed signal can be obtained by the reconstruction equation formula in Eq. (8.24).
3. Calculate the statistics of the reconstructed signals. Assuming the reconstructed signals are $\{Q_k(t); t = 1, 2, \ldots, N; k = 1, 2, \ldots, K\}$, then their mean series and variance series can be determined as follows:

$$\bar{Q}(t) = \frac{1}{K}\sum_k^K Q_k(t) \tag{8.34}$$

$$\sigma_Q^2(t) = \frac{1}{K-1}\sum_{k=1}^K (Q_k(t) - \bar{Q}(t))^2 \tag{8.35}$$

To save memory, the mean series and the variance series can be calculated recursively. The formula to calculate the mean series is:

$$\bar{Q}(t) \Leftarrow 0 \text{ (initial value)} \tag{8.36}$$

$$\bar{Q}(t) \Leftarrow \bar{Q}(t) + \frac{1}{K} Q_k(t), \quad k=1, 2, \ldots, K \tag{8.37}$$

The variance series involves nonlinear calculation. First, rearrange Eq. (8.35) as follows:

$$\begin{aligned} \sigma_Q^2(t) &= \frac{1}{K-1} \sum_{k=1}^{K} Q_k^2(t) - 2Q_k(t)\bar{Q}(t) + \bar{Q}^2(t) \\ &= \frac{1}{K-1} \sum_{k-1}^{K} Q_k^2(t) + \frac{K}{K-1} \bar{Q}^2(t) - \frac{2}{K-2} \bar{Q}(t) \sum_{k=1}^{K} Q_k^2(t) \end{aligned} \tag{8.38}$$

Using Eq. (8.35), it follows that:

$$\sigma_Q^2(t) = \frac{1}{K-1} \sum_{k=1}^{K} Q_k^2(t) - \frac{K}{K-1} \bar{Q}^2(t) \tag{8.39}$$

Therefore, the recursive formula is:

$$\sigma_Q^2(t) \Leftarrow -\frac{K}{K-2} \bar{Q}^2(t) \text{ (initial value)} \tag{8.40}$$

$$\sigma_Q^2(t) \Leftarrow \sigma_Q^2(t) + \frac{1}{K-1} Q_k^2(t), \quad k = 1, 2, \ldots, K \tag{8.41}$$

4. Calculate the alarm thresholds. Based on the mean series and the variance series, the alarm threshold(s) can be determined as follows:

$$T_U(t) = \bar{Q}(t) + \alpha \sigma_Q(t) \tag{8.42}$$

$$T_L(t) = \bar{Q}(t) - \alpha \sigma_Q(t) \tag{8.43}$$

where α is a constant associated with the confidence level, and $T_U(t)$ and $T_L(t)$ are upper and lower bounds of the thresholds, respectively. The alarm thresholds form a threshold band. At any given time t, suppose the probability distribution of $Q(t)$ is Gaussian; then with $\alpha = 3$, there will be a probability of 99.98% that the signal is retained within the threshold band. When $\alpha = 2$ is undertaken, there will be a probability of 95.54% that the signal is within the threshold band.

From the learning phase, the monitoring thresholds are determined. As an example, Fig. 8.11 shows the alarm thresholds corresponding to the example in Fig. 8.8. It is seen that the alarm thresholds have the same shape as the principle components of the signal and completely enclose the original signal. Based on the alarm thresholds, the real-time monitoring can then be conducted by checking the threshold crossing:

$$\text{if } f(t) > T_U(t) \text{ then alarm;} \tag{8.44}$$

$$\text{if } f(t) < T_L(t) \text{ then alarm;} \tag{8.45}$$

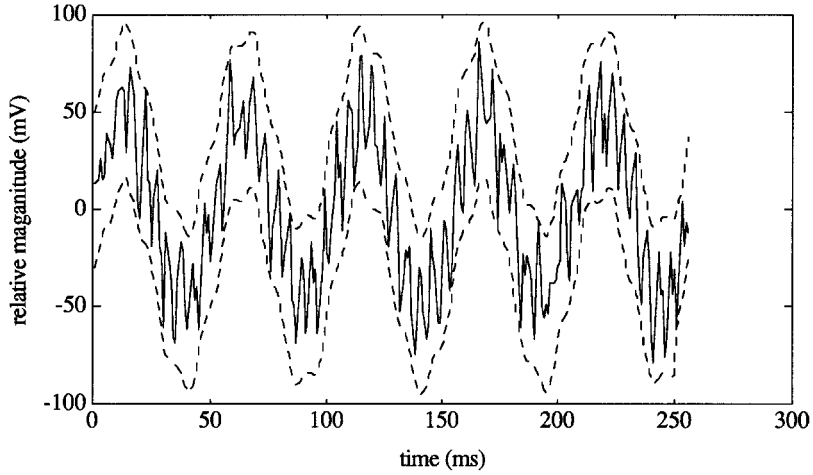


FIGURE 8.11 The alarm thresholds for the example in Fig. 8.8.

In practice, one can use different values of α to set multiple alarm levels. For example, use $\alpha = 2$ to warning and use $\alpha = 3$ to alarm. Furthermore, it should be noted that because the thresholds are established based on the principle components of the training samples, it tends to be smaller, especially when the training samples are limited. To minimize a false alarm, upon the detection of warning, one can choose higher values such as $\alpha = 4$.

Note that the monitoring is done by checking the monitoring signal point-by-point. It requires only simple comparison, and thus can be done very fast and can be used for real-time monitoring. Moreover, to further improve the accuracy of monitoring, when a warning is detected, one can calculate the wavelet packet transform of the signal and compare the feature packets to that of the training samples. If a significant difference is found, than an alarm should be issued. For the example above, such a monitoring index is:

$$S = \sum_t P_5^1(t) + P_5^{11}(t) \tag{8.46}$$

One of the distinct features of this method is that it uses only the training samples from normal system conditions. The system malfunctions, which may be in many different forms, are considered as the complementary of the normal condition. To determine what the system malfunction is, one can use the diagnosis method presented in the following section.

8.5 Use Wavelet Transform for Diagnosis

Using the wavelet transform for engineering diagnosis is also based on wavelet feature packets. Suppose the feature packets have been found, and the diagnosis is to determine the correlation between system conditions and feature packets. Ideally, there would exist a one-to-one correlation between a system condition and one or several feature packets. In this case, there is little need for further computation. However, when there is no simple one-to-one correlation (e.g., one system condition is related to several wavelet packets), it will be necessary to use classification schemes such as pattern recognition, fuzzy classifications, decision trees, and neural networks.

Diagnosis also consists of two phases: learning and decision-making. Learning is to establish a correlation between the feature signals and the system conditions from available learning samples. Decisionmaking is to determine the system condition of a new sample based on the correlation established in learning. This is illustrated in Fig. 8.12. From the figure, it is seen the correlation between feature signals

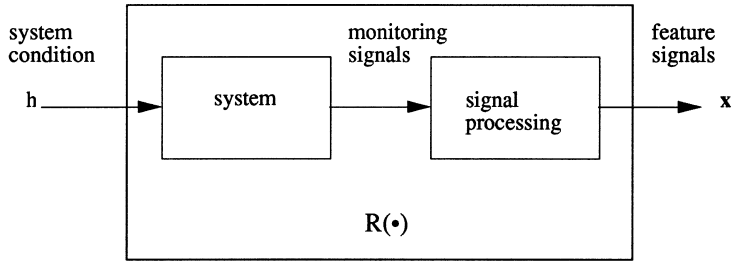


FIGURE 8.12 An illustration of the correlation between feature signals and system conditions.

($\mathbf{x} = [x_1, x_2, \ldots, x_m]$) and corresponding system conditions ($h_j \in \{h_1, h_2, \ldots h_n\}$) can be described by:

$$\mathbf{x} = R(h_j) \tag{8.47}$$

where R is the correlation function. Accordingly, learning is to find R from available learning samples $\mathbf{x}_k$, $k = 1, 2, \ldots, N$, (note that their corresponding system conditions are known). On the other hand, decision-making is to find the corresponding system condition h of a new sample ($\mathbf{x}$) from R, that is:

$$h = R^{-1}(\mathbf{x}) \tag{8.48}$$

Different methods will develop different types of correlation. For example, when pattern recognition is used, the correlation function will be patterns. On the other hand, when an artificial neural network is used, the correlation function will be a neural network. Depending on the correlation, different learning and decision-making methods would be used. In this section, we present a simple and effective method: the pattern recognition method. The other methods can be found in the literature, such as Du, Elbestawi, and Wu (1995).

Pattern recognition methods have been widely used in engineering monitoring and diagnosis (Monostori, 1986; Houshmand and Kannatey Asibu, 1989; Elbestawi et al., 1989). In general, pattern recognition methods can be divided into two groups: statistical pattern recognition methods (also called non-deterministic pattern classification methods) and distribution-free methods (also called deterministic pattern classification methods) (Kandel, 1982). Again, we will limit our discussions to distribution-free methods.

The distribution-free pattern recognition methods are based on the similarity between a sample $\mathbf{x}$ and the patterns that describe the system conditions. From a geometry point of view, the feature signals (when we use the wavelet transforms, they are the feature packets, or the indices of the feature packets) span a m-dimensional space. In the space, each system condition h_j is characterized by a pattern vector $\mathbf{p}_j = [p_{j1}, p_{j2}, \ldots, p_{jm}]^T$. On the other hand, a sample $\mathbf{x}$ is also a vector in the space. The similarity between the sample and a pattern can be measured by the distance between the two vectors. The minimum distance can then be used as the criterion for classifying the sample. Fig. 8.13 demonstrates a simple example, where the three feature are X_1, X_2, and X_2; and the three system conditions are described by patterns $\mathbf{p}_1$, $\mathbf{p}_2$, and $\mathbf{p}_3$. Given a sample $\mathbf{x}$, the distances between the sample and the patterns are d_1, d_2, and d_3, respectively. Since d_1 is the smallest, it is estimated that the sample corresponds to the system condition h_1.

There are a number ways to define patterns and distances. Hence, there are various distribution-free pattern recognition methods. Some of the most frequently used methods include the Mahalanobis method, the linear discriminate method, and the Fisher method. In the Mahalanobis method, the patterns are the means of the learning samples (i. e., $\mathbf{p}_j = \boldsymbol{\mu}_j$, where $\boldsymbol{\mu}_j$ is the mean of the samples whose system condition is h_j), and the distance is defined as:

$$q_j(\mathbf{x}) = (\mathbf{x} - \mathbf{p}_j)^T \mathbf{S}_j (\mathbf{x} - \mathbf{p}_j) \tag{8.49}$$

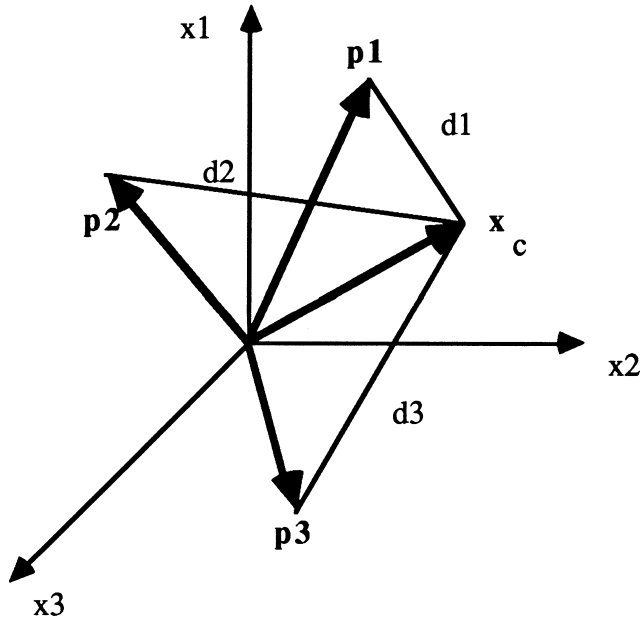


FIGURE 8.13 Illustration of distribution-free pattern recognition methods.

where $\mathbf{S}_j$ is the standard deviation matrix. The linear discriminate method, also called the *K*-mean method, uses the same pattern, but the distance is defined as:

$$q_j(\mathbf{x}) = \sum_{i=1}^{m} w_{ij}(x_i - c_{ij})^2 \tag{8.50}$$

where $\mathbf{w}_j = \{w_{1j}, w_{2j}, \ldots, w_{mj}\}$ and $\mathbf{c}_j = \{c_{1j}, c_{2j}, \ldots, c_{mj}\}$ are the weight and center of pattern *j*, respectively. In the learning phase, using an iteration procedure, the weights and the centers are determined by minimizing:

$$J = \sum_{k=1}^{N}\sum_{j=1}^{n} \delta_{jk} q_j(\mathbf{x}_k) \tag{8.51}$$

where δ_{ij} is a delta function defined below:

$$\delta_{ij} = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{if } i \neq j \end{cases}$$

Similarly, the Fisher method uses the same patterns, but the distance is defined as:

$$q_j(\mathbf{x}) = \mathbf{b}_j^T \mathbf{x} \tag{8.52}$$

where $\mathbf{b}_j$ is determined by maximizing:

$$J = \sum_{j=1}^{n} \mathbf{b}_j^T \mathbf{S}_j \mathbf{b}_j \tag{8.53}$$

In the classification phase, the system condition of a new sample is estimated as follows:

$$h_j^* = \arg\min_j \{h_j\} \tag{8.54}$$

The effectiveness of a pattern recognition method depends not only on the classification function, but also on a number of factors such as the distribution of the learning samples and the definition of system conditions. For example, tool wear in machining processes can be manifested in numerous forms, and each can result in different consequences. Therefore, the monitoring and diagnosis decision is often not based on the system condition, but rather to what degree that the system is in that condition. In this case, advanced methods such as fuzzy logic and artificial neural network may be more effective. For details of these methods, the reader is referred to the related chapter of this series.

8.6 Application Examples

To demonstrate the use of wavelet transforms for engineering monitoring and diagnosis, two examples are presented below: chatter monitoring in turning and tool wear diagnosis in drilling.

Chatter Monitoring in Turning

Chatter is defined as excessive vibration during machining operations. It may be caused by either resonance vibration or nonlinear cutting force and vibration interaction. The experiments for chatter monitoring in turning were conducted on an engine lathe as shown in Fig. 8.14 with the cutting conditions listed in Table 8.1.

The monitoring signal was the vibration (acceleration) signal measured from the tailstock of the lathe. During the turning of a long slim steel bar, chatter gradually developed. At the beginning of the cut, the cutting was stable. When the cutter approached the middle of the workpiece, where the structure is weaker, chatter began to develop, at which time the vibration amplitude escalated. This interim period lasted about 0.5 second. Then, chatter was fully developed, at which the vibration amplitude reached a limit. The purpose of chatter monitoring was to detect the onset of the chatter, (i.e., the interim period).

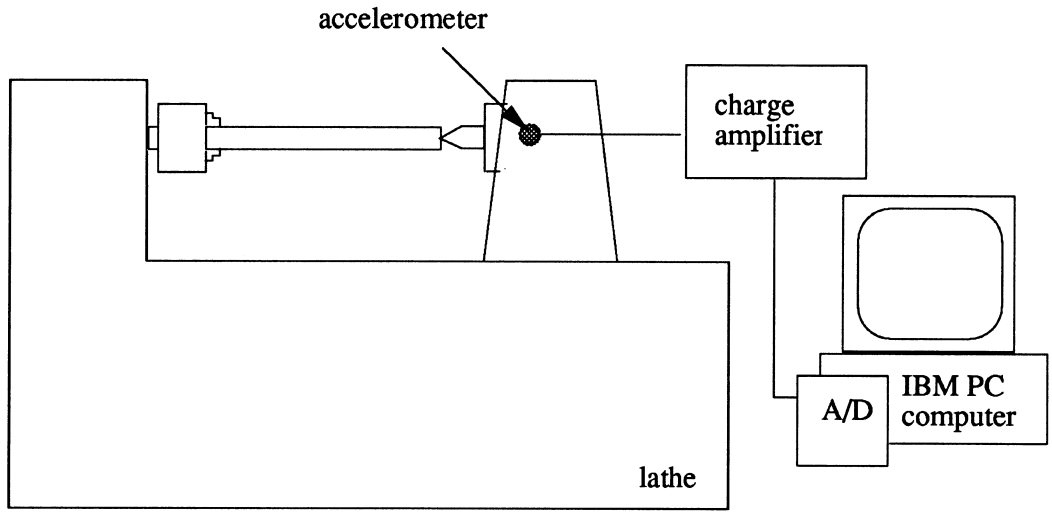


FIGURE 8.14 The experimental setup of chatter monitoring in turning.

TABLE 8.1 The Experimental Conditions for the Turning Example

Work	Length	600 mm
	Diameter	35 ~ 45 mm
	Material	#45 steel
Cutter	Rank angle	10 ~ 13°
	Flank angle	8 ~ 10°
	Material	Uncoated carbide
Cutting condition	Speed	660 RPM
	Feed	0.08 ~ 0.10 mm/rev.
	Depth of cut	0.5 ~ 1.0 mm

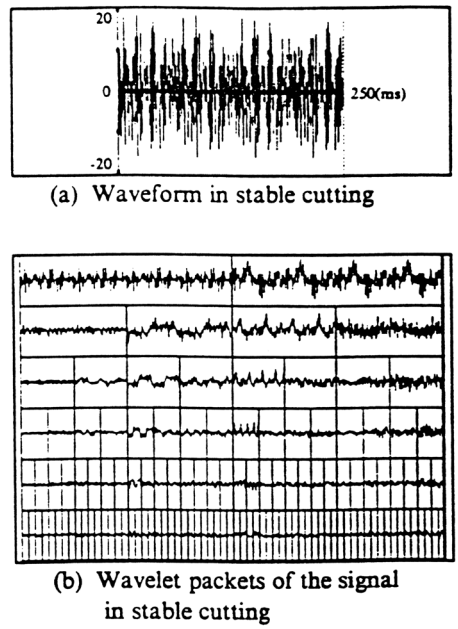
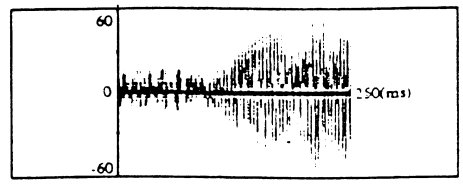
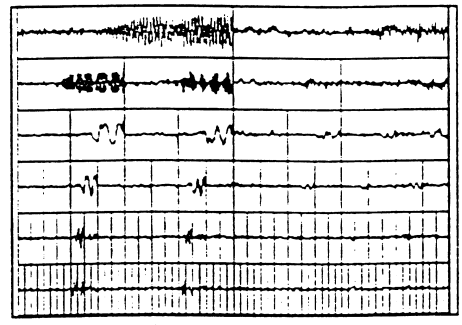


FIGURE 8.15 The vibration signal under normal cutting (a) and its wavelet transforms(b).

During stable cutting, hard spots of workpiece material, unsteady mechanical movements, and unbalance of rotating parts could all cause the fluctuation of cutting forces, which in turn caused vibration with small amplitude but the vibration was quickly damped out. Thus, the vibration signal was characterized by the small vibration amplitude in the time domain and wide band frequencies in the frequency domain. This was confirmed by our experiment as shown in Fig. 8.15a. In the interim period, the vibration signal is characterized by an increase in amplitude in the time domain and the excitation of machine tool vibration modes in the frequency domain. Figure 8.16a shows a typical signal from the interim period. When chatter was fully developed, the vibration signal was dominated by the machine tool vibration, as shown in Fig. 8.17a. Figures 8.15b, 8.16b, and 8.17b are the wavelet packet transforms corresponding to the signals in Figs. 8.15a, 8.16a, and 8.17a. It is seen that for stable cutting, the energy level in every packet is rather close, as shown in Fig. 8.15b. In the interim period, several wavelet packets contained a significantly larger amount of energy. For example, at the 5th resolution, the energy in packets $P_5^5(t)$ and $P_5^{13}(t)$ was much higher compared to the other packets on the same resolution as shown in Fig. 8.16b. When chatter was fully developed, the energy in packets $P_5^5(t)$ and $P_5^{13}(t)$ became predominant, as shown in Fig. 8.17b. From a physical point of view, the packets $P_5^5(t)$ and $P_5^{13}(t)$ were associated with the machine tool vibration modes. Therefore, they were sensitive to the onset of chatter and were selected as the feature packets for monitoring chatter in turning.

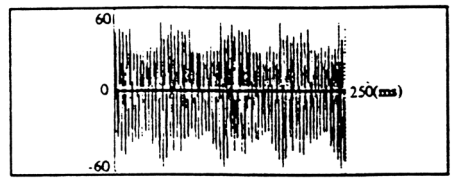


(a) Waveform in interim period

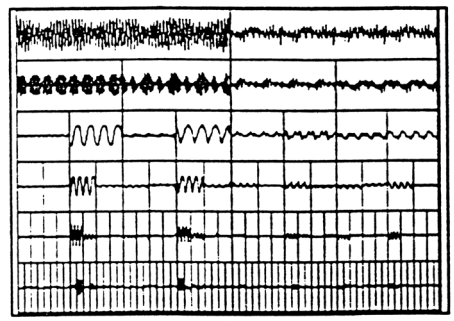


(b) Wavelet packets of the signal in interim period

FIGURE 8.16 The vibration signal under interim cutting and its wavelet transforms.



(a) Waveform in chatter



(b) Wavelet packets of the signal in chatter

FIGURE 8.17 The vibration signal under chatter and its wavelet transforms.

To evaluate the effectiveness of the selected features (the wavelet packets $P_5^5(t)$ and $P_5^{13}(t)$), the feature assessment criteria proposed earlier were used and the results are shown in Table 8.2. From the table, the following observations can be made.

1. The information lost from the original signal (in both the time domain and the frequency domain) was related to the compression ratio and the number of wavelet packets used. For example, using two feature packets, at the compression ratio 1/16, the information retained in the time domain was 74%

TABLE 8.2 Feature Assessment Values of the Vibration Signals under Chatter Conditions

	Compression Ratio						
	1	1/2	1/4	1/8	1/16	1/32	1/64
$\rho_{Qf}(0)$	1	0.95	0.91	0.82	0.74	0.66	0.35
$\gamma_{Qf}(265.5\text{ Hz})$	1	0.98	0.97	0.91	0.83	0.77	0.38
$\rho_{ef}(0)$	0	0.05	0.09	0.18	0.26	0.34	0.65
Wavelet packets	$P_0^1(t)$	$P_1^1(t)$	$P_3^2(t)$ $P_3^4(t)$	$P_4^3(t)$ $P_4^7(t)$	$P_5^5(t)$ $P_5^{13}(t)$	$P_6^{10}(t)$ $P_6^{25}(t)$	$P_6^{25}(t)$
Information retained in time domain	100%				74%	66%	
Information retained in frequency domain	100%				83%	77%	

and the information retained in the frequency domain was 83%. At the compression ratio 1/32, the information retained in the time domain was 66% and the information retained in the frequency domain was 77%. In general, when the compression ratio increased by 1/2, 6 ~ 9% of the information was lost. Hence, to retain more than 70% of the time domain information and 80% of the frequency domain information, the compression ratio 1/16 was chosen.

2. At the compression ratio of 1/16, the wavelet packets $P_5^5(t)$ and $P_5^{13}(t)$ represented 74% of the time domain information and 83% of the frequency domain information. This indicated that the information lost might be different in the time and frequency domains. Hence, if the time domain features are more important, then the time domain assessment criteria should be stressed. On the other hand, if the frequency domain features are more important, the frequency assessment criteria should be stressed. When monitoring chatter, frequency domain information is very important because at the onset of chatter, the machine tool vibration will dominate the vibration signal. $P_{Qf} \geq 0.70$ and $\gamma_{Qf} \geq 0.80$ represented a desirable combination. Consequently, using the automatic feature selection procedure presented in Section 3.3, the wavelet packets $P_5^5(t)$ and $P_5^{13}(t)$ were selected as the feature packets.
3. Based on the selected packets, $P_5^5(t)$ and $P_5^{13}(t)$, Fig. 8.18a shows the reconstructed signal under chatter. Compared to the original signal in Fig. 8.17a, it is seen that the two signals are indeed rather similar. From Fig. 8.18b, it is seen that the cross-correlation function $P_{Qf}(k)$ is close to 0.8, which clearly indicates the strong correlation. Figs. 8.18c and 8.18d are the power spectrum of $f(t)$ and the power spectrum of the reconstructed signal $Q(t)$, respectively. It is seen that the selected packets represent the major characteristics of $f(t)$ in the frequency domain as well.

As pointed out earlier, feature packets can be further compressed by indices such as RMS and peak-to-valley to facilitate monitoring decision-making. In summary, the chatter monitoring procedure is given below:

calculate $P_5^5(t)$ and $P_5^{13}(t)$;
calculate T_5^5 and T_5^{13};
if $T_5^{13} < 21$ or $T_5^{13} < 21$
  then stable cutting
  else
    if $T_5^{13} < 270$ or $T_5^{13} < 200$
      then onset of chatter
      else chatter
    endif
endif

Note that in the above procedure, the monitoring thresholds (21, 270, and 200) were obtained from experiment samples as shown in Figs. 8.15, 8.16, and 8.17. This procedure was tested using 12 different experiment samples under various cutting conditions and it maintains a 100% success rate.

Tool Wear Diagnosis in Drilling

The experiments for tool wear diagnosis in drilling were conducted on a horizontal drilling machine as illustrated in Fig. 8.19 with the experiment conditions summarized in Table 8.3.

Tool wear monitoring in drilling is a difficult task. First, the wear of a drill can be in various forms; for example, there may be uneven wear at different cutting edges. Second, the drill can be self-sharpening during drilling because of the change in the rake angle. In practice, the complexity of the problem is compounded by various factors such as machine tool run-out, fixture vibration, chip formation, coolant, and material inhomogenuity. Hence, even under identical cutting conditions, the life of a drill varies. The drill life can be judged by the power consumption (or torque). The life cycle can be divided into four states: new drill, initial wear (flank wear less than 0.1 mm), normal wear (flank wear around 0.15 mm), and worn drill (flank wear larger than 0.15 mm). Figure 8.20a shows the vibration signal (acceleration) corresponding to a new drill (the sampling frequency is 1 kHz). Figures 8.21a, 8.22a, and 8.23a are the vibration signals obtained when drilling the 52rd, 4600th, and 5100th hole, respectively, using the same drill, which corresponded to the other states of tool wear: initial wear, normal wear, and worn out. The corresponding wavelet packet

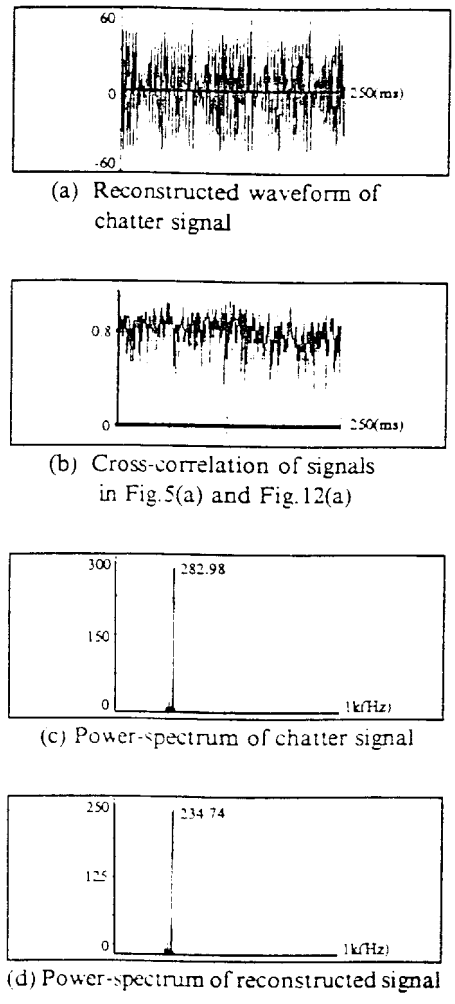


FIGURE 8.18 The reconstruction of the signal in Fig. 8.17 and reconstruction assessment.

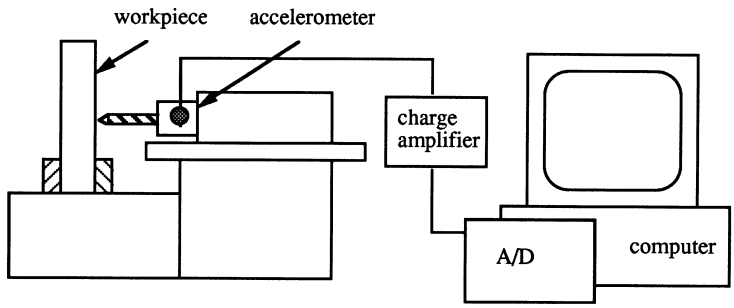
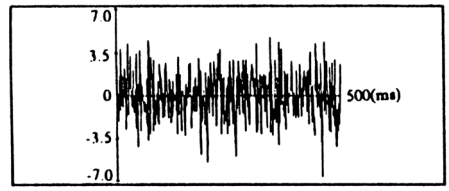


FIGURE 8.19 The experimental setup of tool wear diagnosis in drilling.

TABLE 8.3 The Experimental Conditions for the Drilling Example

Work	Depth of drilling	13.97 mm
	Diameter of drilling	10.67 mm
	Material	Cast iron
Cutter	Type	twist
	Relief angle	10°
	Material	Uncoated carbide
Cutting condition	Speed	1030 RPM
	Feed	0.0127 mm/rev.

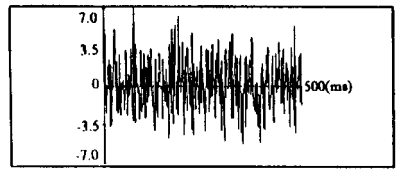


(a) Waveform in new drill condition

(b) Wavelet packets of the signal in new drill condition

FIGURE 8.20 The vibration signal and its wavelet transform when drilling using a new drill.

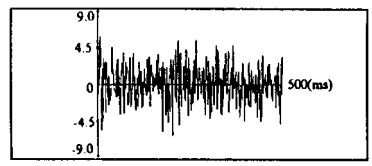
transforms were shown in Figs. 8.20b to 8.23b. From these figures, it is seen that the new drill produced smooth cutting and, hence, results in small vibration. Consequently, there is no predominant wavelet packet, as shown in Fig. 8.20b. Then, there is a period of time during which higher wear rate could be observed. This is due to the fact that the shape edges of a drill might be easily worn off and this period was called initial wear. At this time, the drill might produce unsmooth cuts and thus cause an increase in vibration. As a result, the wavelet packets P_5^9 and P_5^{25}, which corresponded to the machine tool vibration, contained a relatively larger amount of energy as shown in Fig. 8.21b. Next, the drill enters the normal wear state; it cut smoothly and, hence, as shown in Fig. 8.22b, there are no predominant wavelet packets. The normal wear state lasts a period of time during which the drill gradually wears out. Finally, when the drill is worn out, the cuts became rough. As shown in Fig. 8.23b, the energy in feature packets P_5^9 and P_5^{25} raised to a higher level.



(a) Waveform under initial worn drill

(b) Wavelet packets of the signal under initial worn drill

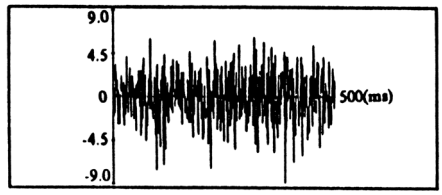
FIGURE 8.21 The vibration signal and its wavelet transform when drilling using a drill with initial wear.



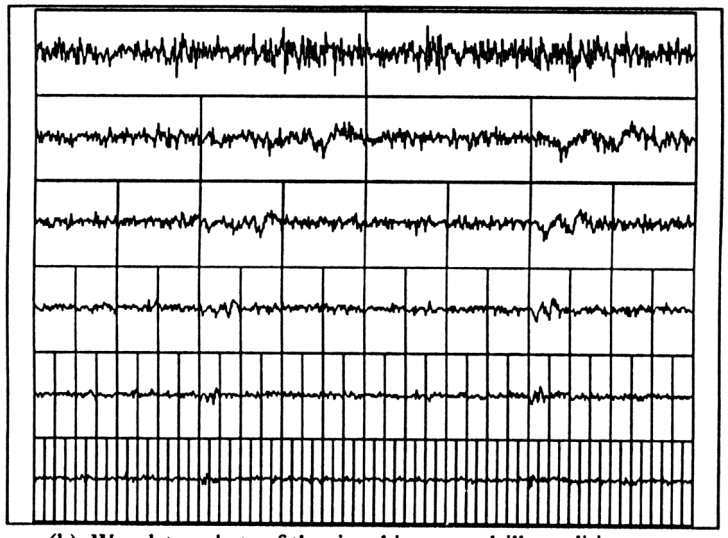
(a) Waveform in normal drill condition

(b) Wavelet packets of the signal in normal drill condition

FIGURE 8.22 The vibration signal and its wavelet transform when drilling using a drill with normal wear.



(a) Waveform in worn drill condition



(b) Wavelet packets of the signal in worn drill condition

FIGURE 8.23 The vibration signal and its wavelet transform when drilling using a worn drill.

From a total of 68 experimental samples, using the feature selection procedure presented in Section 8.3, two wavelet packets P_5^9 and P_5^{25} were identified as the feature packets for tool wear monitoring under the assessment criteria $P_{Qf} \geq 0.70$ and $\gamma_{Qf} \geq 0.90$. The selected packets were very effective. As shown in Fig. 8.24, which corresponds with to Fig. 8.23, the selected packets represented more than 70% information in the time domain, and the power spectrum of the reconstructed signal was nearly the same as that of the original signal (note, the frequency 256 Hz was the natural frequency of the machine tool).

To diagnose the state of tool wear, the feature packets $P_5^9(t)$ and $P_5^{25}(t)$ are further compressed using the peak-to-valley (T_5^9 and T_5^{25}) and crest factor (C_5^9 and P_5^{25}) of the packets, and then classified using the Fisher pattern recognition method described in Section 8.5. This is to set:

$$\mathbf{b} = [T_5^9, T_5^{25}, C_5^9, C_5^{25}]^T$$

Through the learning from 68 experiment samples, it was found that:

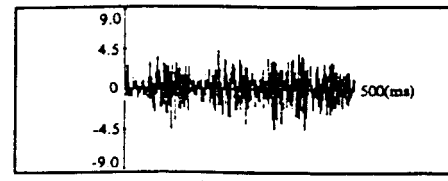
$$\mathbf{b}_{\text{new tool}} = [58, 61, 1.5, 1.3]^T$$

$$\mathbf{b}_{\text{initial wear}} = [110, 123, 2.5, 3.0]^T$$

$$\mathbf{b}_{\text{normal wear}} = [88, 96, 1.5, 1.3]^T$$

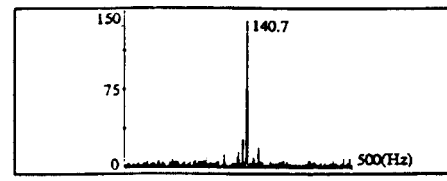
$$\mathbf{b}_{\text{worn tool}} = [220, 211, 2.5, 3.0]^T$$

Based on the test of 24 different experiment samples, the success of diagnosis of the state of wear is 100%.

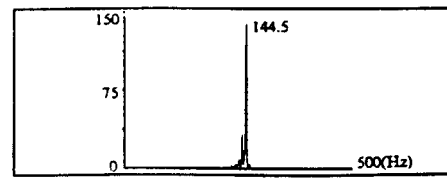


(a) Reconstructed waveform of vibrational signal in worn drill condition

(b) Cross-correlation of signals



(c) Power-spectrum of vibrational signal



(d) Power-spectrum of reconstructed signal

FIGURE 8.24 The reconstructed signal and assessment of the signal in Fig. 8.24.

8.7 Conclusions

Based on the discussions above, the following conclusions can be drawn.

1. The wavelet packet transform is a powerful tool for engineering monitoring and diagnosis. It can capture important features of the sensor signal that are sensitive to the change of system conditions (e.g., chatter and different states of tool wear) but is insensitive to the variation of process working conditions and various noises. Accordingly, accurate and reliable monitoring and diagnosis decisions can be made.
2. The wavelet packet transform decomposes a sensor signal into different components in different time windows and frequency bands. The feature wavelet packets are those containing the principle components of the original signal. The feature packets can be obtained using an automatic feature selection procedure.
3. The effectiveness of the selected packets can be assessed in both time and frequency domains. It is recommended that the selected packets should preserve 70% of the time domain information and 80% of the frequency domain information from the original signal. To do this, the maximum compression ratio is 1/32, because higher compression ratios mean more information loss.

4. The feature wavelet packets can be characterized by quantities such as peak-to-valley value, crest factor, mean, root mean squares, and variance. The quantities can be used for real-time monitoring. On the other hand, when these quantities are correlated to the various system conditions, it is necessary to use classification methods such as pattern recognition to diagnose the system conditions.
5. The presented method was tested by two practical examples: chatter monitoring in turning and tool wear diagnosis in drilling. Both tests resulted in 100% success.

Acknowledgment

The paper is partially supported by the Natural Science and Engineering Research Council of Canada Grant #OGP0121939.

References

Akansu, A. N. and et al., 1993, The binomial QMF–wavelet transform for multiresolution signal decomposition, *IEEE Trans. on Signal Processing,* 41(1), 13–19.

Chen, M. H., Lee, D., and Pavlidis, T., 1991, Residual analysis for feature detection, *IEEE Trans. on Pattern Analysis and Machine Intelligence,* 13(1), 30–40.

Cho, Y. S. et al., 1992, A digital technique to estimate second-order distortion using higher order coherence spectra, *IEEE Trans. on Signal Processing,* 40(5), 1029–1040.

Coifman, R. R. and Wickerhanster, M. V., 1992, Entropy-based algorithms for best basis selection, *IEEE Trans. on Information Theory, Special Issue on Wavelet Transform and Multiresolution Signal Analysis,* 38(2), 713–718.

Daubechies, I., 1988, Orthonormal based of compactly supported wavelet, *Communications in Pure and Applied Mathematics,* 41(7), 909–996.

Daubechies, I., 1990, The wavelet transform, time-frequency localization and signal analysis, *IEEE Trans. on Information Theory,* 36(9), 961–1005.

DeVore, R. A., Jawerth, R. and Lucier, B. J., 1992, Image compression through wavelet transform coding, *IEEE Trans. on Information Theory, Special Issue on Wavelet and Multiresolution Signal Analysis,* 38(2), 719–746.

Du, R., Elbestawi, M. A., and Wu, S. M., 1995, Computer automated monitoring of manufacturing processes. 1. Monitoring decision-making methods, *Trans. of ASME, J. of Eng. for Industry,* 114.

Du, R., Elbestawi, M. A., and Wu, S. M., 1995, Computer automated monitoring of manufacturing processes. 2. Applications, *Trans. of ASME, J. of Eng. for Industry,* May.

Elbestawi, M. A., Marks, J., and Papazafirou, T., 1989, Process monitoring in milling by pattern recognition, *Mechanical Systems and Signal Processing,* 3(3), 305–315.

Houshmand, A. A. and Kannatey Asibu Jr., E., 1989, Statistical process control of acoustic emission for cutting tool monitoring, *Mechanical Systems and Signal Processing,* 3(4), 405–424.

Hummel, R., 1989, Reconstruction from zero crossing in scale space, *IEEE Trans. on Acoustics, Speech, and Signal Processing,* 37(12), 2111–2130.

Li, D. and Mathew, J., 1990, Tool wear and failure monitoring techniques for turning—A review, *Int. J. Machine Tools Manufacture,* 30(4), 579–598.

Kandel, A., 1982, *Fuzzy Techniques in Pattern Recognition,* John Wiley & Sons. New York.

Mallat, S. G., 1989, A theory for multiresolution signal decomposition: the wavelet representation, *IEEE Trans. on Pattern Analysis and Machine Intelligence,* 11(7), 674–691.

Mallat, S. G. 1989, Multifrequency channel decomposition of images and wavelet models, *IEEE Trans. on Acoustics, Speech, and Signal Processing,* 37(12), 2091–2110.

Meyer, Y., 1989, Wavelets and operators, *Proc. of the Special Year in Modern Mathematical Analysis at the University of Illinois (1986–1987), London Mathematical Society Lecture Notes Series,* Cambridge Univ. Press, 37, 256–365.

Monostori, L., 1986, Learning procedures in machine tool monitoring, *Computers in Industry,* 7, 53–64.
Ornstein, D. S. and Weiss, B., 1993, Entropy and data compression schemes, *IEEE Trans. on Information Theory,* 39(1), 78–83.
Ranganath, S., 1991, Image filtering using multiresolution representations, *IEEE Trans. on Pattern Analysis and Machine Intelligence,* 13(5), 426–440.
Rioul, O., 1993, A discrete-time multiresolution theory, *IEEE Trans. on Signal Processing,* 41(8), 2591–2606.
Rioul, O. and Flandrin, P., 1992, Time-scale energy distributions: a general class extending wavelet transforms, *IEEE Trans. on Signal Processing,* 40(7), 1746–1757.
Tansel, I. N. and McLauglin, C., 1993, Monitoring drill conditions with wavelet based encoding and neural networks, *Int. J. Mach. Tools Manufact.,* 33(4), 559–575.
Tewfik, A. H., Sinha, D., and Jorgensen P., 1992, On the optimal choice of a wavelet for signal representation, *IEEE Trans. on Information Theory, Special Issue on Wavelet Transform and Multiresolution Signal Analysis*, 38(2), 747–765.
Wong, P. W., 1993, Wavelet decomposition of harmonizable random processes, *IEEE Trans. on Information Theory,* 39(1), 7–18.