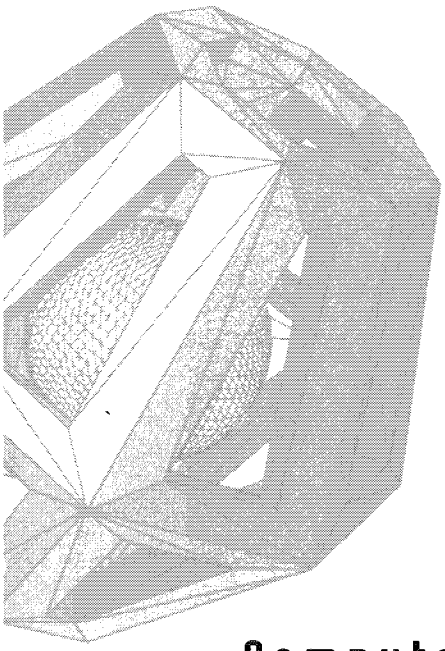
Vol. 5

Manufacturing Processes

Computer Aided and Integrated Manufacturing Systems

A 5-Volume Set

Cornelius T Leondes



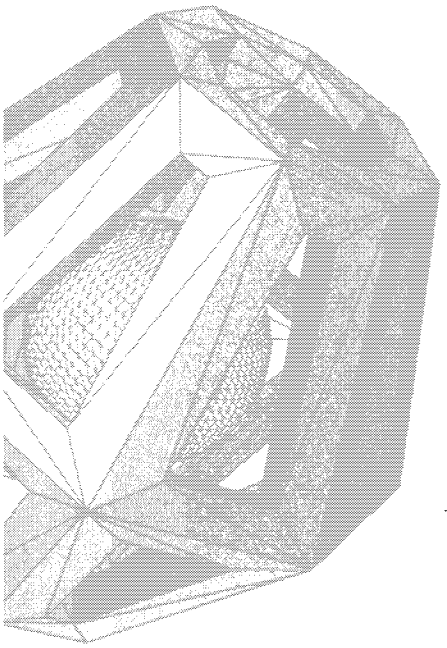
Vol. 5

Manufacturing Processes

Computer Aided and Integrated Manufacturing Systems

A 5-Volume Set

This page is intentionally left blank



Vol. 5
Manufacturing Processes

Computer Aided and Integrated Manufacturing Systems

A 5-Volume Set

Cornelius T Leondes
University of California, Los Angeles, USA

World Scientific

NEW JERSEY • LONDON • SINGAPORE • SHANGHAI • HONG KONG • TAIPEI • BANGALORE

Published by

World Scientific Publishing Co. Pte. Ltd.

5 Toh Tuck Link, Singapore 596224

USA office: Suite 202, 1060 Main Street, River Edge, NJ 07661

UK office: 57 Shelton Street, Covent Garden, London WC2H 9HE

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library.

COMPUTER AIDED AND INTEGRATED MANUFACTURING SYSTEMS
A 5-Volume Set
Volume 5: Manufacturing Processes

ISBN 981-238-339-5 (Set)
ISBN 981-238-979-2 (Vol. 5)

Typeset by Stallion Press

Printed by FuIsland Offset Printing (S) Pte Ltd, Singapore

Preface

Computer Technology

This 5 volume MRW (Major Reference Work) is entitled "Computer Aided and Integrated Manufacturing Systems". A brief summary description of each of the 5 volumes will be noted in their respective PREFACES. An MRW is normally on a broad subject of major importance on the international scene. Because of the breadth of a major subject area, an MRW will normally consist of an integrated set of distinctly titled and well-integrated volumes each of which occupies a major role in the broad subject of the MRW. MRWs are normally required when a given major subject cannot be adequately treated in a single volume or, for that matter, by a single author or coauthors.

Normally, the individual chapter authors for the respective volumes of an MRW will be among the leading contributors on the international scene in the subject area of their chapter. The great breadth and significance of the subject of this MRW evidently calls for treatment by means of an MRW.

As will be noted later in this preface, the technology and techniques utilized in the methods of computer aided and integrated manufacturing systems have produced and will, no doubt, continue to produce significant annual improvement in productivity — the goods and services produced from each hour of work. In addition, as will be noted later in this preface, the positive economic implications of constant annual improvements in productivity have very positive implications for national economies as, in fact, might be expected.

Before getting into these matters, it is perhaps interesting to briefly touch on Moore's Law for integrated circuits because, while Moore's Law is in an entirely different area, some significant and somewhat interesting parallels can be seen. In 1965, Gordon Moore, cofounder of INTEL made the observation that the number of transistors per square inch on integrated circuits could be expected to double every year for the foreseeable future. In subsequent years, the pace slowed down a bit, but density has doubled approximately every 18 months, and this is the current definition of Moore's Law. Currently, experts, including Moore himself, expect Moore's Law to hold for at least another decade and a half. This is impressive with many significant implications in technology and economies on the international scene. With these observations in mind, we now turn our attention to the greatly significant and broad subject area of this MRW.

"The Magic Elixir of Productivity" is the title of a significant editorial which appeared in the *Wall Street Journal.* While the focus in this editorial was on productivity trends in the United States and the significant positive implications for the economy in the United States, the issues addressed apply, in general, to developed economies on the international scene.

Economists split productivity growth into two components: Capital Deepening which refers to expenditures in capital equipment, particularly IT (Information Technology) equipment: and what is called Multifactor Productivity Growth, in which existing resources of capital and labor are utilized more effectively. It is observed by economists that Multifactor Productivity Growth is a better gauge of true productivity. In fact, computer aided and integrated manufacturing systems are, in essence, Multifactor Productivity Growth in the hugely important manufacturing sector of global economies. Finally, in the United States, although there are various estimates by economists on what the annual growth in productivity might be, Chairman of the Federal Reserve Board, Alan Greenspan — the one economist whose opinions actually count, remains an optimist that actual annual productivity gains can be expected to be close to 3% for the next 5 to 10 years. Further, the Treasure Secretary in the President's Cabinet is of the view that the potential for productivity gains in the US economy is higher than we realize. He observes that the penetration of good ideas suggests that we are still at the 20 to 30% level of what is possible.

The economic implications of significant annual growth in productivity are huge. A half-percentage point rise in annual productivity adds $1.2 trillion to the federal budget revenues over a period of ten years. This means, of course, that an annual growth rate of 2.5 to 3% in productivity over 10 years would generate anywhere from $6 to $7 trillion in federal budget revenues over that time period and, of course, that is hugely significant. Further, the faster productivity rises, the faster wages climb. That is obviously good for workers, but it also means more taxes flowing into social security. This, of course, strengthens the social security program. Further, the annual productivity growth rate is a significant factor in controlling the growth rate of inflation. This continuing annual growth in productivity can be compared with Moore's Law, both with huge implications for the economy.

The respective volumes of this MRW "Computer Aided and Integrated Manufacturing Systems" are entitled:

Volume 1: Computer Techniques
Volume 2: Intelligent Systems Technology
Volume 3: Optimization Methods
Volume 4: Computer Aided Design/Computer Aided Manufacturing (CAD/CAM)
Volume 5: Manufacturing Process

A description of the contents of each of the volumes is included in the PREFACE for that respective volume.

Henceforth, Manufacturing Processes will be continually improved and updated with a view towards enhancing productivity significantly, and this is the subject of Volume 5. Production planning will become increasingly effective and will greatly enhance the production process in major areas such as electronics manufacturing systems. Mold design is a complicated process and powerful computer techniques are presented for manufacturing systems in this area. Computer modelling for the determination of optimum manufacturing strategy will increasingly become a way of life in manufacturing systems, and this greatly significant topic is discussed in this volume. Machining operators are one of the most prevalent methods utilized in manufacturing systems, and economic optimization methods in CAM (Computer Aided Manufacturing) systems as they relate to machining operations are treated comprehensively in this volume. An absolutely essential process in manufacturing processes is the construction of solid models in CAD (Computer Aided Design), and highly powerful computer techniques for accomplishing this are presented. Manufacturing systems are generally sophisticated Mechatronic Systems, i.e. the optimal integration of electronic and electromechanic systems, and the computer techniques and applications required in such Mechatronic Systems are discussed in detail. These and numerous other significant techniques are treated rather comprehensively in this volume.

As noted earlier, this MRW (Major Reference Work) on "Computer Aided and Integrated Manufacturing Systems" consists of 5 distinctly titled and well-integrated volumes. It is appropriate to mention that each of the volumes can be utilized individually. The significance and the potential pervasiveness of the very broad subject of this MRW certainly suggests the clear requirement of an MRW for a comprehensive treatment. All the contributors to this MRW are to be highly commended for their splendid contributions that will provide a significant and unique reference source for students, research workers, practitioners, computer scientists and others, as well as institutional libraries on the international scene for years to come.

This page is intentionally left blank

Contents

CHAPTER 1

TECHNIQUES AND APPLICATIONS OF PRODUCTION PLANNING IN ELECTRONICS MANUFACTURING SYSTEMS

JOUNI SMED, MIKA JOHNSSON*, TOMMI JOHTELA and OLLI NEVALAINEN
Turku Centre for Computer Science (TUCS)
and Department of Mathematical Sciences,
University of Turku, FIN-20014 Turku, Finland.
*Email: *johnsson@cs.utu.fi*

The electronics industry is a major part of modern manufacturing, and electronic systems play an increasingly important role in the majority of today's products. Electronic systems are usually implemented with printed circuit boards (PCBs), and, consequently, PCB assembly has become an important sector of the electronics manufacturing industry overall. However, operating effectively in this industry is becoming more difficult as the companies must compete with high quality standards, rapidly changing technologies, short production cycles, and increasing product variety and complexity. In addition, the capital equipment cost of electronics assembly industry facilities are high in comparison to the usual turnover of a company. As a result, production planning decisions need to made more and more frequently due to continuous changes in the production conditions.

In this work we discuss production planning in electronics assembly—and in particular, in PCB assembly. Our intention is to identify the typical problems arising from production planning and to give a survey of the solution methods suggested in the literature. In addition to this theoretical perspective, we will briefly review applications designed for production planning in PCB assembly.

This work is organized as follows. We begin with an introduction to flexible manufacturing systems in general and present a framework for production planning systems in Sec. 1. Next, we study the fundamentals of PCB assembly process in Sec. 2 and survey the relevant literature in Sec. 3. In Sec. 4 we review existing commercial applications for production planning, and in Sec. 5 study more closely one of these systems. Finally, in Sec. 6 we sum up the discussion and outline few important topics for the future research.

Keywords: Production planning; electronics manufacturing systems; flexible manufacturing systems; automated manufacturing.

1. Introduction

Let us begin our discussion by recalling some basic concepts of industrial production. Five basic types of production operations can be classified according to the degree repetitiveness involved[20]: project, jobbing, batch, flow, and process. The *project form* includes large-scale complex products, and it involves the allocation

and coordination of large-scale input to achieve a unique product. *Jobbing* describes a situation where the manufacturing of a whole product is considered as one operation and the work must be completed on each product before starting on the next. In the *batch form*, the volume of products to be manufactured is larger than in jobbing. A regular and consistent demand for a product such that the item can be produced for stock identifies the *flow production*. Lastly, the *process production* requires that the material is involved in a continuous process.

The type of production affects also the plant layout. There are three basic forms of layout[20]: process, product, and group. The *process layout*, in which all the plants associated with a particular type of process are grouped together, is typical in jobbing and batch production. In the *product layout*, which occurs in flow and process forms of production, the plant is laid out according to the sequence of processes required by the product. A *group* (or cellular) *layout* is typical in batch production, and it involves the recognition that many of the products have similarities in their makeup and they can thus be grouped into (product) families.

There are two basic principles for decomposing a manufacturing facility into subsystems: decomposition based on processes and decomposition based on products.[42] *Process based facility decomposition* leads to equipment being arranged into functional machining areas (*work centers*) dedicated to general manufacturing processes (i.e. *job shop*). *Process planning* is the systematic determination of the detailed methods by which parts can be manufactured from raw material to finished products. *Computer-aided process planning* (CAPP) *determines a set of instructions and machining parameters required to manufacture a part and prepares data for production planning and scheduling activities.*[85] As Zijm and van Harten[120] observe, CAPP bridges the gap between *computer-aided design* (CAD) and *computer-aided manufacturing* (CAM), and thus CAPP represents the "/" in CAD/CAM. *Process control* refers to the automatic monitoring and control of a process by an instrument or system configured or programmed to respond appropriately to process feedback.[102]

Product based decomposition utilizes the principle of group technology by dedicating machines to cells in order to produce the associated families of parts (i.e. *cellular manufacturing*). *Production planning* refers to the *process of establishing strategies for producing finished products so that manufacturing resources are used efficiently.* When there is a large number of production variables and a long planning horizon, the problem can be approached by breaking it hierarchically into a series of decision levels.[109] *Production control* is the *systematic planning, coordination and direction of all manufacturing activities to ensure that products (of adequate quality) are made on time and at reasonable cost.*[102] To summarize, in production planning we first *make a plan* anticipating the future events and after that *follow the plan*, whereas in production control we simply *react to the events* as they occur during the production.

In the remainder of this section we discuss flexible manufacturing systems (FMSs) in Sec. 1.1, and develop a common methodology for building practical production planning systems in FMSs in Sec. 1.2.

1.1. *Flexible manufacturing systems*

In the late 1950s, several ideas for improving manufacturing began to surface. One of the earliest was the idea of group technology for manufacturers who had to make a variety of different but similar parts. In some types of industry—for example, chemical and oil-refining industries—*automated manufacturing* has a long history, but in the batch-manufacturing industries—like the metalworking industry and the electronics industry—the concept of automated manufacturing was introduced in the early 1970s. By then, a number of technological developments, in particular flexible manufacturing systems, offered solutions to the problems of job shop environments.

Automated manufacturing has a wide variety of potential benefits to offer. One of the most important advantages is the increased ability to respond to changes in demand and changes in the products, which is essential in the today's view of short production cycles. Other advantages include shorter lead times, reduction in the work-in-process levels and improved machine utilization. At the same time it is not an easy task to fully attain these possibilities, because the reality of the shop floor rarely coincides with the theoretical models (as we shall see in Sec. 1.2.1).

A *flexible manufacturing system* (FMS) aims at achieving a similar level of effiency for manufacturing several different product types as in the mass production of a single product type. An FMS comprises a *group of programmable production machines integrated with automated material handling equipment which are under the direction of a central controller to produce a variety of parts at non-uniform production rates, batch sizes and quantities.*[53] The machines or work stations are used to perform operations on parts, and each operation requires a number of tools that can be stored in the limited capacity tool magazine of the machines. An automatic tool interchanging device switches the tools during production. Because this interchange is relatively quick, the machine can perform several operations with *virtually no setup time* between the operations, if the required tool is present in the tool magazine.

Electronics assembly (especially printed circuit board assembly) plants are usually FMSs. However, the terminology associated with FMS, which originates from the metal industry, can be somewhat confusing when applied to electronics assembly. For example, the concept of "tool" refers to a *feeder*, which contains the electronic component to be mounted, rather than the actual tool (or *nozzle*) which does the printing operation. We discuss the technical aspects of electronics assembly at greater length in Sec. 2.

The most prevalent analytical approach to real-time FMS control attempts to hierarchically decompose the problem into a number of more easily manageable subproblems, which relate to a variety of decisions concerning *long-term*, *medium-term* or *short-term planning*. One of the main reasons for decomposing the general planning problem is that this problem is too complex to be solved globally, whereas it is easier to solve each subproblem one at a time. The solution to the global problem can then be obtained by solving the subproblems successively. Naturally,

this solution is not likely to be globally optimal, even if all subproblems are solved to optimality. Nonetheless, this approach is a productive and popular way to tackle hard problems.

A typical hierarchical classification scheme of FMS[28] discerns:

(1) *Strategic level* or *long-range planning* which concerns the initial deployment and subsequent expansion of the production environments (e.g. the *design and selection of the equipment and of the products* to be manufactured).
(2) *Tactical level* or *medium-range planning* which determines the allocation patterns of the system production capacity to various products so that external demands are satisfied (e.g. by solving *batching* and *loading problems*).
(3) *Operational level* or *short-range planning* which coordinates the shop floor production activities so that the higher level tactical decisions are observed (e.g. by solving *release* and *dispatching problems*).

Maimon and Shtub[82] and Johnsson[54] relate these objectives to electronics assembly: in the strategic level, the planning focuses on determining the best set of production equipment for the operation (e.g. running a simulation on how much money should be invested in new equipments and what kind of machines should be purchased[35]). These decisions are usually made on an economical basis, and they are revised over long operational periods, typically measured in several months.[34,66] At the tactical level, the decisions concern machine and line configurations, production schedules, batch sizes, and work-in-process levels. Finally, the operational level addresses the day-to-day operation of the equipment (e.g. how to manufacture a product). *The tactical and operational problems have to be solved frequently, and consequently, the existing production planning systems concentrate on these levels.*

Notwithstanding the similarities, there are also differences between PCB assembly and FMSs. Klegka and Driels[66] study four cases of PCB assembly and conclude that FMS analysis is inappropriate for PCB assembly system analysis: FMS analysis chooses among multiple paths through manufacturing systems, and the best path depends on the state of the system that is changing as the workload changes. In PCB assembly, however, the production cycle is fixed (e.g. receiving inspection, panel preparation, screen paste, paste volume inspection, component placement, solder joint reflow, visual inspection, and X-ray inspection) and that results a single, sequential, and somewhat deterministic manufacturing process. Hence, a linear cost model may be adequate to evaluate and identify the cost elements of a PCB manufacturing system.

Zhou and Leu[119] list three features of PCB assembly distinct from conventional systems for automated assembly of mechanical parts:

(1) Each PCB requires numerous insertions, and the activities for each board are highly repetitive.

(2) There is no strict sequence which has to be followed, and the components can be inserted (almost) in any order (i.e. they have no or just a few precedence constraints).
(3) Only a single manipulator can operate at the time on the board even if the machine has multiple manipulators (e.g. see Ref. 46).

Jain *et al.*[52] discuss setup problems in FMSs and PCB assembly. They translate the tool setup problem of FMSs into PCB assembly system and show that the FMS formulation is a relaxed version of PCB component setup problem: in PCB assembly the tooling is more constrained due to the restrictions on the width of the component reels.

1.2. *Production planning in FMSs*

Despite the differences mentioned earlier, we will, for the remainder of this section, regard PCB assembly as an FMS. In Sec. 1.2.1, we begin with a discussion of the common problems apparent in the theoretic approaches to construct a practical production planning system. After that, in Sec. 1.2.2, we suggest a methodology for overcoming these problems and describe a general framework for modeling production planning problems.

1.2.1. *Problems in constructing a practical production planning system*

According to Ammons *et al.*[9] the control of an FMS requires a complex interaction of two components:

(1) Computers to perform automated control and routing activities.
(2) Humans to supervise the automation, to monitor system flows and output, to intervene in the unexpected operation of the system, and to compensate the effect of unanticipated events.

Especially in dynamic production environments (i.e. in FMSs which are subject to limited resources, random machine failures or multiple optimization criteria) the problem of controlling and scheduling the production process is best tackled by a synergy of the computer's scheduling algorithms and the human's effective internal heuristics. In this "interactive scheduling" the production planner remains in control and is able to affect the scheduling process by using his experience and intuition via computer support. In other words, the production planning system should act as a decision support for the production planner.

However, literature references to practical systems where this interaction has been realized are rare, and the models—even if based on reality—tend to be oversimplified. According to Saygin *et al.*[96] the existing software tools are typically (1) too slow and cannot react to changing dynamic shop floor conditions; (2) based on simplistic formulations of the reality that ignore important constraints; (3) based on a single objective function or simplistic trade-offs; and (4) difficult to install and

integrate into preexisting commercial shop floor systems. In general, the gap between theory and practice can usually be attributed to the following factors:

(1) Researchers fail to address the right problems.
(2) The given solutions are too complex to use.
(3) Findings are presented in terms that are foreign to the practitioners.
(4) Researchers focus on certain problems and omit other, often more important issues.
(5) The realities of the shop floor are ignored.

As Johnsson notes,[54] these observations are valid in electronics assembly, where problems are usually tackled by first modeling an existing problem, then finding a solution method to the problem, and after that validating both the solution method and the model by solving some randomly generated artificial test cases. However, this approach does not shed much light on the *practicality* of the method.

In addition to interactivity, real-world scheduling problems usually differ (and often quite radically) from the mathematical models presented in literature. Pinedo lists twelve differences[91]:

(1) Theoretical models assume that the scheduling problem is static, whereas in the real world new jobs to be scheduled can emerge at any time and the schedule is constructed without a perfect knowledge of the near future. The *production environment is dynamic* by nature (e.g. jobs may arrive unexpectedly, urgent prototype series may cut in the predefined sequence, machines may break down or have a temporary reduction in the production rate, or the required components may not be available at the present time).
(2) Resequencing problem is rarely addressed in literature even though it is present in most of the actual problems. Production planning is based on a *rolling horizon*, which leads to *rescheduling* or *reactive scheduling*, where the schedule is constantly updated and revised to meet events occurring randomly.
(3) In the real world, *production environments are more complicated than the models presented in literature*, which often disregard machine, job and time dependent processing restrictions and constraints.
(4) The mathematical models assume that the weights or priorities of the jobs are fixed and do not change over time; in practice, the *weight of a job often fluctuates over time* (e.g. a low-priority job may become suddenly a high-priority job).
(5) *Preferences* are usually not taken into account in mathematical models. In reality, even if a job can be scheduled on a given machine, there may be a preference to schedule it on another one.
(6) Most theoretical models overlook the *machine availability constraints* and assume that machines are available at all times, whereas the real-world production plants have deterministic and random processes which prevent machines

from operating (e.g. shift patterns, preventive maintenance, breakdowns and repairs).

(7) Most penalty functions considered in the literature are piecewise linear (e.g. the tardiness of a job or the unit penalty), whereas, in practice, there usually exists a committed *duedate.*

(8) Theoretical research tends to focus on models with a single objective. In the real world, there are usually *a number of objectives*, whose weights may vary over time and may even depend on the subjective preferences of the production planner in charge.

(9) In practice, whenever the workload appears to be excessive and the due dates appear to be too tight, the problem can be tackled by assigning *extra shifts* and *scheduling overtime.*

(10) The stochastic models studied in the literature usually assume special *processing time distributions* (e.g. exponential distribution). In automated assembly, the processing time is fixed with a very high probability, and with a very low probability there is an additional random time that is exponentially distributed with a very large mean (i.e. if a robot performs a task, the processing time is fixed, and if, by accident, something goes wrong, the processing time immediately becomes significantly larger).

(11) *Successive processing times* on the same machine tend to be highly positively correlated in practice, whereas theoretical models usually assume that all processing times are independently drawn from given distributions.

(12) In practice, the *processing time distribution* may be subject to change due to learning or deterioration.

In spite of the differences between the real-world and the mathematical models, Pinedo notes that the general consensus is that the theoretical research done in the past has not been in vain, but it has provided valuable insights into many scheduling problems. These insights have proven to be useful in the development of the algorithmic framework for a large number of real-world production planning systems.

1.2.2. *Structure of a production planning system*

The methodology for solving the production planning problems can be divided into four stages:

(1) Familiarization with the problem environment.
(2) Modeling the problem.
(3) Designing and implementing algorithms to solve the modeled problem.
(4) Integrating the algorithm to an existing system or including it in a new system.

The far-reaching decisions made in the initial stages influence the overall usability of the system. For example, if the model fails to represent the important aspects of the

real-world problem, no algorithm (no matter how cleverly designed and effectively implemented it is) can give results which would satisfy the production planner. In our experience, this modeling process cannot be overlooked nor its importance underestimated: a poor algorithm solving an accurately modeled problem gives better real-world results than an accurate algorithm solving poorly modeled problem.

After the initial familiarization stage, the construction of a production planning system begins with building a model which represents the production environment. At the same time one must bear in mind that this model is always an idealization of the actual problem: a coarse model may be easier to understand but it may lack some important aspects, whereas a detailed model may be a more accurate representation but much harder to understand. Because of this duality there are two approaches for using the model: if there is uncertainty about the accuracy of the model, we may want to grant the final decision to a human user, and in this case the model is used to point out the important aspects of the actual problem and possibly for suggesting some solutions. An alternative approach is to solve the problem by using an algorithm which utilizes an objective function based on the model for evaluating the solutions.

Figure 1 illustrates the role of the model in this scheme. A system based on visualization allows the production planner to interact and analyze the schedule, whereas an algorithm driven system solves the given problem efficiently and is independent from the user. Although both approaches have their benefits, extremes should be avoided when designing a production planning system. An algorithm is capable of solving a combinatorial problem inexhaustibly, whereas a human tends to try only few possible solutions before choosing one. Instead, a human usually has some "outside" knowledge about the reality concerning the problem, whereas the algorithm "sees" nothing but the model. Therefore, the usability of a production planning system, in essence, depends on the balance between these two points of view: the computer should provide the user with sufficient support for making the actual decision (e.g. generate a number of good schedules from which the

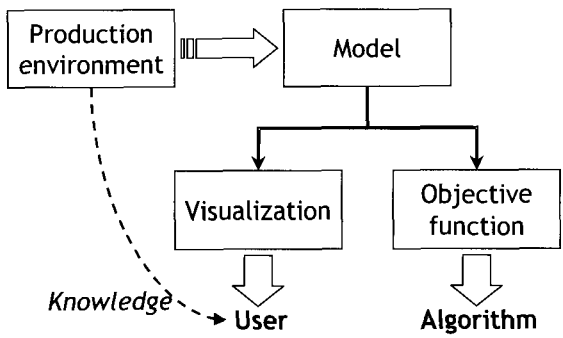


Fig. 1. A model of the production environment can be used as a basis for visualization or for calculating an objective function.

user chooses—and possibly refines—one for the production). Saygin *et al.*[96] conclude that the human scheduler should remain in control and be able to affect the scheduling process, because the model is an abstraction of the reality and, therefore, cannot capture all the characteristics of a problem, and because it is hard to transcribe the entire knowledge of the human scheduler in a computable form. Ammons *et al.*[9] express a similar view: "an 'optimal' real-time scheduling system is one that effectively combines computer scheduling algorithms and artificial intelligence methodologies within the context of the versatile capabilities of the human supervisor". Also, Martin-Vega[86] lists the integration of human and technical resources to enhance workforce performance and satisfaction as one of the six grand challenges for future research.

Figure 2 gives a more detailed view of the structure of a general production planning system. There are two ways, which correspond to the division shown in Fig. 1, to interact with the system: either directly by altering the production plan or indirectly by controlling the algorithm with the objective function and parameter settings. In the former case, the user makes alterations in the graphical representation of the production plan; the system updates the production plan accordingly or informs the user if the suggested change violates some hard constraint of the model. In the latter case, the user adjusts the objective function by setting weights for different criteria. The objective function is then used by an optimization algorithm, which generates a new production plan. After that, the new plan is simulated in order to discern predefined characteristics (e.g. lateness, earliness, workload, line balance, buffer sizes; see Refs. 35, 51, 71 and 97), which are used in the next iteration of the objective function and can be visualized to the user.

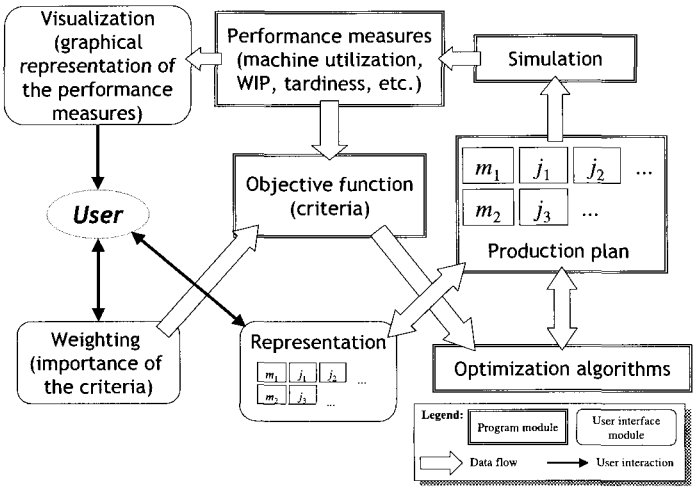


Fig. 2. The interaction between the components of a general production planning system.

Essentially, the modules represented in Fig. 2 are self-contained and connected by well-defined interfaces. For example, the optimization algorithm alters the production plan, which gives feedback whether the alteration violates any of the hard constraints, and receives an evaluation of the new plan from the objective function. Apart from these definements, the algorithm can be designed and implemented independently from the rest of the system.

Smith and Peters[101] present a more general framework for production planning and control systems. They observe that the development of FMS control systems is still implementation-specific and no tools exist to automate this development process. As a result, changes to the systems are often difficult to make, which has lead to many instances where FMS installations have failed to live up to expectations due to the inflexibility of the underlying control software. As a solution, Smith and Peters present an FMS control system concept, which comprises three components: a resource model instance (which combines the system configuration and the production requirements), a decision maker module, and an execution module. The *resource model instance* provides structural information required for constructing the other two modules, as well as the operational information required to run the system. The *decision-making module* decomposes the production requirements into specific instructions for the *execution module*, which interacts with the physical equipment and personnel on the shop floor to implement the given tasks. Within this general framework, the model of Fig. 2 corresponds to the decision-making module and clarifies the interaction between the system and the user.

2. Printed Circuit Board Assembly

A common characteristic in the printed circuit board (PCB) assembly industry is that customers demand more functions and flexibility each year in the products they buy. In addition, consumers expect reliable and cheap products, and thus a common goal in the PCB assembly industry is to put more functions into a board with the same size and cost.[108]

PCB assembly requires complete agility and reliability, which are only achievable with the use of robotics.[49] Manual assembly methods may provide the needed flexibility, but they cannot provide the reliability and speed of robotic automation. When properly tooled, robotic assembly allows quick change from one product to another, handling a higher mix of products with reliability rates well in excess of non-robotic systems. PCB assembly is characterized by designs that range from simple and low-value board assemblies to very complex and high-value board assemblies. Production volumes for different products vary in a very wide range—from millions to less than ten. One assembly system may encounter the assembly of PCBs with frequent design changes in small-batch production, whereas another system may assemble PCBs with a design that is fixed for six months or longer.

A recent development in PCB assembly is the growing role of *contract manufacturing*. Many original equipment manufacturers (OEMs) have abandoned the

assembly line in favor of outsourcing the manufacturing functions to contract manufacturers (CMs). CMs differ from OEMs that they build a variety of products for many different customers, whereas OEMs build only their own products. Despite the wider product variety and more dynamic product demand, CMs are expected to operate more efficiently than OEMs. This trend further emphasizes the importance of developing better production methods and systems.[84]

This section is organized as follows: In Sec. 2.1 we present the technical fundamentals and concepts of PCB assembly (for further details, see Refs. 43, 70, 102, 110 and 118).Section 2.2 describes most common machine types used in PCB assembly. Finally, in Sec. 2.3, we concentrate on different plant layouts.

2.1. *Fundamentals*

A *printed circuit board* (PCB)—or *printed wiring board* (PWB)—is a substrate of a glass fabric impregnated with a resin (an organic polymer which, when mixed with a curing agent, crosslink to form a thermosetting plastics; usually epoxy). A PCB consists of one or more layers of metal conductors and insulating material that allow for electronic components to be electronically interconnected and mechanically supported. A PCB of smaller dimension is commonly referred to as a *card*. A *panel* is an array of (usually identical) separate circuits fabricated on a single substrate.

The simplest form of PCB is the single-layer, single-sided board, which contains metalized conductor on one side of the board only. Greater levels of complexity and component density can be achieved by making double-sided and multi-layered boards. In double-sided assembly, the PCB is assembled with components on both sides of the substrate, and multilayering permits tracks to cross over one another, giving the designer more freedom in component layout.

Electronic components are either inserted through holes (e.g. griplet, axial and radial components) in the copper tracks and soldered in position, or are placed directly on to the surface of the board and soldered. These two distinctly different methods of manufacturing PCBs have given rise to different branches of manufacturing technology. The conventional method is known as *through hole plated assembly*, which is still popular for many applications, especially for low volume and manual assembly. Modern *surface mount technology* (SMT) utilizes smaller "flat" components which are well suited to automated assembly process. They are common in small consumer products (e.g. cellular phones), whereas in the larger ones (e.g. televisions and computer monitors), where the competitive product price is a key factor, through-hole components are still widely used.

Components have also other properties that affect the assembly process. The size of the component defines the recognition camera type, the feeder size and the nozzle (or tool) which must be used when the component is handled. Furthermore, component polarity, orientation in the input tape and different handling speeds (e.g. pickup, recognition, placement and turret) affect the operation of the insertion machines.

2.2. *Insertion machines*

Almost all machine types on the market operate in a similar fashion: the substrate is either placed (by the operator) or automatically transported to the staging area. After that, the components are "picked" from assigned pickup bin locations by vacuum and usually realigned either mechanically or optically before they are placed into the appropriate locations on the board. McGinnis *et al.*[87] recognize five fundamental operations common to all machines:

1. Positioning for retrieval of a component from feeder.
2. Retrieving the component from feeder.
3. Transporting the component from the feeder to the circuit board.
4. Positioning for placing the component on the circuit board.
5. Placing the component on the circuit board.

Some machine types are flexible in the sense that they can handle a wide range of different substrate sizes as well as a wide range of different component types, whereas others are restricted to a condensed set of components, which they can operate at a much higher speed.

A *feeder* supplies the placement head with components in the proper orientation. Notwithstanding the machine type, the feeder capacity of the machine is usually expressed in the number of 8 mm tape feeders, which is used in almost all currently available machine types. Other feeder types include a stick (or tube) of components, a vibratory slope feeder, and a tray feeder. A *component setup* comprises the required operations to replace one tape feeder to another, and a *machine setup* comprises the required component setups, width adjustments, tooling plate changeovers and printing program updates to change manufacturing from one PCB type to another.

The three most common machine types are:

(1) *Insertion machines* which have either a fixed head and a moving table or a moving head and a fixed table (Fig. 3). The printing head is connected to only one feeder, and a separate machine (which is often called a *sequencer*) produces an appropriate feeder tape, if different component types are needed.
(2) *Pick-and-place machines* which have a moving printing head, a fixed table and fixed feeders (Fig. 4). The head travels to pick a component from the feeder slot, moves it to the component insertion location, prints the component, and finally moves back to the next feeder location.
(3) *Rotary turret machines* which have moving insertion heads, a moving table and moving feeders (Fig. 5).

Currently, the rotary turret machine is the fastest machine type in common use; it can insert one component in less than 0.1 seconds and with a 0.01 mm accuracy. Such a high-speed surface mount component handler and placer is also known as a *chip shooter*.

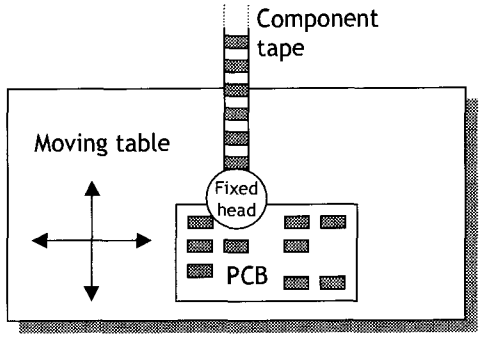


Fig. 3. Insertion machine.

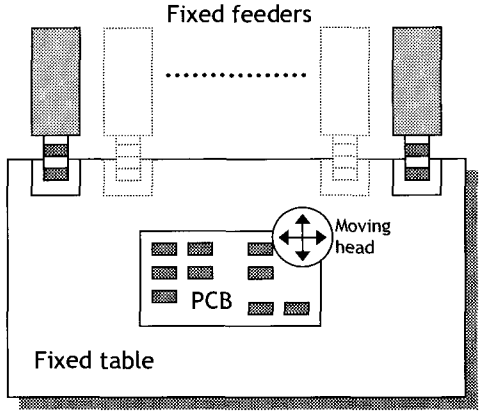


Fig. 4. Pick-and-place machine.

There are several variations of the basic machine types: Some machines have multiple insertion heads, duplicated feeders and duplicated tables.[1,46,112] Moreover, the insertion head may include several different nozzles to make it possible to operate with different component types; however, a nozzle change may require some time and slow down the overall operation speed. Finally, in some machines submachine units are added to increase the overall capacity.

2.3. *Plant layout*

Wittrock[115] presents the *flexible flow line* (FFL) environment which comprises several machine banks or production phases. In this model, the machines in a single machine bank are identical with each other, and a product can be processed in any machine belonging to the machine bank, or it can skip the phase altogether. Each product passes the phases in a predefined order, and the transfer between stages is accomplished with the help of magazines or some other form of transport. The setup time between different products, which is assumed to be negligible, is

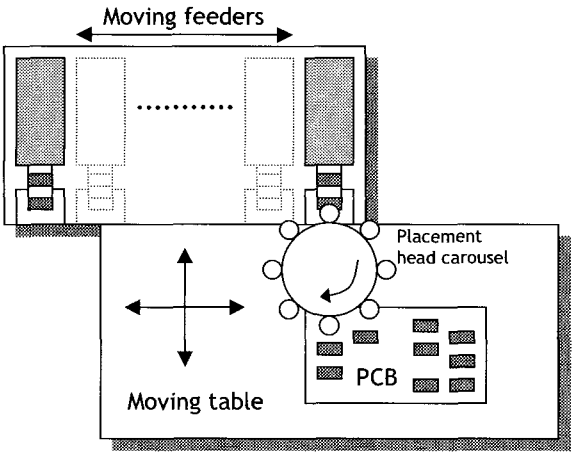


Fig. 5. Rotary turrent machine.

ignored. Hence, the processing time is a function of the processed product and the phase.

Johnsson *et al.*[57] introduce a *generalized flexible flow line* (GFFL) environment, which is a generalization of the FFL. The GFFL environment also comprises successive machine banks, but the type of the machines can vary even inside a particular machine bank (unlike in FFL). The machine type defines the speed of the machine, and thus the processing time in GFFL is a function of the product and the machine type. Moreover, setup times are also taken into account (unlike in FFL).

A typical electronic assembly line layout resembles GFFL, because the production is usually organized in successive workphases. Figure 6 gives an example of an existing production plant, where the phases are determined by the component type inserted in the corresponding phase[61] (similar plant layouts are described in Refs. 51, 64 and 76). In this case, both through-hole and surface mounted components are assembled, and the phases are organized such that griplets are printed first on the board. Axial components are inserted in the next phase, followed by radial components. Finally, the surface mounted components are inserted on the board.

The automated line is usually followed by the insertion of odd-shaped component, which is still done, at least partially, manually, because the automated insertion of the most complicated components is hard (or too expensive) to accomplish.[55] In some cases companies have even abandoned automatic insertion and returned back to manual work because of the increased flexibility.[92] In *manual insertion*, the board is set up on a tray, and the operator, following the assembly instructions, obtains components from labeled bins and inserts them manually onto the board. In semi-automatic insertion, the board is set up on a semi-automated component insertion machine, which consecutively opens and moves to near the operator the

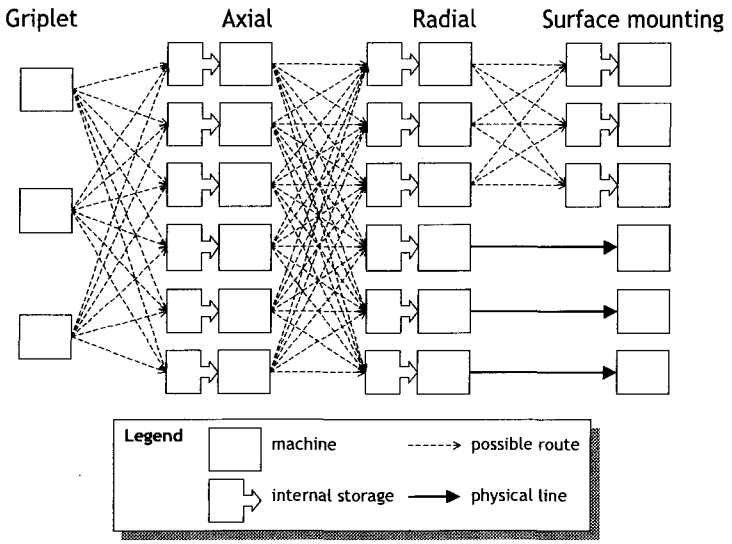


Fig. 6. A typical PCB assembly facility including four production phases. The internal storages buffer the products before they are transported to the machine. The dashed lines indicate possible routes between machines. A physical line is usually a conveyor belt which couples two or more machines together.

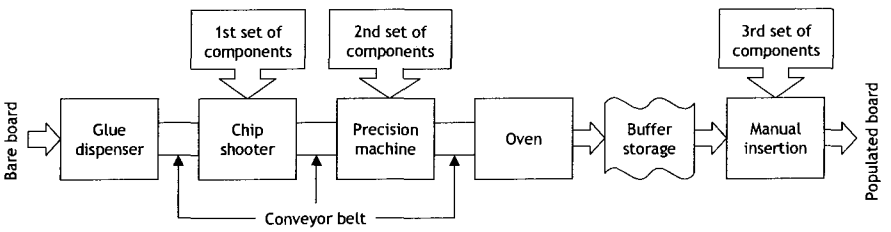


Fig. 7. Workphases in an SMT assembly line.

bin containing the required component, and, at the same time, shines a point of light on the locations of the board where the component is to be inserted.[63,95]

In addition to component insertion, the line may include other phases. Some machines require that the component tape is preprocessed on another machine.[56] The surface mount onsertion is usually preceded by a glue dispenser and succeeded by an oven which hardens the adhesive and fixates the components as shown in Fig. 7[99] (similar lines are also described in Refs. 29, 44 and 117). Typically, the production also includes inspection and testing phases.

The PCBs are transferred from one phase to another either manually (e.g. in magazines which can hold 10–100 PCBs) or with a conveyor belt. The conveyor-linked machines are usually referred as being *coupled*, and a system with a batch transfer as *uncoupled*. Because the change of board type causes a setup, boards of the same type are collected in a batch, in which they are operated successively.

The batch size can vary considerably. A mass-product PCB may remain in active production for several months, whereas a prototype batch usually comprises only few PCBs.

3. Literature Review

Johnsson[54] classifies the literature on PCB assembly according to the number of different PCB types and machines present in the problem. Accordingly, the four main problem classes are:

(1) **One PCB type and one machine** (1–1) class comprises *single machine optimization* problems, where the goal is to minimize the printing time of the machine. The class can be further divided into four subclasses[19,26,40]:
   (a) Feeder arrangement problems (1–1a).
   (b) Placement sequencing (or insertion order) problems (1–1b).
   (c) Nozzle assignment problems (1–1c).
   (d) Component retrieval problems (1–1d).

(2) **Multiple PCB types and one machine** (M–1) class comprises *setup strategies for single machine*, where the goal is to minimize the setup time of the machine. The setup strategies can be classified followingly[74]:
   (a) Minimum setup strategy (M–1a).
   (b) Group setup strategy (M–1b).
   (c) Partial setup strategy (M–1c).

(3) **One PCB type and multiple machines** (1–M) class concentrates on *component allocation to similar machines*, where the usual objective is balancing the workload of the machines in the same line (usually by eliminating bottlenecks).[37,87]

(4) **Multiple PCB types and multiple machines** (M–M) class or *scheduling problems* usually concentrate on
   (a) Allocating jobs to lines (M–Ma) which includes routing, lot sizing and workload balancing between lines.
   (b) Line sequencing (M–Mb).

Other classification schemes have been presented by many authors. Crama *et al.*[26] associate a production plan with the following issues:

(1) Partition the set of board types into *families* which are to be assigned to different lines of placement machines.
(2) For each board type, determine a *partition of the set of component locations* on this board (i.e. decide which locations are going to be served by which machine).
(3) for each machine, determine a *feeder rack assignment* (i.e. an assignment of feeders to position in the feeder rack).

(4) for each pair consisting of a machine and a board type, solve a *component placement sequence* (i.e., an order in which components are placed at the locations on the board by the machine).
(5) For each pair consisting of a machine and a board type, make a *component retrieval plan* (i.e. decide for each component from which feeder it is to be retrieved).

Cases 3, 4 and 5 correspond to problem classes (1–1a), (1–1b) and (1–1d), case 1 is a job allocation problem (M–Ma), and case 2 is a component allocation problem (1–M).

Foulds and Hamacher[40] list the problems associated to PCB assembly:

(1) The allocation of component types to machines.
(2) The allocation of component types to feeder location at each machine.
(3) The pick and place sequence.

Here, case 1 corresponds to class (1–M), and cases 2 and 3 classes (1–1a) and (1–1b).

Feldman and Feuerstein[37] present a hierarchical four-level optimization strategy to reduce changeover, setup and processing times based on the hierarchy of the assembly task:

(1) *Job allocation* and *job clustering* for each assembly line.
(2) *Line balancing.*
(3) Optimization of the *feeder configuration.*
(4) Optimization of the *insertion sequence.*

In this scheme, cases 3 and 4 correspond to problem classes (1–1a) and (1–1b) respectively, case 2 is equal to problem class (1–M), and case 1 is equal to problem class (M–M).

McGinnis *et al.*[87] recognize three levels of decisions:

(1) Selection of machine groups and part families and assignment of families to groups.
(2) Allocation of components to machines when a group has more than one machine.
(3) Arrangement of component feeders and sequencing of placement operations for each machine and PCB.

Here the levels correspond to problem classes (M–M), (1–M) and (1–1ab), respectively.

In this work we adopt the classification of setup management strategies (M–1) proposed by Leon and Peters[74]—with the exception that in our scheme the unique setup strategy corresponds to single machine optimization (1–1):

(a) *Unique setups* consider one board at a time and specify the component-feeder assignment and the placement sequence such that the placement time is

minimized. This is a common strategy when dealing with a single product and a single machine in a high-volume production environment.

(b) *Group setups* form families of similar parts such that setups are incurred only between families.

(c) *Minimum setup* strategy attempts to sequence boards and determine component–feeder assignments to minimize the total changeover time.

(d) *Partial setups* are characterized by partial removal of components from the machine when changing over from one product type to the next.

Another classification of setup strategies is given by McGinnis *et al.*[87] and Ammons *et al.*[8]:

(1) Single setup strategy: Configure a group of machines to produce a family of board types using a single setup which is sufficient for the entire family. There are two possible options for achieving this:

   (i) *Unique setup strategy*: A single setup strategy applies to a family which contains only one product type (i.e. mass production).

   (ii) *Family setup strategy*: Several product types are in the same family. There is usually a tradeoff which must be explicitly examined; the family setup eliminates the need for machine setups between board types at the price of potentially increasing the assembly time for each individual board type in the family.

(2) Multisetup strategy: Because a limited component staging capacity on the placement machines prohibits the use of the single setup strategy, some additional setups must be performed within a family. In this scenario, the objectives generally include the minimization of the production time consumed by setups along with the minimization of WIP levels. There are two possible multi-setup strategies:

   (a) *Decompose and sequence* (DAS): Break the family into smaller sets of PCB types (possibly of size one), then sequence the subsets so as to minimize the incremental setups between the subsets. This strategy requires some additional WIP at the risk of potentially having to remove a component type to process one subset of PCBs and then having to restage it for a later subset.

   (b) *Partition and repeat* (PAR): Partition the required components into subset such that the machine group has enough staging capacity for each subset. Populate PCBs with components by configuring the machines with each subset of component types in turn. This approach requires the accumulation of a batch of partially populated PCBs but it includes relatively few setups.

In this scheme, the unique, family and DAS setup strategies correspond classes (1–1), (M–1b) and (M–1a), respectively. The PAR setup strategy (see also Refs. 31 and 32) is a special case of (M–1b), where the boards in the group are loaded

several times to the same machine but with different component setup. Maimon and Shtub[82] classify four problem types according to how many times (one or many) each component type and job (PCB) is loaded: In type 1 each PCB and each component is loaded only once, type 2 constraints each component to be loaded only once, type 3 constraints each PCB to be loaded only once, and in type 4 there are no limitations.

In the remainder of this section, we try to give a condensed presentation of the research done so far. We review the literature according to Johnsson's classification[54]: single machine optimization in Sec. 3.1, setup strategies for a single machine in Sec. 3.2, component allocation to similar machines in Sec. 3.3, and miscellaneous scheduling problems in Sec. 3.4.

3.1. *Single machine optimization (1–1)*

Although we can recognize four distinct subproblems (feeder arrangement, placement sequencing, nozzle assignment and component retrieval) in single machine optimization, they are strongly intertwined and, therefore, usually not solved altogether independently. For example, an optimal placement sequence does not guarantee optimal printing time if the feeder assignment is neglected (and vice versa). Another aspect of the problem is the wide variety of different machine environments considered in the literature. Most of the work have been based on (variants of) pick-and-place machines, but lately the trend has shifted towards rotary turret machines as they have become more popular in industry. Also, manual and semi-automated operations are considered by some authors.

Ball and Magazine[12] study the problem of determining the component insertion sequence and feeder arrangement for a *pick-and-place machine*. They regard the former problem as a special *traveling salesman problem* (TSP). The authors model TSP as a stacker crane or rural postman problem, and present a heuristic algorithm for solving it. The metrics used in the formulation affects the solution. If the placement head moves only in one orthogonal direction at a time, the Manhattan metric is used for measuring the length of the movements and one can find an optimal solution in polynomial time. If the head can move in the x- and y-directions simultaneously, Chebyshev metric can be applied for the distance calculations but the solutions are then suboptimal. However, the authors prove that in the latter case the maximum error of their solution is bounded. The feeder arrangement problem can be structured as the classical assignment problem when considered separately of the insertion sequencing. Although there are efficient algorithms for solving the problem, in this case the cost of assigning a component to a feeder location cannot be expressed as a simple cost coefficient, and, therefore, they cannot obtain optimal solutions.

Leipälä and Nevalainen[72] recognize the insertion sequence and feeder assignment problems in the single machine problem, and solve them separately. The optimal insertion sequence for a fixed feeder setup can be obtained by considering the

problem as a three-dimensional asymmetric traveling salesman problem. The optimal assignment of the components to the feeders for a fixed insertion sequence can be formulated as a quadratic assignment problem. The overall problem can then be solved heuristically, which brings suboptimal but, in practice, satisfactory solutions.

Chang and Young[23] introduce a hypothetical machine design for simultaneous mounting. The machine comprises one or more robots or a robotized pick-and-place head, a component supply device (movable or fixed), and an assembly holding device (movable or fixed). To optimize the printing, the algorithm should determine stop positions for a PCB, the components printed at each stop position and the placement of components in the delivery head. The authors present a mixed-integer programming model of the problem and a branch-and-bound based greedy heuristic algorithm for solving it.

Leu and Ji[75] recognize three basic machine types—insertion, pick-and-place, and rotary turret—and discuss methods for solving the component insertion sequence in each of these. In an insertion machine, where a separate machine (a sequencer) produces an input tape for components, the problem is considered to be a TSP. In a pick-and-place machine with a moving head, a fixed table and fixed feeders, the problem can be model as a rural postman, linear assignment or quadratic assignment problem. In a rotary turret machine, the authors model the problem as a TSP and use a *genetic algorithm* (GA) for solving it.

Supinski *et al.*[105] discuss the planning of the printing order and the generation of the robot code on a single machine. These tasks comprise three stages: CAD data importation, PCB assembly planning, and code generation. In the second stage, the authors consider three separate workphases. In adhesive deposition the authors model the problem as a TSP and use the arc-opt or 3-opt method for solving it. In solder deposition, a two-step procedure first clusters the solder pads and then applies a TSP formulation to each individual cluster. In component insertion, the objective is to minimize the number of components to be manually inserted on the board.

Zhou and Leu[119] describe a *Petri net model* for solving feeder assignment, printing order and tool assignment on a single machine. The authors present an ordinary (or non-timed) Petri net model for analyzing the system behavior (e.g. deadlock-freeness, buffer boundedness, reversibility and conflict-freeness), and a temporal Petri net model for evaluating the system performance (e.g. productivity and machine utilization).

Foulds and Hamacher[40] present methods for identifying optimal bin locations, which surround the worktable on all four sides, and for determining a component insertion sequence for the board. The authors model the former problem as a single-facility location problem, and the latter problem as a TSP.

Zijm and van Harten[120] discuss a *hierarchical decomposition* for a modular component placement system, which comprises several identical successive insertion machines. To minimize the cycle time of a batch of PCBs, the following problems

must be solved. Firstly, determine the number of feeders for each component type, and if more than one feeder contains the same component, specify the board locations which are delivered from each feeder (solved with a greedy heuristic). Secondly, allocate feeders to the insertion machines (solved with a local search heuristic). Thirdly, assign the feeders to the feeder positions of a machine (solved with the standard Hungarian method). Finally, specify a placement sequence for a given feeder assignment (solved with the 2-opt heuristic). If a solution on a higher level leads to infeasible or inferior solutions on a lower level, a feedback loop allows the program to change the higher-level solution, thus avoiding the problem.

Wang[112] compares three layout design methods for a dynamic pick-and-place machine with a mobile worktable and magazine, and a single placement head. The layouts include one-magazine-and-one-board (1M1B), one-magazine-and-two-boards (1M2B), and two-magazines-and-one-board (2M1B). Wang *et al.*[113] solve the placement sequence and feeder arrangement for a machine where the board and feeder carriages move on x-direction and the head on y-direction. First, the insertions are sorted according to their x-coordinate on the board, and the sorted list determines the placement sequence. By using this sequence, a matrix of the component exchange frequency (which describes how often a component pair is subsequent is the sequence) is built, and the problem is then modeled as a TSP and solved accordingly.

Sanchez and Priest[95] present a method based on artificial intelligence and an expert system with sequencing decision rules for *semi-automatic assembly.* The approach divides the task into four phases. Firstly, design criteria which simplify operators insertion routines are chosen. Secondly, the insertion rules are prioritized (e.g. insert small components before large ones, insert all component of the same type successively, insert components in a sequence which minimizes the movements of the machine bed). Thirdly, the components are grouped into insertion sequence classes. Lastly, the board movements are modeled as a TSP and solved with the nearest neighbor heuristic. Khoo and Ng[63] discuss a similar problem formulation but apply a genetic algorithm for solving the placement sequence. They compare the results to Sanchez and Priest and observe an improvement in the total distance travelled by the machine bed.

Johnsson *et al.*[55] discuss designing an efficient and easy-to-learn sequencing for *manual installation of components.* Favorable sequences can be characterized by several nonstrict rules (e.g. use both hands equally in the insertion, proceed from top to bottom and from the edges to the center, and take the closest next part). The authors present three different sequencing methods: line sweeping, traveling salesman clustering, and weighted 3-matching.

Sadiq *et al.*[94] present a method for arranging feeders in a SMT machine in a low-volume, high-mix environment. First, the slots are assigned by using a heuristic rule, and, after that, the slot rearrangement process tries to locate the components so that they are adjacent in the insertion sequence. The goal is to assign the components

inside a contiguous group to make the components required by a PCB adjacent, and thus save feeder carriage movements.

J. Ahmadi and co-workers have studied *dual delivery pick-and-place machine* in several papers. J. Ahmadi *et al.*[2] present an emulator for studying the effect of different printing process parameters. The overall system is modelled by subproblems. Component allocation problem tries to maximize the number of board completions attainable by a single allocation of components. Partitioning problem minimizes the idle time caused by imbalance in the use of the manipulators and nozzle changes. The authors give a detailed discussion of these problems,[3] where they present mathematical formulations and solve them with a mixed integer programming package. R. Ahmadi and Kouvelis[4] concentrate on the same machine and present methods for solving single product staging problem (SPSP) and multiproduct staging problem (MPSP). SPSP involves the assignment of the required component feeders for the specific board type to the two carriers, while simultaneously assigning the required tools (for dispensing each component type) to the tool magazines by observing capacity constraints. The objective is to minimize the total time to assembly the PCB. The authors give an integer programming formulation as well as present a Lagrangian relaxation based branch-and-bound algorithm for SPSP. In MPSP, the objective is to minimize the total time to assemble a given set of multiple PCB types. Finally, J. Ahmadi *et al.*[1] study the feeder assignment problem by formulating it as a reel positioning problem (RPP), where the goal is to minimize the direction changes and the sum of movements of the feeder carrier. In addition to the mathematical formulation, the authors give a heuristic algorithm based on solving the shortest path in a layered network. A two-manipulator machine is also considered by Hernandez and Leon,[46] who discuss interference avoidance of the manipulators working on the same area.

Crama *et al.*[27] and van Laarhoven and Zijm[111] present hierarchical decomposition schemes for a line of *3-headed pick-and-place machines*. In both papers the goal is to minimize the processing time of the bottleneck machine. Crama *et al.* identify six subproblems: (1) determining how many components each machine must mount and with what equipment; (2) assigning feeder types to machines; (3) determining what components each head must mount; (4) clustering the locations into subsets of three; (5) determining the sequence of pick-and-place operations; and (6) assigning feeders. These subproblems are then solved by using simple heuristic algorithms. Correspondingly, van Laarhoven's and Zijm's problem hierarchy has five steps—determining equipment for the heads on each machine, assigning components to the machines, assigning feeders, clustering components for pick-and-place operations, and sequencing the cluster and component within each cluster—which are mainly solved with simulated annealing.

Optimizing feeder arrangement and insertion sequence of a *rotary turret machine* is discussed by several authors. Bard *et al.*[13] model the placement sequence of such a machine as a TSP and solve it with nearest neighbor heuristic. Moreover, they

formulate the feeder assignment and component retrieval problems as a quadratic integer program and solve it with Lagrangian relaxation. Yeo *et al.*[117] present a rule-based system, which employs simple heuristics. Feeders are assigned according to a one-pitch incremental heuristic. The placement sequence is modeled as a TSP and solved with the nearest neighbor method. Crama *et al.*[26] use hierarchical decomposition, and solve the feeder assignment, the component placement sequence and the component retrieval plan with local search heuristics. The overall objective is to minimize the sum of makespans on the bottleneck machine on a single production line. Altinkemer *et al.*[7] present an integrated model and heuristic for assigning the feeders and sequencing the placement operations. First, the delivery problem is solved for each component type at every possible feeder location. A feasible solution can be used as the cost of assigning the component type to the feeder location in question. The assignment problem can then be solved by using these costs.

Crama *et al.*[25] discuss the *component retrieval problem* (CRP) in detail. In CRP, the placement sequence of components on the board, and the assignment of component types to (possibly multiple) feeder slots are given. The problem is then to decide from which feeder slot each component should retrieved. Naturally, this problem emerges only if at least one component type is duplicated in the feeders. The authors describe a PERT/CPM network model of CRP, and present a polynomial algorithm for solving the problem.

Educational and research topics have been considered by Bodner and co-workers and Feldman and co-workers. Bodner *et al.*[18,19] present a virtual machine model, in which three mechanisms—component feeder, board locator, and placement mechanism—interact. The authors describe two approaches of studying process planning and equipment configuration: *prescriptive models* which emphasize the determination of a near-optimal solution to the given problem, and *descriptive models* which emphasize the performance evaluation of a specified set of solutions to the problem. In this view, a virtual prototype is a descriptive model that allows one to assess performance of a machine or system off-line. Feldman *et al.*[37,38] describe a similar approach for modeling the whole production line. It allows for the demonstration of the production starting from planning and ending to the diagnosis of simple assembly processes.

3.2. *Setup strategy for single machine (M–1)*

Coble and Bohn[24] recognize two approaches to reduce setup times: (1) reduce the time to set up a feeder; and (2) reduce the number of feeders to be set up. The authors argue that most of the research have concentrated on the latter approach and ignored the former. Consequently, the authors present a two-part approach for reducing the time to set up a feeder. First, processes are re-engineered using SMED (single minute exchange of dies) concepts (which were originally developed for metal fabrication). After that, a factory information system with wireless computers and barcode scanners are used both to reduce the setup time and to increase the setup

accuracy. There are two kinds of setup operations: offline setup, which can be done while the machine working; and online setup, which can be done only when the machine is shut down. The SMED approach begins with recognizing internal and external setup tasks and assigning the external tasks to be done offline. Next, as many internal operations as possible are converted to external ones to further reduce online setup. Finally, both internal and external operations are streamlined. The authors also discuss other procedures including hot swapping (fill one feeder carrier while the other is being used), a barcode system for feeders, a computer system for locating parts, adding more operator to the lines, and operator training.

In the remainder of this subsection, we concentrate on the second approach (i.e. reducing the number of feeders to be set up).

3.2.1. *Minimum setup strategy (M–1a)*

Minimum setup strategy attempts to sequence the PCBs and determine feeder assignments to minimize the total component setup time. The idea is to perform only the feeder changes required to assemble the next PCB batch. Here we aim to avoid additional feeder changes or reorganization, which would reduce the placement time. In general, similar product types are produced in sequence so that little changeover time incurs.

Lofgren and McGinnis[78] point out two key decisions: We must solve in what sequence the PCB batches are to be processed by the machine, and what component types should be staged on the machine for each PCB type. The authors present a sequencing algorithm (SEQ) based on labeling concept, in which the PCBs are first sequenced and after that a setup is determined for each PCB. The heuristic determines myopically the "best" component types to remove and to add whenever the existing setup is not sufficient.

Barnea and Sipper[14] consider a case of one machine and recognize two subproblems: sequence and mix. They present a mathematical model of the problem and use a heuristic approach based on the *keep tool needed soonest* (KTNS) policy introduced by Tang and Denardo.[106] In each iteration, the algorithm generates a new partial job sequence by using a sequencing algorithm—which decides the next job to be added in the sequence—and mix algorithm, which updates the component mix with KTNS.

Jain *et al.*[52] present a four-stage method for optimizing the setup. Firstly, a greedy heuristic maximizing the component similarity of the jobs is used to determine an initial processing sequence. Secondly, the components are assigned to feeders according to KTNS policy. Thirdly, the jobs are rearranged in the sequence by applying 2-opt heuristic and KTNS. Finally, because the production is a continuous process, at the end of the sequence the frequently used components are preserved for the next production period (i.e. the approach gives heed to the rolling horizon framework).

Günther *et al.*[44] present a typical SMT production line and apply the minimum setup strategy approach, in which the PCBs are sequenced so that each subsequent PCB has the maximum number of components common with its predecessor. The authors discern three different subproblems—job sequencing, component setup and feeder assignment—and solve each of them with heuristic algorithms.

Narendran and co-workers have studied a heuristic approach which starts from an initial setup (a seed) and sequences the PCBs by looking for the similarities between the current setup and the PCBs remaining to be sequenced. Rajkumar and Narendran[93] form the sequence by considering the overall component requirement of the PCB and the number of extra components required in the setup. Kumar and Narendran[68] add a third constraint—slack time—and observe better result in comparison to dispatching rules (e.g. earliest duedate, shortest processing time and least remaining slack) used normally for single machine scheduling with duedates.

Dillon *et al.*[33] discuss minimizing the setup time by sequencing PCBs on a surface mount technology production line. The authors present four variants of a greedy heuristic which aim at maximizing iteratively the component commonality whenever the PCB type changes. This is realized by using a component communality matrix from which board pairs with a high number of common components can be identified.

3.2.2. *Group setup strategy (M–1b)*

In the group setup strategy the feeder assignment is determined for a group or a family of similar PCBs. Any board in this group can be produced without changing the component setup, which is only required when switching from one group to another. Because the placement time for a specific board is in general larger than in unique setup strategy, some efficiency can be potentially lost. However, this is compensated by less frequent setup operations, which compensates the losses in machine speed especially in high-mix, low-volume production. There are variations of the group setup strategy, where a certain set of common or standard components are left on the machine, while the rest of the components (residual or custom) are added or removed as required for a particular board.

Carmon *et al.*[22] describe a group setup (GSU) method for a high-mix low-volume production environment. The products are divided into groups, each of which is produced in two stages: set up common components and insert them to the PCBs of the whole group, and set up the residual components and insert them on each PCB separately. The same authors[83] compare GSU to sequence dependent scheduling (SDS) on three performance measures—line throughput, average WIP level and implementation complexity—and conclude that in general SDS performs better on the last two areas. Practical analysis of these two methods also appear in Refs. 29 and 69.

Maimon and Shtub[82] present a mixed-integer programming formulation and a heuristic method for grouping a set of PCBs to minimize the total setup time.

A user-defined parameter indicates whether multiple loading of PCBs and components is allowed (cf. partition and repeat strategy[87]). This approach is developed further by Daskin *et al.*[31] Their goal is to minimize the total component and PCB loading costs subject to a capacity constraint. The authors present a mathematical formulation for the PCB-grouping problem, show that the problem is $\mathcal{NP}$-complete, and give a branch-and-bound based heuristic algorithm for solving it.

Shtub and Maimon[98] establish that grouping PCBs is an extension of the set-covering problem and present a general heuristic approach based on cluster analysis and similarity measures (e.g. Jaccard's similarity coefficient) which are traditionally found in the literature concerning group technology. Here the goal is to minimize the total production time, but since insertion times are assumed to be constant, the objective reduces to minimizing the total setup time of the groups.

Hashiba and Chang[45] study a single machine case when the objective is to minimize the number of setups. They decompose the setup problem into three subproblems—grouping PCB types, sequencing the groups, and assigning components for jobs—and apply heuristic algorithms to each of them individually. Furthermore, the authors experiment with a simulated annealing method and observe that it gives better solutions than the heuristic decomposition approach.

Luzzatto and Perona[81] present a heuristic method for grouping PCBs to minimize the setup size. Although the authors consider a production line consisting of several workphases, their model enables the solution for each workphase to be obtained separately from the others.

Bhaskar and Narendran[17] apply graph theory for grouping PCBs. The PCBs are modeled as nodes and their similarities as weighted arcs between the nodes. After that, a maximum spanning tree is constructed for identifying the PCB groups.

Xu *et al.*[116] form PCB groups and divide the feeder slots into three "feeder bays": fixed, semi-fixed, and configurable (cf. Refs. 8 and 99). The fixed feeder bay comprises the most frequently used components and it remains constant throughout the production, whereas the semi-fixed feeder bay is changed whenever the group changes and the configurable feeder bay whenever the board type changes.

Smed *et al.*[99] give an integer programming formulation of the *job grouping problem* (cf. Ref. 28), and compare several heuristic algorithms based on greedy, clustering and repair-based local search methods. Johtela *et al.*[62] expand the problem to account multiple and possibly conflicting grouping criteria—such as different substrate widths, adhesive types, and production priorities—by modeling them as fuzzy sets.

Ohno *et al.*[89] group PCBs to minimize the sum of the assembly and setup times. The authors present a multi-type PCB assembly (MPCBA) optimization problem, which is divided into three subproblems: insertion sequence problem (ISP), reel position problem (RPP), and optimal assembly mode problem (OAMP). ISP is modeled as a TSP for fixed reel positions (i.e. feeder assignment), and solved with a 2-opt heuristic. RPP is viewed as an assignment problem, where the cost is the sum of weighted tour costs of the TSPs for the group, and it is solved with an evolution

strategy. Evolution strategy is also applied to OAMP, which is modeled as a set partitioning problem with TSP type constraints.

In addition to placement machines, the group setup strategy has been proposed for *sequencer problems*. Fathi and Taheri[36] present a heuristic algorithm to group products for minimizing the setup of a sequencer for an axial placement machine. Sule[104] applies group technology and develops a heuristic method to minimize the changeover cost and to balance the workload between sequencers.

3.2.3. *Partial setup strategy (M–1c)*

Partial setup strategy specifies that only a subset of the feeders on a machine are changed when switching from one product to the next. Because the goal is to minimize makespan, the partial setup strategy resides between the unique setup strategy (where only the placement time for each individual PCB is minimized) and the minimum setup strategy (where only the changeover time of each PCB is minimized).

Leon and Peters[73] present a heuristic for composing partial setup and compare its solutions to the corresponding unique, minimum and group setups. Obviously, unique setup dominates when batch sizes are large (such as in mass production). The group setup strategy dominates the minimum setup strategy, because it considers all the PCBs. The partial setup strategy performs well under all scenarios. The same authors[74] confer similar results from a broader set of experiments. Peters and Subramanian[90] analyze four partial setup strategies—unique setup, sequence dependent setup, tradeoff dependent setup, and minimum setup—and conclude that no single fixed strategy dominates in all scenarios.

3.3. *Component allocation to similar machines (1–M)*

Only few papers considering the case of similar (sequential) machines in the same production line have been put forth. Here, the most eminent criterion is workload balancing so that the bottlenecks of the line are eliminated.

Lofgren and McGinnis[79] present a soft configuration decision, which has an impact on two key criteria: workload on each machine (i.e. balancing), and material handling (i.e. machine visits). The soft configuration problem specifies the attributes which are available on each machine and, therefore, it determines the set of operations which could be performed. The authors consider three operating policies: *dynamic* (where tools are added or removed so that the required operation can be done in one machine without routing it to another), *static* (which specifies configuration for a finite production horizon and routes jobs to appropriate machines) and *pseudo-dynamic* (where some attributes are static and the rest dynamic). The authors give heuristic algorithms for static and dynamic operating policy, where the objective is to maximize machine utilization and minimize material flow transactions.

Ben-Arieh and Dror[15] consider assigning components in a case of two insertion machines, so that all boards in a production plan can be produced, and the output rate is maximized. They give a mathematical formulation of the problem and solve it with a heuristic algorithm.

Askin *et al.*[10] discuss a surface mount technology plant with multiple identical machines. They present a four-stage approach for grouping the boards and allocating the components to the machines. First, the boards are grouped into production families. Next, for each family, the component types are allocated to the machines. After that, the families are divided into board groups with similar processing times. Finally, the groups are scheduled. The objective is to minimize the makespan for assembling a batch of boards and to reduce the mean flowtime. The authors present and compare three heuristic methods—component-assignment/workload balancing algorithm (CAWB), workload balancing algorithm with shortest total processing time (WBASPT), and natural board subfamily algorithm (NBSA)—and conclude that CAWB and WBASPT outperform NBSA.

Watkins and Cochran[114] consider a line of similar insertion machines, and propose a heuristic method for rebalancing the workload by moving components from the bottleneck machine to other machines. However, each move is associated with a cost, and the method finishes when the cost of a move outweighs savings.

McGinnis *et al.*[87] give a mathematical model for component allocation for both coupled and uncoupled machines. Ammons *et al.*[8] continue the work by considering component allocation to two or more placement machines, when the objective is to balance a combination of the assembly time and the machine setup time. When machines are coupled, the workload balancing reduces to maximizing the throughput of the bottleneck machine on line. The authors approach the component allocation problem by developing two heuristic methods based on list processing and branch-and-bound technique. Furthermore, they give an integer programming formulation, which tries to minimize the maximum combined PCB assembly and machine setup time for each PCB over all machines.

Brandeau and co-workers consider assigning components to machines in an assembly plant with *different types of workphases* (automatic, semi-automatic and manual). The goal is to minimize the total setup and processing cost for assembling all boards. Brandeau and Billington[21] present two heuristic algorithms: *stingy component* (which tries to avoid assigning less frequently used components to the automatic workphase) and *greedy board* (which tries to assign a whole board to a single workphase instead of splitting it). After a set of tests, the authors conclude greedy board to be a better method of the two, because the setup cost are high relative to the insertion costs. Hillier and Brandeau[48] extend the same problem by presenting a new mathematical model and an improved heuristic based on branch-and-bound technique. The same authors[47] further expand the mathematical model by introducing a workload balancing criterion and introducing CMWB (cost-minimizing, workload balancing) heuristic.

3.4. *Scheduling (M–M)*

Production can be analyzed on different levels: on the *long-term level*, the analysis uses general presumptions and a simplified mathematical model, whereas a *short-term analysis*, as a rule, is based on simulation. On the highest level of production planning, the problems are general, and they cannot be solved without the support and feedback from the lower levels of planning.

Gershwin *et al.*[41] present a decomposition approach for determining dispatch dates for an FMS, in which the machines are unreliable to satisfy the production requirements. The authors consider whether the parts should be dispatched to satisfy weekly production requirements. The problem is solved in two stages. First, the instantaneous production rates are solved by considering a high level continuous dynamic programming problem. After that, a combinatorial algorithm determines the dispatch times at the lower level.

Johri[58] considers scheduling from a practical point of view, and presents a technique for sequencing PCB batches. The presented method begins with a realistic scenario, in which the production has been active for some time, and new jobs are inserted to existing production sequence. At each insertion, the goal is to minimize the imbalance of the machine workloads and to ensure that the duedates are not violated. In addition, the number of setups is minimized, if possible. The method works as follows. First, it determines desirable production rate for each workstation. After that, the inserted jobs are grouped according to their duedates. Next, the method determines the group with the least slack time, and calculates penalties for each job in that group. Finally, the job with the minimum penalty (other heuristic rules are also possible) is inserted into the sequence. When the situation is updated, iteration continues until all jobs are sequenced.

Johri[59] discusses the design of a production line with respect to the capacity constraints and flowtime minimization. The intention is to determine the amount of different machine types, buffer sizes, lot sizes and the loading sequence, when the products, processing times on each machine type and the machine downtimes are given. The number of machines required in a workphase are obtained from a simple equation, as well as the average flowtime and capacity of the line. The average in-process inventory can be acquired by using Little's law (i.e. it equals the average flowtime multiplied by the production rate), and by minimizing this value the desired flowtime can be attained.

Lofgren *et al.*[80] study the *station routing problem* (SRP), where the sequence of the component assembly operations is determined so that it minimizes the number of workstation visits. The authors use a graph theoretic approach, in which a precedence graph is partitioned according to workstations and used to determine a workstation sequence which has the minimum cardinality. SRP is shown to be $\mathcal{NP}$-hard, and therefore the authors test and analyze several different heuristics for the problem.

Klegka and Driels[34,66] put forth a simplified model for measuring costs in long-term production planning. It yields intuitive results when used for comparing different production line layouts. The authors observe that the most economical assembly system is obtained when enough manual assembly personnel are provided to reduce the maximum station time to that of the slowest assembly machine. In batch production, even small lot sizes are economically justified, since the effect of frequent changes is less significant than might be expected.

Ben-Arieh and Maimon[16] consider a case of two different sequential machines which use the same sequence (i.e. a permutation schedule). The objective is to minimize the mean flowtime, and the authors solve the problem by using simulated annealing approach.

R. Ahmadi and Wurgaft[6] consider a mid-variety, mid-volume production environment with synchronized flow manufacturing, where the materials move smoothly and continuously from one operation to the next. The objective is to maximize the throughput rate. Each time there is a major change in the product mix and demand, there are three problems to be solved: (1) how many stations should be created; (2) how many machines each station should have (i.e. machine allocation problem); and (3) which operations are performed by which station (i.e. staging problem). To balance the total workload the maximum workload of a station should be minimized. The authors give a mathematical formulation of the problem as a quadratic integer program.

Dagnino[30] presents a system which combines product design, assembly planning, line balancing and the generation of the shop floor drawings. The main problem is general process planning, which involves deciding how to use the capabilities of the shop floor to manufacture a given product. In PCB assembly, the author recognizes the following stages: (1) identifying the components to be printed on a PCB; (2) determining the required assembly operations; (3) assigning resources to the assembly operations; (4) sequencing the assembly operations; (5) line balancing; and (6) developing shop aids. The major issue is the hierarchy of these decisions and dependencies between the stages. When a decision is made on a higher level, the lower-level characteristics are generalized. However, at some point, the technical details cannot be omitted, and the system must be able, for example, to calculate accurate operation times for each individual machine.

Khoshnevis *et al.*[64] also recognize the potential of a general scheduling system, and present an integrated system for assembly planning and schedule generation. In assembly planning, the system prepares a detailed sequence of operations to transform a set of disjoint components into a final product. In scheduling, the system assigns manufacturing resources to the operations indicated in the assembly plan in such a way that some relevant criteria (e.g. duedates) are met. The main idea of the solution algorithm is to assign jobs to machines on the basis of the availability of machines (or of a quantifying factor which is calculated with heuristic rules) and the requirements of the unfinished jobs.

Klincewicz and Rajan[67] consider the component grouping problem in an environment of multiple board types and multiple workcells. The problem is to decide how should the components be assigned (i.e. partitioned) among the workcells so that the number of machine visits is minimized and the workload is balanced. The authors give an integer programming formulation, where minimizing the machine visits is modeled as an objective function and the load balancing as a constraint (i.e. each workcell has a maximum allowable deviation from the average insertion volume). GRASP (greedy random adaptive search procedure) method (i.e. a local search replicated many times with different, randomly chosen starting points) is used to solve the problem. The authors present two variants of the initial starting points: Component-based variant assigns a pair of components to the same workcell if a large number of the items to be produced use both components, whereas code-based variant looks at the larger volume board types and attempts to pack the components belonging to a board type on as few new workcells as possible. The initial solutions are then improved by moving components from one workcell to another or by exchanging components between two workcells.

Dessouky *et al.*[32] convert the scheduling problem of flexible assembly lines to a flow line scheduling problem. This approach assumes that each machine processes the product at most once, the setup time depends only on the machine type, processing time variability is negligible, and setup times are small in comparison to batch processing times. The goal is to maximize the throughput while keeping WIP at a minimal level without increasing the resources of the assembly process. The batch sizes and the number of each machine type are given in the problem formulation. The authors present a solution method comprising two phases. In the first phase, machines are grouped into workstations which are visited by each product in the same sequence. Thus, the problem is converted into a flow line configuration problem. Moreover, the task is to determine a permutation schedule, which maintains the same sequence down the line in each workstation. In the second phase, the products are assigned to the first workstation (since the subsequent workstations will have the same sequence) by using Campbell's heuristic.

Feo *et al.*[39] also present an integrated decision support system, INSITES (integrated scheduling and throughput evaluation system), for scheduling, inventory and throughput evaluation. It provides the shop floor managers with the real-time information needed to make critical production planning and control decisions. The user inputs the PCBs, duedates, quantities, WIP, and machine state. After that, the system computes the daily production quantities and generates MIS (management information system) reports. Because planning includes several conflicting objectives (e.g. duedates, workload balancing, throughput maximization and waiting time reduction), the system offers a set of schedules, from which the user can select one for the production. The authors point out that instead of being constant, the load times fluctuate with the job priorities, parts status, processing requirements and customer demands. Variability of the processing capacity defers the work and

delays the batches. The system assigns the lot sizes so that they divide the demand evenly, and schedules the jobs according to the slack values.

Kim *et al.*[65] compare four sequencing algorithms, where the objective is to minimize the sum of tardinesses. NEH algorithm sequences the jobs according to a simple objective function (e.g. in increasing order of duedates or in decreasing of total workload). The ENS (extensive neighborhood search) method begins with a proper initial sequence and tries to improve it with swap operations. RBO (rolling block optimization) algorithm also begins with an initial job sequence. After that, it resequences a block of sequential jobs (i.e. window) trying to improve the local sequence while the rest of the jobs remain intact. The window gradually moves through the sequence until it reaches the end. The fourth studied approach, *tabu search*, is based on local search methods, where getting back to previously found solutions is prevented with a tabulist.

Johnsson *et al.*[57] introduce a *generalized flexible flow line* (GFFL) scheduling problem, where the products visit successive machine banks, and the processing time is a function of the product and the machine type. The authors present a mathematical model of the problem and describe an interactive production scheduler, which can be used for analyzing schedules created either manually or by a heuristic algorithm.

Lin *et al.*[76] present a Bi-Level Scheduling System (BLISS), which integrates both product-level and board-level scheduling on a capacitated multiple flowline and strives to determine economical lot sizes and job sequences. Product level scheduling shifts a complete set of boards for a product so that no WIP inventory is built up. Because this does not necessarily use the production capacity optimally, the schedule is then improved with a board-level scheduling, which minimizes tardiness but may increase WIP. The authors approach the problem by applying general scheduling methods (e.g. Palmer's algorithm, Gupta's algorithm and CDS heuristic). From their experiments, the authors conclude that the two-level approach suits well when duedates are very tight or product demand is highly variable.

Nesbit[88] describes general requirements for constructing a computer system for analyzing assembly processes. The possible benefits of such a system include reduced machine setup times, increased quality of the final product, and better throughput.

R. Ahmadi and Kouvelis[5] present a mathematical framework for designing and configuring the layout of electronics assembly lines. When the machines are tightly coupled due to small buffer space, throughput can be increased and WIP levels reduced by decreasing the setup times and balancing the machine workload. Also, configuration can be based on the assumption that around 80% of the cumulative product demands are known to follow a regular pattern. The authors analyze three design approaches. In the *mini-line* approach, PCBs are divided into families, and each family is produced in a dedicated line. In the *flexible flow line* (FFL), there is only one line and all the components required for the assembly of the various PCBs are concurrently present on the line. Thus, major line setups can be avoided but the

machines are bound to be under-utilized. In the *hybrid line* approach, several PCB families are produced in the same mini-line. The authors formulate the *electronics assembly line design problem* (EDP), from which they define special cases for each of aforementioned approaches. From the computational experiments, the authors conclude that FFL dominates the other design in terms of throughput performance but with increasing setup times, the mini-line provides higher throughput than the others. Moreover, the authors discuss the suitability of the designs to different production environments.

Balakrishnan and Vanderbeck[11] discuss a high-mix, low-volume production plant with several identical assembly lines. Each line includes a chip-shooter which is the bottleneck of the line. The aim is to group products and assign the groups to different lines so that workload is balanced and the setup times are minimized. Because these goals are contradictory, the presented model tries to minimize the setup costs while keeping the workload within prespecified limits. A partial setup (which should not be confused with the partial setup strategy of Leon and Peters[74]) comprises permanent and temporary components (i.e. a standard and custom setup[8]). The authors argue that this partial setup is a compromise between "complete" (i.e. unique) and "incremental" (i.e. minimum) setup, because it reduces setup operations by exploiting component commonalities and is easier to handle than incremental setups. The authors give an IP formulation of the problem and reformulate it as a capacitated set covering (CSC) model. CSC has two subproblems: production selection (PS) decides to which machine a product should be assigned so that setup times are minimized and workload is balanced; setup optimization (SO) decides which components are permanent and which temporary for a group of products on a single machine so that setup costs are minimized. PS is solved with list heuristics, local search and simulated annealing. SO is solved with a combination of heuristics, lower bounds and enumeration.

Lin and Tardif[77] consider multiple board types in a single line. The line is set up once, and there are no setups between the boards. The problem is to partition the component types to the machines in the line so that all the boards can be processed quickly and with a good workload balance. The board demands and machine breakdown occurrences are predicted with a probability distribution. Moreover, it is assumed that there are no precedence constraints on the order of component placements, the placement time is constant for all the components placed by a machine, and a new batch cannot be processed until the current one is completed. Although heuristic methods (e.g. greedy and two-dimensional vector packing heuristic) often provide satisfactory results, they are not applicable here, because they do not consider demand and capacity uncertainty. Therefore, the authors present a stochastic mixed-integer programming formulation for the problem, where the objective is to minimize the makespan for all the PCBs during a production cycle.

Spedding and Sun[103] describe a cost system which uses an activity based costing (ABC) model. Since much of the significant costs in producing an item are not

volume related, ABC should provide a better cost model. The idea is to develop a discrete event simulation (DES) model by observing the actual production times of activities in a plant and then characterizing their statistical variation. Activities are classified into four general categories according to when they are performed: unit level (each time a unit is produced), batch level (each time a batch is produced), product level (needed to support the manufacturing of a product), and facility level (needed to sustain the factory's performance; e.g. rent, depreciation, and insurance). The authors apply ABC modeling in a simple SMT assembly line.

3.4.1. *Simulation-based systems*

Simulation has been a widely used approach for modeling PCB assembly plants. Almost all papers which compare scheduling algorithms use simulation for testing the methods. However, because the focus in these papers is on developing actual optimization algorithms, the description of the simulation model is usually terse, and testing and analyzing settle for a modest set of test cases and bypass statistical tests.

Shevell *et al.*[97] present an early simulation system for analyzing a complex PCB manufacturing system. They also discuss the validation of the system and compare the results with real-world data. Taylor[107] also describes a simulation tool for evaluating flexible control strategies.

Johtela *et al.*[61] describe a simulation tool for creating and analyzing schedules in a GFFL environment. The system simulates the production from an initial situation to a given moment in the future. The input defines the product batches, their allocation and sequence for each machine and the duedates. The output includes summaries of the production period including the lateness of the batches and machine workload charts. The user can reconsider the allocation and sequencing of the batches and repeat the simulation and update operations to find a better balancing of the workload. Häyrinen *et al.*[50] describe several scheduling algorithms for GFFL problem, which are implemented and integrated into the system.

Jackson and Johansson[51] present a discrete event simulator to support decentralized decision making. Production control benefits from that the workers have a forecast of the coming workload in real time. The system is directly linked to the information system of the company and it updates the state of the production process every ten minutes. Each machine has a forecast for six hours, which is shown on the screensaver on every PC in the shop floor. The simulation operates on both strategic level (development of a new production system or improvement of a production system) and operational level (estimates of capacity needed for a product mix, consequences of WIP for a particular product mix, and consequences in delivery dates due to a particular plan).

Estremadoyro *et al.*[35] describe a simulation tool, Electronics Manufacturing Simulator (EMS), for modeling and analyzing assembly line layouts. The system uses simple specifications for forming a processing line, generates alternative designs,

approximates throughput, simulates the operation with different parameters, and produces a set of output summaries for the given layout. The system comprise a line definer and static analyzer, which produces station processing times (used in estimating the maximum throughput of each station) and line capacity.

Smed *et al.*[100] discuss the structure and operation of an interactive scheduling system for a high-mix low-volume SMT assembly line involving multiple criteria (e.g. different adhesive types, conveyor belt configurations, and priorities). The criteria are modeled with fuzzy sets, and the user can set weights for the importance of the criteria. Thus, the poor satisfaction of some criterion can be compensated with other criteria. The user can also manipulate the schedule directly via a graphical user interface as well as employ feeder and printing order optimizers selectively.

4. Commercial Software Systems

There are several software tools for production planning available in the market but relatively few of them are specialized to the PCB assembly. Generic tools are seldom suitable for production planning in PCB manufacturing due to the special requirements associated with it. Firstly, the system must have knowledge about the machinery used in the production environment, its capabilities and limitations. Secondly, a production planning tool should have a tight integration with the existing systems, such as CAD/CAM (*computer aided design/manufacturing*) or MRP (*material resource planning*) systems. Thirdly, in practice the usability of a system reduces drastically, if it does not produce NC-codes for the machines. Since PCB manufacturing facilities are usually highly automatized, the goal of a production planning system is not to impose any further work to the personnel.

In this section we review briefly some of the most popular software systems designed to support PCB assembly. The systems and their key features are listed in Table 1. We concentrate on the optimization features, albeit optimization is rarely the primary task in these systems. The systems are mainly designed for automatizing the PCB design and manufacturing processes, such as joining *bill of materials* (BOM) files and CAD files together and generating NC-codes for machines. We have to point out, however, that our experience with some of these system is limited, and, therefore, we are in some cases forced to resort to information gathered from the vendors' web pages and brochures.[60] The possible errors, omissions and inaccuracies are due to this fact. Moreover, the list of the machine vendors' systems includes only a few entries although almost all machine types in the market have their own proprietary system. However, the current trend is that the machine vendors are starting to co-operate with the general system providers.

In Table 1 the systems are divided into two categories: tools provided by the machine vendors and tools provided by independent companies. A machine vendor's system usually supports only its own equipment and includes a concise interface and some simple code optimization routines. Conversely, since production plants may comprise multiple machines from different vendors, the independent systems aim at

Table 1. A summary of the features of commercial PCB assembly support systems.

	General Information			Supported Machine Types			Integration			Optimization				
	Vendor	Package	Platform(s)	Pick & Place Machines	Rotary Turrent Machines	Thru Hole Machines	Input from CAD	BOM & CAD Joining	NC-code Generation	Printing Order	Feeders	Setup	Line Balancing	Scheduling
Independent Companies	Dareba Corporation	cim-SCAN	DOS, Windows, Unix	•	•	•	•		•	•	•	•		•
	Mitron Corporation	CIMBridge '97	Windows	•	•	•	•	•	•	•	•	•	•	•
	Tecnomatix-Unicam	UniCam	Windows	•	•	•	•	•	•	•			•	
	Software Inc.	Factory Advisor	Windows									•	•	•
	Tecnomatix Technologies Ltd.	EXALINE	Windows, Unix	•	•	•	•	•	•			•	•	
	Unisoft Corporation	EZ-CIM/EZ-CAM	Windows	•	•	•	•	•	•	•			•	
	FABMaster Inc.	FABMaster	Windows	•	•	•	•	•	•	•	•	•	•	•
	AIS Corporation	CircuitCAM	Windows	•	•	•	•	•	•	•				
	Optelco	Optel	Windows	•	•	•				•	•	•	•	•
	Harelda Oy.	SMDOPTIMIZER	Windows	•	•	•	•	•	•	•	•	•	•	•
Machine Vendors	Universal Instruments Corporation	UIC	Windows, OS/2	•	•	•	•		•	•	•			
	Fuji America Corporation	FujiCam	Windows	•	•	•	•	•	•	•			•	
		XCME	Windows							•				
	Panasonic	PanaCIM	Unix	•	•	•	•	•	•	•	•	•	•	
	Factory Automation	PanaPRO	Windows	•	•	•	•	•	•	•	•	•	•	
	Sony	TIMMS	OS/2	•	•	•	•	•	•	•	•	•	•	•

providing better support (e.g. a centralized component library, better CAD support, product transportation from one machine to another) for various machine types and makes. They also support both rotary turret and pick-and-place machine types. Some include support for also through hole machines in addition to the surface mount devices. Most of these systems run on a PC under Microsoft Windows. Some systems also contain networking capabilities that allow them to integrate into a local-area network. These systems can obtain on-line data from the machines and use it to control the production.

The optimization—the word "optimization" is used here loosely, since in some systems the term "optimal solution" actually means a feasible solution—features of the systems vary greatly. Some use a hierarchical decomposition strategy to optimize each level of production ranging from the printing order and feeder arrangement optimization of a single machine to the scheduling of the entire production environment. Others offer only a few means to improve the operation of a machine or balance the component allocation between machines. The features listed in Table 1 are printing order and feeder optimization of the supported machines (1–1), setup strategy optimization (M–1), balancing the load between machines in a line (1–M), and scheduling the jobs, which includes their allocation to the lines (M–M). Since in many cases it is difficult to estimate the quality of the solutions produced by the systems, we have not evaluated the utility of the optimization algorithms used in the systems in greater detail.

5. An Example of a Production Planning System

In the previous section, we gave a summarized view of the currently available applications for production planning in PCB assembly. To elaborate the topic of practical systems, we will next discuss an integrated production planning system (for further details, see Ref. 100). The system is used for solving three distinct problems (see Fig. 8): to group PCBs according their components (i.e. group setup strategy), to assign the components of each group to feeder slots (i.e. feeder optimization), and to minimize the printing time of each individual PCB type on the basis of the feeder setup of group (i.e. printing order optimization).

In the first place, the board data, production demand and duedates are derived from CAD and MRP data. The actual production planning begins with grouping, which can be done algorithmically or manually. After that, the user can sequence the jobs in each group according to his preferences and knowledge of the actual state of the production, which enables him to adapt to sudden changes in the production environment. A screenshot of the system is shown in Fig. 9. *Job Repository* window is used to input and edit jobs, and it also acts as a repository for jobs to be scheduled. The jobs are represented by icons whose appearance indicates the attributes of the jobs. The *Machine Schedule* window visualizes the grouping and allows the user to rearrange the jobs; each column corresponds to a job group in a schedule, and each cell in a row represents a particular job. *Job Inspector* and *Selected*

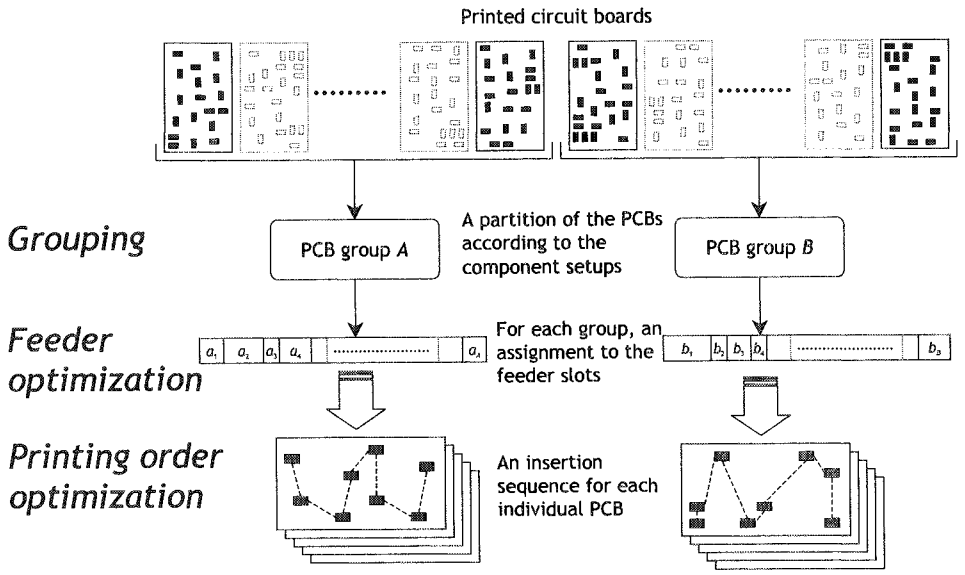


Fig. 8. The problems tackled by the integrated production planning system.

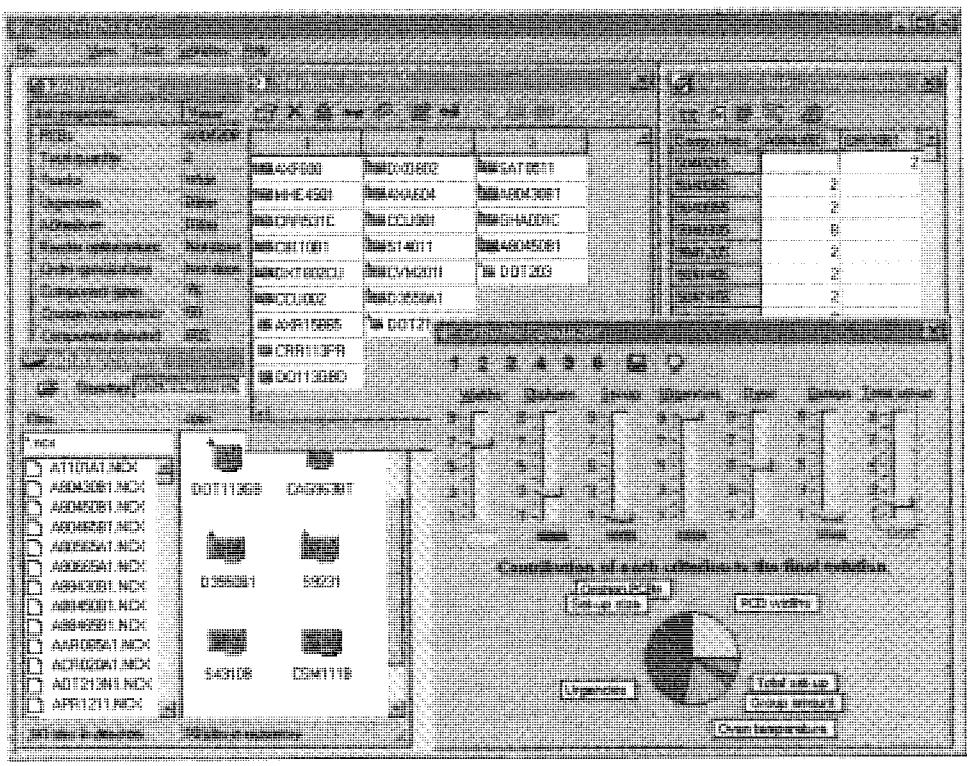


Fig. 9. A screenshot of the production planning system.

Jobs/Components windows give information about the PCBs and their components (e.g. how many feeder slots the jobs require or how long it takes to process the jobs in a machine).

The system is used for planning a typical high-mix low-volume production, where the daily production program includes ten or less different products, and the setup times form a significant part of the total production time. Therefore, the main objective is to minimize the setup times by using the group setup strategy. In addition to the component setup, the system also considers other criteria when forming the groups. Firstly, the widths of the PCBs vary, and the change of the conveyor width causes an interrupt in the printing process. Secondly, some PCBs require component printing on both sides, and in order to avoid unnecessary storaging, the other side should be printed as soon as possible after the first side. Thirdly, the surface mounted components are fixated in an oven which must be heated or cooled if the type of the adhesive used in the board changes. Finally, the PCBs are classified according their urgency, and the PCBs with similar urgencies should reside in the same group. The slider bars in the *Criteria Equalizer* window are used to adjust the relative importances of the criteria (the scale is from one to nine where nine corresponds to the highest importance and one to the lowest). The pie diagram shows how much attention a criterion gets in the final objective function. The multiple criteria optimization algorithm, which is based on a repair-based local search heuristic, is used to improve the current grouping (for further details, see Ref. 62). The algorithm can be stopped at any time and the currently best solution is available to the user.

When the grouping is satisfactory, the user can define the feeder assignments for the groups. The goal is to arrange the components in the feeder slots so that the total printing time of all the PCBs in the group is minimized. Since this is a difficult task, the system uses heuristic algorithms which take into account the closeness of different components on the PCBs, different component handling speeds (e.g. turret, recognition, placement, pickup and table) and component widths. All this information is stored in the component library of the system. The heuristic gives a feeder setup that enables a fast printing of all the PCBs in a group.

Once the feeders have been assigned, the system has to decide the printing order for all PCBs in the group by using an algorithm that is specifically tailored for the machine type in question (i.e. the system uses information about the machine configuration for fine-tuning). The algorithm is based on a local search heuristic which uses discrete event simulation in the cost function. The user can choose from three different optimization algorithms (quick, medium and slow), and the quality of the solution depends on what algorithm is chosen (i.e. the slowest method is presumed to give the best solutions). On average, the printing times provided by the optimization algorithm are more than ten percent better than the times obtained by using the machine vendor's optimization system.

The actual machine codes are converted into a generic format, which enables the system to use any kind of numerical control (NC) code: When the machine type

changes, the converter can be changed or—if a corresponding converter does not yet exist—it can be implemented in a short period of time. Because all processing is done with the generic code format and the machine configuration files are used to fine-tune the optimization algorithms, the system can be used in different machine configurations. Consequently, the system is general and can be easily tailored to suit new machine types.

6. Final Remarks

We have reviewed in this paper theoretical and practical approaches to production planning in PCB assembly. By regarding PCB assembly as a specialization of a general FMS, we gave a methodology for constructing practical production planning systems and discussed the problems associated with it. We described PCB assembly in different levels: the physical fundamentals, the operation of the insertion machines, and the classification of plant layouts. In the literature review, we classified the problems into four categories—single machine optimization, setup strategy, component allocation to similar machines, and scheduling—and gave a brief summary of the research done so far. Finally, we also looked the problem domain from another perspective. We gave a summary of the commercial applications currently available for PCB assembly, and presented one production planning system in more detail.

To conclude, we would like to point out six key topics for the future research on production planning in PCB assembly:

6.1. *Rolling horizon*

Although the production plan is made for a given period of time, the production rarely begins with an empty line, nor does the line remain empty, when the duedate of the last job of the current plan expires. Yet, this rolling horizon framework is scarcely considered in the problem formulations of the PCB assembly literature.

6.2. *Applicability*

Many of the solution procedures overlook the problems associated with the machine operation and the human workers. For example, partial setup strategy may, in some cases, provide the best theoretical solution for a given setup problem, but it may also result in the human operator—who is required to change few feeders whenever the board type changes—becoming prone to make more mistakes than if he performs larger feeder changeovers less often. Likewise, the technical considerations (e.g. machine code generation), in the main, are brushed aside in the literature, and thus the suggested solution procedures may have little applicability in actual production environments.

6.3. *Dynamic production*

Reality rarely follows a plan; there are machine breakdowns, component shortages and maintenance delays, urgent prototype series surpass normal production, and the production plan itself can be subject to sudden alterations during the production period. Therefore, a practical production planning system must be able to cope with this kind of dynamic production and to give new or refined solutions whenever the integrity of the plan is challenged.

6.4. *Multiple criteria*

The bulk of research done in PCB assembly contemplates optimizing one—or rarely a few—criterion (e.g. component setup or duedates). In reality, there are usually several more or less important practical aspects which affect the use of the solution. These aspects either define the space of admissible solutions (e.g. release dates, operation durations, setup times and resource availability) or characterize the quality of scheduling decisions (e.g. duedates, productivity, frequency of tool changes and WIP levels). Some of these multiple criteria must be satisfied for a schedule to be valid, while others may not always be satisfied and might need to be relaxed.

6.5. *User interaction*

We discussed the importance of involving the human production planner in the decision making process earlier in this paper. To summarize, as long as the production planning systems are designed for not completely automated manufacturing processes (such as PCB assembly), the production planner must retain the final word on the production plan to be carried out. This means that the planner must be able to override the algorithmic solutions and effectively take the control if an exceptional situation requires it.

6.6. *Integration*

The lack of cooperation with other systems (such as CAD/CAM and inventory management) is a common reason why a new production planning system can be reluctantly accepted by the shop floor personnel. Production data should be interchanged automatically via a network—it must not depend on routine manual input. Here the key issue is the seamless integration of the production planning system to the other existing systems.

References

1. J. Ahmadi, R. Ahmadi, H. Matsuo and D. Tirupati, Components fixture positioning/sequencing for printed circuit board assembly with concurrent operations, *Operations Research* **43**, 3 (1995) 444–457.

2. J. Ahmadi, S. Grotzinger and D. Johnson, Emulation of concurrency in circuit card assembly machines, Technical Report RC 12161, IBM T. J. Watson Research Center, Yorktown Heights, NY, 1986.
3. J. Ahmadi, S. Grotzinger and D. Johnson, Component allocation and partitioning for a dual delivery placement machine, *Operations Research* **36**, 2 (1988) 176–191.
4. R. H. Ahmadi and P. Kouvelis, Staging problem of a dual delivery pick-and-place machine in printed circuit card assembly, *Operations Research* **42**, 1 (1994) 81–91.
5. R. H. Ahmadi and P. Kouvelis, Design of electronic assembly lines: An analytical framework and its application, *European Journal of Operational Research* **115**, 1 (1999) 113–137.
6. R. H. Ahmadi and H. Wurgaft, Design for synchronized flow manufacturing, *Management Science* **40**, 11 (1994) 1469–1483.
7. K. Altinkemer, B. Kazaz, M. Köksalan and H. Moskowitz, The integrated modeling of printed circuit board manufacturing, *European Journal of Operational Research* (2000) forthcoming.
8. J. C. Ammons, M. Carlyle, L. Cranmer, G. DePuy, K. Ellis, L. F. McGinnis, C. A. Tovey and H. Xu, Component allocation to balance workload in printed circuit card assembly systems, *IIE Transactions* **29**, 4 (1997) 265–275.
9. J. C. Ammons, T. Govindaraj and C. M. Mitchell, Decision models for aiding FMS scheduling and control, *IEEE Transactions on Systems, Man, and Cybernetics* **18**, 5 (1988) 744–756.
10. R. G. Askin, M. Dror and A. J. Vakharia, Printed circuit board family grouping and component allocation for a multimachine, open-shop assembly cell, *Naval Research Logistics* **41** (1994) 587–608.
11. A. Balakrishnan and F. Vanderbeck, A tactical planning model for mixed-model electronics assembly operations, *Operations Research* **47**, 3 (1999) 395–409.
12. M. O. Ball and M. J. Magazine, Sequencing of insertions in printed circuit board assembly, *Operations Research* **36**, 2 (1988) 192–201.
13. J. F. Bard, R. W. Clayton and T. A. Feo, Machine setup and component placement in printed circuit board assembly, *International Journal of Flexible Manufacturing Systems* **6** (1994) 5–31.
14. A. Barnea and D. Sipper, Set-up reduction in PCB automated assembly, *Computer Integrated Manufacturing Systems* **6**, 1 (1993) 18–26.
15. D. Ben-Arieh and M. Dror, Part assignment to electronic insertion machines: Two machine case, *International Journal of Production Research* **28**, 7 (1990) 1317–1327.
16. D. Ben-Arieh and O. Maimon, Annealing method for PCB assembly scheduling on two sequential machines, *International Journal of Computer Integrated Manufacturing* **5**, 6 (1992) 361–367.
17. G. Bhaskar and T. T. Narendran, Grouping PCBs for set-up reduction: A maximum spanning tree approach, *International Journal of Production Research* **34**, 3 (1996) 621–632.
18. D. A. Bodner, M. Damrau, P. M. Griffin, L. F. McGinnis and A. McLaughlin, Virtual prototyping of electronics assembly systems, *Proceedings of the 1998 Industrial Engineering Research Conference*, 1998.
19. D. A. Bodner, M. Damrau, P. M. Griffin, L. F. McGinnis, A. McLaughlin, M. L. Spearman and C. Zhou, Virtual machine models in electronic assembly, *Proceedings of the 1997 Deneb International Conference and Technology Showcase*, Troy, MI, 1997.
20. W. Bolton, *Production Planning and Control* (Longman Scientific and Technical, Essex, UK, 1994).

21. M. L. Brandeau and C. A. Billington, Design of manufacturing cells: Operation assignment in printed circuit board manufacturing, *Journal of Intelligent Manufacturing* **2** (1991) 95–106.
22. T. F. Carmon, O. Z. Maimon and E. M. Dar-El, Group set-up for printed circuit board assembly, *International Journal of Production Research* **27**, 10 (1989) 1795–1810.
23. C.-M. Chang and L. Young, A simultaneous-mounting process for automated printed circuit board assembly, *International Journal of Production Research* **28**, 11 (1990) 2051–2064.
24. S. B. Coble and R. E. Bohn, Setup time reduction for electronics assembly: Combining simple (SMED) and sophisticated methods, Technical Report 97–03, University of California, San Diego, CA, 1997.
25. Y. Crama, O. E. Flippo, J. van de Klundert and F. C. R. Spieksma, The component retrieval problem in printed circuit board assembly, *International Journal of Flexible Manufacturing Systems* **8** (1996) 287–312.
26. Y. Crama, O. E. Flippo, J. van de Klundert and F. C. R. Spieksma, The assembly of printed circuit boards: A case with multiple machines and multiple board types, *European Journal of Operational Research* **98**, 3 (1997) 457–472.
27. Y. Crama, A. W. J. Koleen, A. G. Oerlemans and F. C. R. Spieksma, Throughput rate optimization in the automated assembly of printed circuit boards, *Annals of Operations Research* **26** (1990) 455–480.
28. Y. Crama, A. Oerlemans and F. Spieksma, Production Planning in Automated Manufacturing, Volume 414 of *Lecture Notes in Economics and Mathematical Systems* (Springer-Verlag, 1994).
29. B. Cyr, S. Lambert, G. Abdul-Nour and R. Rochette, Manufacturing flexibility: SMT factors study, *Computers and Industrial Engineering* **33**, (1–2) (1997) 361–364.
30. A. Dagnino, Integrated architecture for assembly planning in an electronics manufacturing environment, *Integrated Manufacturing Systems* **5**, (4/5) (1994) 77–86.
31. M. S. Daskin, O. Maimon, A. Shtub and D. Braha, Grouping components in printed circuit board assembly with limited components staging capacity and single card setup: Problem characteristics and solution procedures, *International Journal of Production Research* **35**, 6 (1997) 1617–1638.
32. M. M. Dessouky, S. Adiga and K. Park, Design and scheduling of flexible assembly lines for printed circuit boards, *International Journal of Production Research* **33**, 3 (1995) 757–775.
33. S. Dillon, R. Jones, C. J. Hinde and I. Hunt, PCB assembly line setup optimization using component commonality matrices, *Journal of Electronics Manufacturing* **8**, 2 (1998) 77–87.
34. M. Driels and J. S. Klegka, An analysis of contemporary printed wiring board manufacturing environment in the USA, *International Journal of Advanced Manufacturing Technology* **7** (1992) 29–37.
35. D. N. Estremadoyro, P. A. Farrington, B. J. Schroer and J. J. Swain, Simulation of memory chip line using an electronics manufacturing simulator, In S. Andradóttir, K. J. Healy, D. H. Withers, and B. L. Nelson, editors, *Proceedings of the 1997 Winter Simulation Conference*, Atlanta, GA, 1997, 1330–1337.
36. Y. Fathi and J. Taheri, A mathematical model for loading the sequencers in a printed circuit pack manufacturing environment, *International Journal of Production Research* **27**, 8 (1989) 1305–1316.

37. K. Feldmann and R. Feuerstein, Developments and challenges in automatic assemblies of electronic products, Technical report, Institute for Manufacturing Automation and Production Systems, 1998.
38. K. Feldmann, J. Franke and B. Zöllner, Optimization of SMT-systems by computer-aided planning, simulation and monitoring, *IEEE/ISHM '90 IEMT Symposium*, Italy, 1990, 102–113.
39. T. A. Feo, J. F. Bard and S. D. Holland, Facility-wide planning and scheduling of printed wiring board assembly, *Operations Research* **43**, 2 (1995) 219–230.
40. L. R. Foulds and H. W. Hamacher, Optimal bin location and sequencing in printed circuit board assembly, *European Journal of Operational Research* **66** (1993) 279–290.
41. S. B. Gershwin, R. Akella and Y. F. Choong, Short-term production scheduling of an automated manufacturing facility, *IBM Journal of Research and Development* **29**, 4 (1985) 392–400.
42. N. N. Z. Gindy and S. M. Ratchev, Integrated framework for selection of machining equipment in CIM, *International Journal of Computer Integrated Manufacturing* **11**, 4 (1998) 311–325.
43. D. Golding, PCB assembly, *Assembly Automation* **15**, 2 (1995) 10–13.
44. H. O. Günther, M. Gronalt and R. Zeller, Job sequencing and component set-up on a surface mount placement machine, *Production Planning and Control* **9**, 2 (1998) 201–211.
45. S. Hashiba and T. C. Chang, Heuristic and simulated annealing approaches to PCB assembly setup reduction, In G. J. Olling and F. Kimura, editors, *Human Aspects in Computer Integrated Manufacturing. IFIP Transactions B-3* (North Holland, Elsevier Science Publishers, Amsterdam, The Netherlands, 1992) 769–777.
46. W. Hernandez and V. J. Leon, An overview of the operations analysis of a two-head flexible assembly machine with interference avoidance, *Computers and Industrial Engineering* **33**, (1–2) (1997) 405–408.
47. M. S. Hillier and M. L. Brandeau, Cost minimization and workload balancing in printed circuit board assembly, Manuscript, 1997.
48. M. S. Hillier and M. L. Brandeau, Optimal component assignment and board grouping in printed circuit board manufacturing, *Operations Research* **46**, 5 (1998) 675–689.
49. G. W. Holcomb, Justifying flexible automation for PCB assembly, *Assembly Automation* **15**, 2 (1995) 14–16.
50. T. Häyrinen, M. Johnsson, T. Johtela, J. Smed and O. Nevalainen, Scheduling algorithms for computer-aided line balancing in printed circuit board assembly, *Production Planning and Control* (2000) forthcoming.
51. M. Jackson and C. Johansson, Real time discrete event simulation of a PCB production system for operational support, In S. Andradóttir, K. J. Healy, D. H. Withers, and B. L. Nelson, editors, *Proceedings of the 1997 Winter Simulation Conference*, Atlanta, GA, 1997, 832–837.
52. S. Jain, M. E. Johnson and F. Safai, Implementing setup optimization on the shop floor, *Operations Research* **43**, 6 (1996) 843–851.
53. N. K. Jha, editor, *Handbook of Flexible Manufacturing Systems* (Academic Press, San Diego, CA, 1991).
54. M. Johnsson, *Operational and Tactical Level Optimization in Printed Circuit Board Assembly*, PhD thesis, University of Turku, TUCS Dissertations 16, 1999.
55. M. Johnsson, T. Leipälä and O. Nevalainen, Determining the manual setting order of components on PC-boards, *Journal of Manufacturing Systems* **15**, 3 (1996).

56. M. Johnsson, T. Leipälä, T. Pulliainen and O. Nevalainen, Integrated program generation system for a PC-board assembly line, Technical Report 52, Turku Centre for Computer Science, 1996.
57. M. Johnsson, S. Peltonen, T. Leipälä and O. Nevalainen, Work load balancing of a generalized flexible flow line in printed circuit board production, In R. V. Mayorga, editor, *Proceedings of the Fifth IASTED International Conference on Robotics and Manufacturing*, Cancun, Mexico, IASTED, IASTED/ACTA Press, 1997, 382–389.
58. P. K. Johri, A heuristic algorithm for loading new work on circuit pack assembly lines, *International Journal of Production Research* **28**, 10 (1990) 1871–1883.
59. P. K. Johri, Engineering a circuit board assembly line for a desired capacity and flowtime, *Journal of Manufacturing Systems* **10**, 6 (1991).
60. T. Johtela, M. Johnsson, J. Smed and O. Nevalainen, Links to commercial software systems for PCB assembly, 1999, http://www.cs.utu.fi/scheduling/pcbsystems/.
61. T. Johtela, J. Smed, M. Johnsson, R. Lehtinen and O. Nevalainen, Supporting production planning by production process simulation, *Computer Integrated Manufacturing Systems* **10**, 3 (1997) 193–203.
62. T. Johtela, J. Smed, M. Johnsson and O. Nevalainen, Fuzzy approach for modeling multiple criteria in the job grouping problem, Technical Report 227, Turku Centre for Computer Science, Dec. 1998.
63. L. P. Khoo and T. K. Ng, A genetic algorithm-based planning system for PCB component placement, *International Journal of Production Economics* **54**, 3 (1998) 321–332.
64. B. Khoshnevis, G. Bottlik and A. R. Azmandian, Simultaneous generation of assembly plans and schedules in electronics assembly operations, *Integrated Manufacturing Systems* **5**, 4/5 (1994) 30–40.
65. Y.-D. Kim, H.-G. Lim and M.-W. Park, Search heuristics for a flowshop scheduling problem in a printed circuit board assembly process, *European Journal of Operational Research* **91** (1996) 124–143.
66. J. S. Klegka and M. Driels, Case studies in printed wiring assembly, *Manufacturing Review* **4**, 4 (1991) 286–292.
67. J. G. Klincewicz and A. Rajan, Using GRASP to solve the component grouping problem, *Naval Research Logistics* **41** (1994) 893–912.
68. K. R. Kumar and T. T. Narendran, A heuristic for sequencing PCBs with due-dates, *International Journal of Operations and Production Management* **17**, 5 (1997) 446–467.
69. S. Lambert, B. Cyr, G. Abdul-Nour and J. Drolet, Comparison study of scheduling rules and set-up policies for a SMT production line, *Computers and Industrial Engineering* **33**, (1–2) (1997) 369–372.
70. T. L. Landers, W. D. Brown, E. W. Fant, E. M. Malstrom and N. M. Schmitt, *Electronics Manufacturing Processes* (Prentice-Hall, Englewood Cliffs, NJ, 1994).
71. A. M. Law and M. G. McComas, Simulation of manufacturing systems, In S. Andradóttir, K. J. Healy, D. H. Withers, and B. L. Nelson, editors, *Proceedings of the 1997 Winter Simulation Conference*, Atlanta, GA, 1997, 86–89.
72. T. Leipälä and O. Nevalainen, Optimization of the movements of a component placement machine, *European Journal of Operational Research* **38** (1989) 167–177.
73. V. J. Leon and B. A. Peters, Replanning and analysis of partial setup strategies in printed circuit board assembly systems, *International Journal of Flexible Manufacturing Systems* **8**, 4 (1996) 389–412.
74. V. J. Leon and B. A. Peters, A comparison of setup strategies for printed circuit board assembly, *Computers and Industrial Engineering* **34**, 1 (1998) 219–234.

75. M. C. Leu and Z. Ji, Computer-aided process planning for printed circuit board assembly, *Proceedings of NEPCON East Conference*, Boston, MA, 1991, 289–297.
76. F.-R. Lin, M. J. Shaw and A. Locascio, Scheduling printed circuit board production systems using the two-level scheduling approach, *Journal of Manufacturing Systems* **16**, 2 (1997) 129–149.
77. W.-L. Lin and V. Tardif, Component partitioning under demand and capacity uncertainty in printed circuit board assembly, *International Journal of Flexible Manufacturing Systems* **11**, 2 (1999) 159–176.
78. C. B. Lofgren and L. F. McGinnis, Dynamic scheduling for flexible printed circuit card assembly, *Proceedings IEEE Systems, Man, and Cybernetics*, Atlanta, GA, 1986, 1294–1297.
79. C. B. Lofgren and L. F. McGinnis, Soft configuration in automated insertion, *Proceedings of the 1986 IEEE International Conference on Robotics and Automation*, San Francisco, CA, 1986, 138–142.
80. C. B. Lofgren, L. F. McGinnis and C. A. Tovey, Routing printed circuit cards through an assembly cell, *Operations Research* **39**, 6 (1991) 992–1004.
81. D. Luzzatto and M. Perona, Cell formation in PCB assembly based on production quantitative data, *European Journal of Operational Research* **69**, 3 (1993) 312–329.
82. O. Maimon and A. Shtub, Grouping methods for printed circuit boards, *International Journal of Production Research* **29**, 7 (1991) 1370–1390.
83. O. Z. Maimon, E. M. Dar-El and T. F. Carmon, Set-up saving schemes for printed circuit boards assembly, *European Journal of Operational Research* **70**, 2 (1993) 177–190.
84. C.-H. Mangin, Line efficiency and productivity measures, *Surface Mount Technology*, Oct. 1999, 114–116.
85. H. B. Marri, A. Gunasekaran and R. J. Grieve, Computer-aided process planning: A state of art, *International Journal of Advanced Manufacturing Technology* **14** (1998) 261–268.
86. L. A. Martin-Vega, DMII: Past, present and future perspectives, Keynote presentation in the 25th International Conference on Computers and Industrial Engineering, New Orleans, LA, Mar. 1999.
87. L. F. McGinnis, J. C. Ammons, M. Carlyle, L. Cranmer, G. W. DePuy, K. P. Ellis, C. A. Tovey and H. Xu, Automated process planning for printed circuit card assembly, *IIE Transactions* **24**, 4 (1992) 18–30.
88. A. Nesbit, Maximising throughput of the component assembly process, *Soldering and Surface Mount Technology* **10**, 1 (1998) 6–9.
89. K. Ohno, Z. Jin and S. E. Elmaghraby, An optimal assembly mode of multi-type printed circuit boards, *Computers and Industrial Engineering* **36**, 2 (1999) 451–471.
90. B. A. Peters and G. S. Subramanian, Analysis of partial setup strategies for solving the operational planning problem in parallel machine electronic assembly systems, *International Journal of Production Research* **34**, 4 (1996) 999–1021.
91. M. Pinedo, *Scheduling: Theory, Algorithms, and Systems* (Prentice-Hall, Englewoods Cliffs, NJ, 1995).
92. B. Powell, The "spider" and the yen, *Newsweek*, Aug. 7, 1995, 46.
93. K. Rajkumar and T. T. Narendran, A heuristic for sequencing PCB assembly to minimize set-up times, *Production Planning and Control* **9**, 5 (1998) 465–476.
94. M. Sadiq, T. L. Landers and G. D. Taylor, A heuristic algorithm for minimizing total production time for a sequence of jobs on a surface mount placement machine, *International Journal of Production Research* **31**, 6 (1993) 1327–1341.

95. J. M. Sanchez and J. W. Priest, Optimal component-insertion sequence-planning methodology for semi-automatic assembly of printed circuit boards, *Journal of Intelligent Manufacturing* **2** (1991) 177–188.
96. C. Saygin, S. E. Kiliç, T. Tóth and F. Erdélyi, On scheduling approaches of flexible manufacturing systems: Gap between theory and practice, In T. Borangiu and I. Dumitrache, editors, *Intelligent Manufacturing Systems 1995 (IMS '95): A Proceedings Volume from the 3rd IFAC/IFIP/IFORS Workshop*, (Pergamon Press, Oxford, UK, 1997) 61–66.
97. S. F. Shevell, J. A. Buzacott and M. J. Magazine, Simulation and analysis of a circuit board manufacturing facility, In J. Wilson, J. Henriksen, and S. Roberts, editors, *Proceedings of the 1986 Winter Simulation Conference*, 1986, 686–693.
98. A. Shtub and O. Maimon, Role of similarity in PCB grouping procedures, *International Journal of Production Research* **30**, 5 (1992) 973–983.
99. J. Smed, M. Johnsson, M. Puranen, T. Leipälä and O. Nevalainen, Job grouping in surface mounted component printing, *Robotics and Computer-Integrated Manufacturing* **15**, 1 (1999) 39–49.
100. J. Smed, T. Johtela, M. Johnsson, M. Puranen and O. Nevalainen, An interactive system for scheduling jobs in electronic assembly, *International Journal of Advanced Manufacturing Technology* (2000) forthcoming.
101. J. S. Smith and B. A. Peters, Simulation as a decision-making tool for real-time control of flexible manufacturing systems, *IEEE International Conference on Robotics and Automation*, Leuven, Belgium, May 1998, 586–590.
102. SMTnet, Glossary, 1999, http://www.smtnet.com/.
103. T. A. Spedding and G. Q. Sun, Application of discrete event simulation to the activity based costing of manufacturing systems, *International Journal of Production Economics* **58** (1999) 289–301.
104. D. R. Sule, A heuristic procedure for component scheduling in printed circuit pack sequencers, *International Journal of Production Research* **30**, 5 (1992) 1191–1208.
105. M. R. Supinski, P. J. Egbelu and E.-A. Lehtihet, Automatic plan and robot code generation for a PCB assembly, *Manufacturing Review* **4**, 3 (1991) 214–224.
106. C. S. Tang and E. V. Denardo, Models arising from a flexible manufacturing machine, *Operations Research* **36**, 5 (1988) 767–784.
107. G. D. Taylor, Jr, Simulation of memory chip line using an electronics manufacturing simulator, In O. Balci, R. P. Sadowski, and R. E. Nance, editors, *Proceedings of the 1990 Winter Simulation Conference*, New Orleans, LA, Dec. 1990, 567–569.
108. S.-H. G. Teng and S. S. Garimella, Manufacturing cost modeling in printed wiring board assembly, *Journal of Manufacturing Systems* **17**, 2 (1998) 87–96.
109. S. D. Thompson and W. J. Davis, An integrated approach for modeling uncertainty in aggregate production planning, *IEEE Transactions on Systems, Man, and Cybernetics* **20**, 5 (1990) 1000–1012.
110. Universal Instruments Corporation, 1999, http://www.uic.com/.
111. P. J. M. van Laarhoven and W. H. M. Zijm, Production preparation and numerical control in PCB assembly, *International Journal of Flexible Manufacturing Systems* **5** (1993) 187–207.
112. C. Wang, Layout designs for robotic PCB assembly, *Soldering and Surface Mount Technology* **10**, 2 (1998) 36–48.
113. C. Wang, L.-S. Ho and D. J. Cannon, Heuristics for assembly sequencing and relative magazine assignment for robotic assembly, *Computers and Industrial Engineering* **34**, 2 (1998) 423–431.

114. R. E. Watkins and J. K. Cochran, A line balancing heuristic case study for existing automated surface mount assembly line setups, *Computers and Industrial Engineering* **29**, (1–4) (1995) 681–685.
115. R. J. Wittrock, An adaptable scheduling algorithm for flexible flow lines, *Operations Research* **36**, 3 (1988) 445–453.
116. Z. Xu, K. Carlson, R. Kurschner and S. Randhawa, An integrated methodology for surface mount PCB configuration, *Computers and Industrial Engineering* **35**, (1–2) (1998) 53–56.
117. S. H. Yeo, C. W. Low and K. H. Yong, A rule-based frame system for concurrent assembly machines, *International Journal of Advanced Manufacturing Technology* **12**, 5 (1996) 370–376.
118. P. Zarrow, Automatic components placement systems, *Circuits Assembly Magazine*, Oct. 1996.
119. M. Zhou and M. C. Leu, Petri net modeling of a flexible assembly station for printed circuit boards, *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, Sacramento, CA, Apr. 1991, 2530–2535.
120. W. H. M. Zijm and A. van Harten, Process planning for a modular component placement system, *International Journal of Production Economics* **30–31** (1993) 123–135.

CHAPTER 2

COMPUTER TECHNIQUES AND APPLICATIONS FOR CONCURRENT MOLD DESIGN SYSTEMS IN MANUFACTURING

YUH-MIN CHEN*,‡ and RONG-SHEAN LEE†

*Institute of Manufacturing Engineering,
†Department of Mechanical Engineering,
National Cheng Kung University,
Tainan, Taiwan 70101, ROC
Email: ‡ymchen@mail.ncku.edu.tw

Injection molding has been used extensively to produce components for appliances and computer products. This manufacturing process can reduce overall manufacturing costs. Mold design is essential in molding product and process development. This chapter discusses how computer techniques and concurrent engineering are applied to mold design and manufacturing.

Keywords: Mold design; concurrent engineering; computer based tools.

1. Introduction

Net shape manufacturing uses dies or molds to produce parts in finished (net) or near-finished (near-net) dimensions.[3] Injection molding, a net shape process, has been used extensively to produce components for appliances and consumer products due to its potential for high production rates and material, energy, time and labor conservation. Related investigations have conferred that this manufacturing process can reduce overall manufacturing costs.[23,53,54]

Molding product and process development includes product design, process design, mold design, and mold manufacturing process planning. Mold design is essential in molding product and process development, as evidenced by that (a) the cost and quality of a mold determines the cost and quality of a product and (b) molds are normally expensive and sophisticated.[1,26,29,31]

Production has been increased by the widespread use of computer-based tools such as CAD, CAE, and CAM, in molding product and process development,

as well as mold design.[28,36,48] However, in practice, mold design is performed separately from product design and mold manufacturing process planning. In addition, inefficient communication between individuals performing these activities results in long mold development time and incompatibility within the injection molding industry.

Concurrent engineering has been employed in recent years to shorten the time to market, reduce product and development costs, and increase product quality by resolving relevant issues in the early stages of the development process.[16,44,45] Owing to its prominent role in molding product and process development, mold design is viewed as the most important application area for concurrent engineering, thus deserving extensive study.

This chapter discusses how computer techniques and concurrent engineering methodology are applied to mold design and manufacturing. Concurrent engineering and business process re-engineering[35] are first reviewed as the basis for developing a concurrent process for net shape product and process development as well as developing a concurrent mold development process. To develop a concurrent process, the conventional process is analyzed and represented in terms of an "as-is" model that indicates the process activities and their sequence. Using the "as-is" model, process re-engineering is performed with the output of a "to-be" model. To support the concurrent process practice, the functional requirements for computer aided concurrent net shape product and process development are identified. In light of these requirements, we propose a collaborative framework for a computer-based environment for concurrent net shape product and process development. Also discussed herein are the techniques for developing an application- or process-specific tool, such as a mold design system, in a computer aided concurrent net shape product and process development environment.

2. Review of Basic Concepts

This section reviews concurrent engineering as well as business process re-engineering and business process modeling. They are used as the basis for concurrent process development.

2.1. *Concurrent engineering*

Concurrent engineering (CE) improves product marketability. In this section, we discuss the basics of concurrent engineering, its definition, importance, characteristics, and implementation.

In the late 1980's, concepts such as team design, simultaneous engineering, concurrent engineering, and integrated product and process development emerged to improve product quality and reduce the cost and time to market. They are all considered product development-related issues from conception to disposal, in a concurrent and integrated manner.

To clarify these concepts, the U.S. Institute for Defense Analysis (IDA) defined concurrent engineering as:

Concurrent engineering is a systematic approach to the integrated, concurrent design of products and their related processes, including manufacture and support. This approach is intended to cause the developers, from the outset, to consider all elements of the product life cycle, from conception to disposal, including quality, cost, schedule, and user requirements.[50]

Concurrent engineering requires that firms restructure their corporate organization and redesign their engineering processes. It also adopts various management, reengineering process, system architectures, and information technologies.[51,52] In practice, implementing concurrent engineering includes the following aspects:

(1) Employment of a systematic approach to develop concurrent products and processes.
(2) Application of design principles and engineering methods to efficiently and effectively optimize products and processes.
(3) Use of multi-functional teams to carry out integrated product and process design.
(4) Utilization of an integrated computer based engineering environment (i.e. computer aided concurrent engineering environment).

The first aspect indicates the management actions to improve the organizational procedures used to develop a product. In addition to providing better process operation, techniques for engineering process modeling and reengineering, engineering process control and coordination are employed to form a systematic development process. Methods such as Quality Function Deployment (QFD) constitute a systematic means of fulfilling the customers' requirements and transforming those requirements into engineering and production solutions, accounting for its extensive use in product development.[16]

In product and process optimization, methods such as value engineering, cost estimating, failure mode effect analysis, statistical process control (SPC), and Taguchi method are employed to analyze and tune component and process parameters for enhanced system performance, manufacturability, or maintainability.[7,18,21,27,46]

Multidisciplinary teams are the most significant elements of concurrent engineering. A multi-functional team capable of achieving its goal should be structured properly and managed effectively.[10] The design and structure of a group is a senior management function, while the daily operations of a team involves individuals from each product life cycle area and at different levels.

An integrated computer-based engineering environment provides information for rapid and intelligent decision-making throughout the entire development process.[9]

The environment should provide tools for information management, such as an engineering data management system (EDMS),[43] computer-mediated communication utilities, e.g. E-mail and desktop conferencing systems, and group decision support systems, process control and coordination tools, and application tools such as CAx tools, and Design for X systems.

2.2. *Business process re-engineering*

Process re-engineering rationalizes and facilitates the engineering process concurrent in reducing excessive delay or costs. It consists of the tasks of process goal identification, problem identification and resolution, process rationalization and concurrentization, and validation as shown in Fig. 1.

The first step of process re-engineering clarifies the goals of the process and ensures that the goals satisfy the corporate mission(s) and strategy(ies). *Strategic analysis* is performed to develop the strategy and methods for achieving the process goals. Similarly, the strategy for achieving the process goals should also meet the corporate strategy. A cause-effect diagram can be used for strategy analysis.

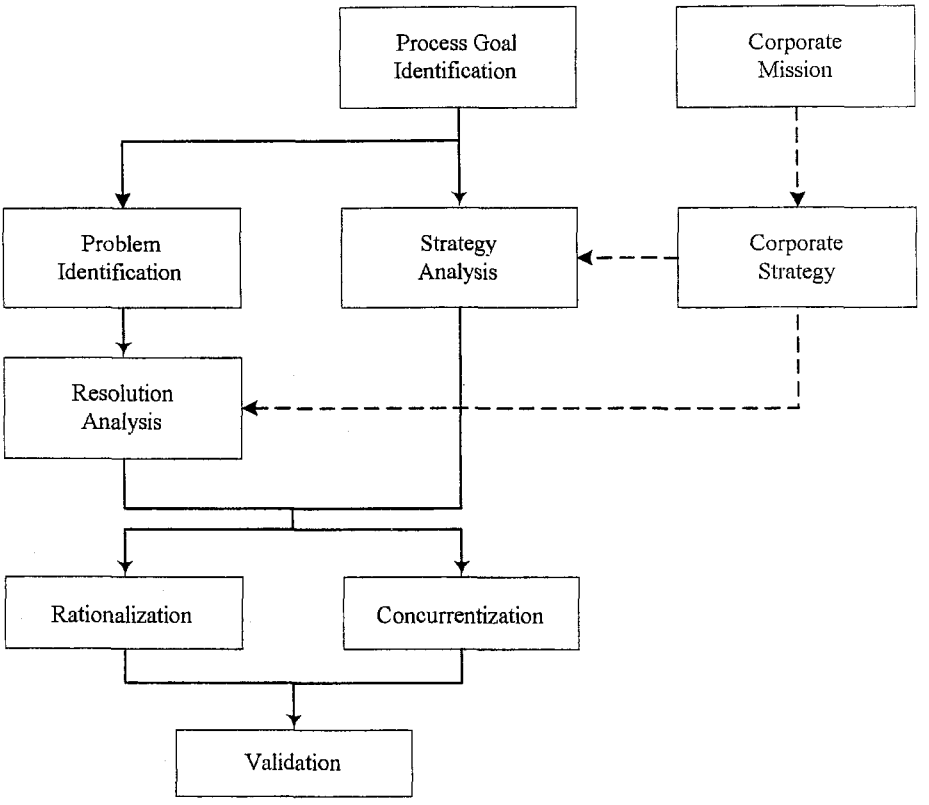


Fig. 1. Steps of process re-engineering.

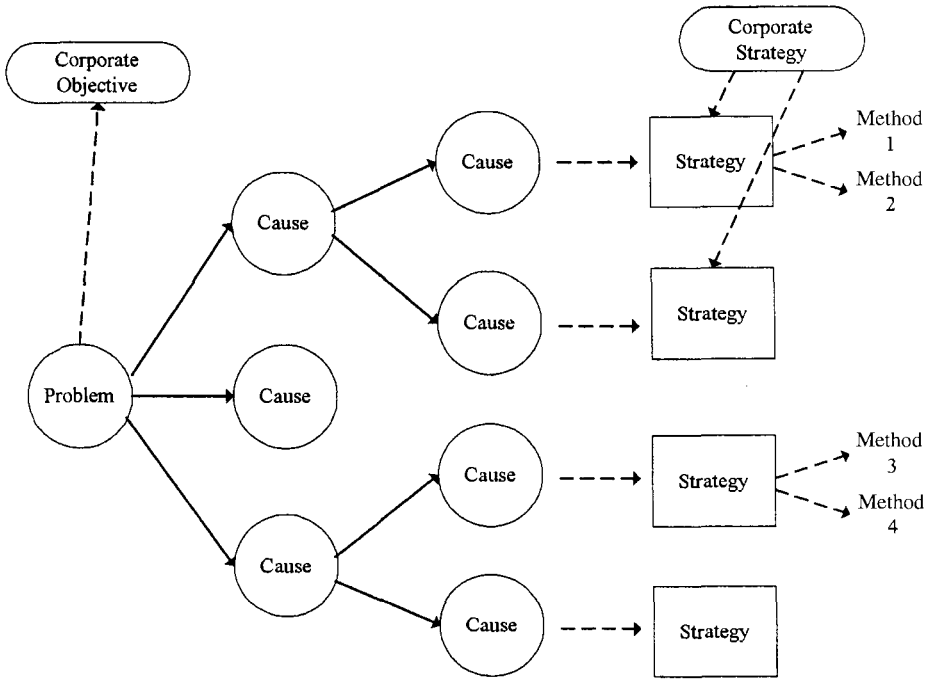


Fig. 2. Cause-effect diagram for problem identification and resolution analysis.

Problem identification and resolution attempts to identify the problems in the process and analyze their causes. The resolutions, each of which may consist of one or several methods, are then developed to resolve the problem. Cause-effect diagrams are also used in the analysis. Figure 2 illustrates an example of a cause-effect diagram for problem identification and resolution analysis.

Process rationalization attempts to ensure that each activity in the engineering process contributes to be process goals. *Value analysis* is conducted on each activity by the following:

(1) Identifying the functions of the activity.
(2) Clarifying whether if the function helps achieve the process goals.
(3) Identifying the reason for the existence of each activity.
(4) Clarifying the importance of the output of the activity to its consumer(s).
(5) Checking if this activity can eliminated or replaced.

Other principles of process rationalization include the following:

(1) Shortening the duration of activities.
(2) Eliminating redundant activities.
(3) Discarding non-value-added activities.

(4) Partitioning a complicated activity into simpler and more controllable sub-activities.
(5) Combining two series and closely related activities, and eliminating cycles.[38]

Process concurrentization is conducted by analyzing how activities based on the input, output, and coordination information of activities are related. The relationships between activities can be classified as *sequential, concurrent* or *collaborative.* The sequential relationship between activities defines a strict ordering relationship between two activities, where an activity should be fully executed before the next one. Activities with concurrent relationships are both the successors of a certain activity and do not interact with each other and, therefore, can be performed concurrently. Activities with collaborative relationships should be performed cooperatively or collaboratively through information and knowledge sharing as a result of their strong interactions.

Validation checks if the to-be process is acceptable and better than the as-is process. Theoretically, process simulation and evaluation should be performed based on the parameters of quality, cost and time. However, since most engineering activities are non-procedural and the quality, cost, and time consumed heavily depend on the product complexities and actors' experiences, how to develop an evaluation model and determine the parameter values are still relevant issues.[2] Since this research does not address these issues, validation at this stage is done by simply comparing the number of activities in the to-be and as-is processes and the number of steps in each of their activities.

2.3. *IDEF0 and business process modeling*

A suitable process modeling method is required for engineering process analysis. Since engineering processes are part of enterprise business processes, for process modeling, an integral view should be provided from the perspective of enterprise modeling.[19] Therefore, this section first reviews the principles of enterprise modeling and discusses a representative approach for engineering modeling.

2.3.1. *Enterprise modeling*

Enterprise modeling focuses mainly on supporting analysis of an enterprise. An enterprise can be viewed as a large collection of concurrent business processes executed by a set of functional entities that contribute to the business objectives.[59] Enterprise modeling develops interrelated models to describe various facts of an enterprise by performing activities with methods to address some desired modeling outcome.[5,24,25,59] Functionality, behavior, organization, and information are the most common aspects used in modeling enterprises.[19,25,59] Functional aspects define what must be done, behavioral aspects describe how and when something must be done, informational aspects indicate what data are used or produced and

their relationships, and finally, organizational aspects define who must do what, when and where.[19]

Most representative enterprise modeling works include ISO work, CEN ENV 40 003, CIMOSA, GRAI/GIM, PERA, ARIS, GERAM.[59] Although having different concerns in an enterprise, most of these modeling works comply with the principles of enterprise modeling, such as separation of concerns, functional decomposition, and generically. Details of this work and additional principles can be found in Ref. 59.

2.3.2. *Integrated definition functional modeling method*

IDEF0, a part of the IDEF family of methods, is a structural analysis and modeling technique particularly designed to model the decisions, actions, and activities of organizations or complex and interrelated systems.[42] IDEF0 is also known as a functional modeling method for analyzing and communicating the functional perspective of a system owing to its ability to effectively assist system analysis and communication between the analyst and the customer.

The result of IDEF0 functional modeling is a hierarchical functional decomposition, consisting of five basic elements: functional blocks and input, output, controls, and mechanisms of each functional block. The functional blocks represent the activities and tasks of the system and process under investigation. Inputs to a function are materials transformed by the function. Outputs of a function are the results transformed from the inputs by the function. Controls to a function are the constraints, criteria, or conditions governing the performance of the function. Mechanisms are the means used to perform or the resources used to support the function.

IDEF0 modeling is characterized by its hierarchical decomposition. Higher level activities imply more general and abstract concepts, which can be exploded into lower level activities representing more specific and detailed concepts.

3. Concurrent Net Shape Product and Process Development

This section presents the development of a concurrent process for net shape product and process development using concurrent engineering and engineering process improvement. The conventional net shape product and process development is first analyzed to identify the involved activities and their interactions. The analysis result(s) is denoted as an "as-is" model using Integrated Definition for Functional modeling (IDEF0) technique. Development process re-engineering is then performed based on the as-is model. The result of re-engineering is a "to-be" model that shows the concurrent net shape product design and process development.

3.1. *Net shape product and process development*

3.1.1. *Net shape design and manufacturing*

Net shape manufacturing produces the final shapes of discrete parts to finish (net) or near finish (near net) dimensions that are obtained by using dies or molds. Related

manufacturing processes can be melt processes (such as die casting, sand casting, injection molding), shaping from powder (powder metal forming), and forming from sheet and billet. Depending on the specific process, the initial material may be a shapeless liquid or powder, a billet of simple geometry, or a sheet material of simple geometry. The final part generally has a relatively complex geometry with a well-defined shape, size, accuracy, appearance, and physical properties. In contrast to metal removal processes where a sequence of discrete operations are performed to reach the final shape, the final shape in net shape operations is achieved through the geometry of a die or mold.

Basically, manufacturing engineering for net shape manufacturing consists of two phases: process selection and process design. Process selection for net shape manufacturing is based on functional requirements, the geometry (shape, size, surface finish, and tolerance), and the material (composition and heat treatment) of a part. At process design, decisions are made regarding the overall manufacturing maintenance and support costs associated with a specific product. These decisions heavily depend on the geometric characteristics of the part. A tremendous amount of heuristic and empirical knowledge is available and widely used.[1,26] However, a problem that arises when using this knowledge in computer aided systems is the inability to completely and appropriately represent the complex geometry in ways deemed appropriate for reasoning about the manufacturing process.

3.1.2. *Activities of net shape product and process development*

Figure 3 shows the as-is model of net shape product and process development. Activities of net shape product and process development include conceptual design, preliminary design, process selection, design for net shape processes, process design, die/mold design, and die/mold manufacturing process planning. All activities contain iterative decision-making based on empirical and scientific knowledge. In conceptual design, a sketch or a conceptual product model is configured based on the product functional requirements. Preliminary design concentrates mainly on constructing the preliminary product geometry. The preliminary product geometry includes components that meet the product functional requirements, as well as their relationships.

During process selection a particular process is constrained by factors such as the preliminary product geometry, lot sizes, equipment availability, tolerances required, material requirements, and desired mechanical properties of the final product. The designer, according to his/her experience and knowledge, assesses the manufacturing alternatives. Basically, two principles are followed for process selection: (1) the decision to select a net shape process should be made by simultaneously considering the part, material(s), machine to be used, and tooling; (2) the process selection must be closely coupled with the design process.

After a manufacturing process is selected, a detailed design or design for net shape process is performed to refine the preliminary product geometry into one

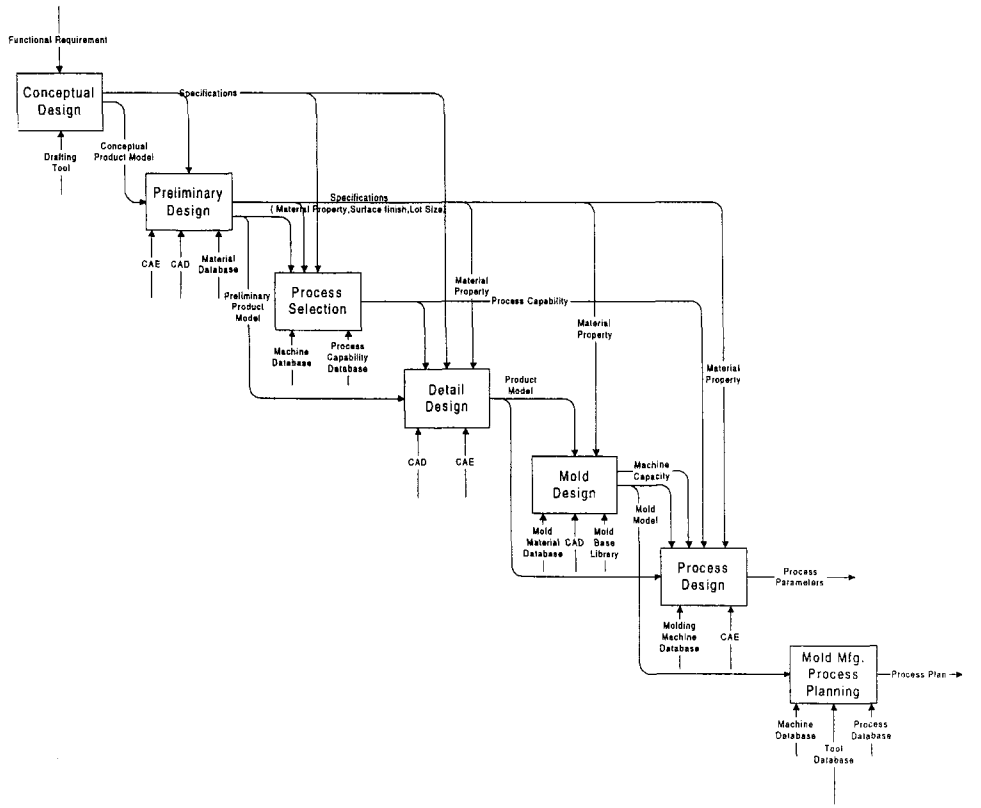


Fig. 3. The as-is model of net shape product and process development.

which is functionally acceptable and compatible with the selected manufacturing process. This involves a wide variety of design expertise, engineering knowledge and the selected process. In additional producibility, factors affecting the life and cost of dies/molds and manufacturing process optimization are also considered to achieve higher quality, lower product costs and a shorter manufacturing cycle time.

During process design, the manufacturing cycle time must be reduced, and the overall manufacturing maintenance and support costs associated with a specific product must be determined as well. These decisions depend heavily on the geometric characteristics of the product and the results of the die/mold and tool design.

Producibility and the cost of dies/molds are the major considerations in die/mold design. However, the product shape and process design affects such considerations. For example, the shape of injection mold cavities nearly complements the product shape. Similarly, the forming sequence for forgoing is determined based on the product geometry, indirectly affecting the geometry of the forming dies. Die/mold design involves similar operations in product design with additional requirements for numerical computations and checks. In addition, several questions

must be resolved during die/mold design: actual die/mold geometry, accounting for shrinkage and warpage, placement of the parting line relative to the part, locations and sizes of the gates, and specification of the operating conditions.

Process planning for die/mold fabrication can be defined as systematically determining the detailed methods by which dies/molds can be manufactured economically. In general, the inputs to process planning are design data, raw material data, facilities data and quality requirement data. Both design and quality requirement data are defined and specified during die/mold design based on the product specifications and requirements.

3.1.3. *Conventional CAD/CAM applications in net shape design and manufacturing*

Increasing the efficiency and quality of net shape product design and manufacturing by utilizing and developing computer-based tools have received extensive interest. According to Fig. 4, the applications of computer-aided design and manufacturing tools in net shape design and manufacturing can be classified into the following areas[3]:

(1) Application of CAD tools for product preliminary design, die/mold design.
(2) Computer-aided simulation of material flow to predict the die fill, defect formation and die stresses.
(3) Application of artificial intelligence and expert system methodology in product design, process design, die/mold design, and die/mold manufacturing process planning.
(4) Application of CAM for die/mold manufacturing.

However, in practice, designers and engineers, separately and with different tools, conduct the net shape design and manufacturing tasks. Therefore, although a product can occasionally be designed with CAD tools, other development tools cannot directly use the design results. Quite frequently, the product design meets the product functional requirements but is not manufacturable. The lack of communication between design and manufacturing incurs numerous inconsistency problems in manufacturing enterprises. Figure 4 also illustrates the sequential nature of the conventional net shape product and process development procedure. Consequently, the iterative development process takes a greater amount of time due to the trial and error of each development activity and the difficulty of compromising the concerns of different aspects.

3.2. *Net shape development process reengineering*

Business process reengineering, concurrent engineering, and multi-functional teamwork are adopted in process reengineering. Development process reengineering provides a uniform standard for net shape product and process development to increase

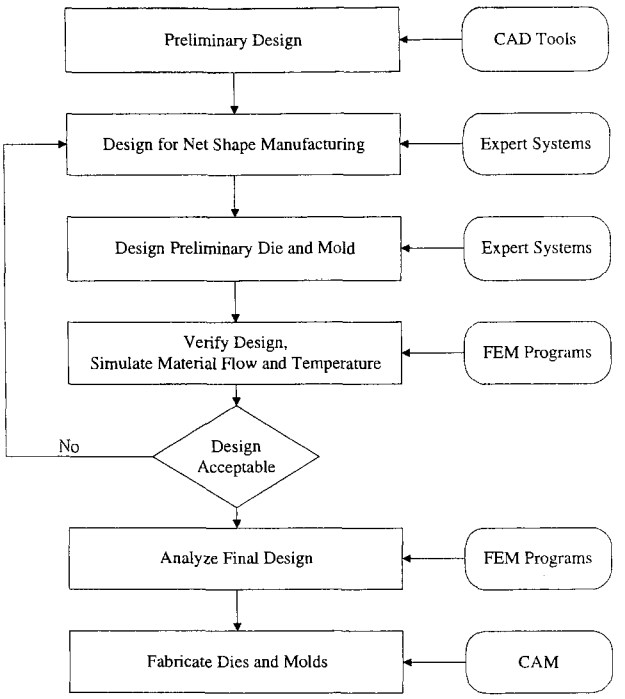


Fig. 4. CAD/CAM applications in conventional net shape design and manufacturing.

development efficiency and consistency. In this section, injection molding product and process development are used as examples of process reengineering.

Based on the above, a concurrent development process is developed through the following steps:

(1) Analyze the net shape development activities into levels of tasks.
(2) Evaluate each task for its contribution to the entire process.
(3) Adjust the process sequence based on the inputs and outputs to and from each task.
(4) Analyze the relationships and interactions among tasks and classifying task relationships into "concurrent", "collaborative", and "sequential" categories based on the task interactions.
(5) Develop a concurrent development model on the basis of the above results.

Before performing reengineering, each development activity is further refined into different levels of tasks, as shown in Fig. 5, based on the as-is model. The product design activity includes preliminary design, moldability assessment, parting line specification, and detailed design for injection molding. A material can be selected before the moldability assessment or during process design. Mold design involves parting line specification, mold preliminary design, cavity and core layout, molding feature layout, and cooling feature layout.

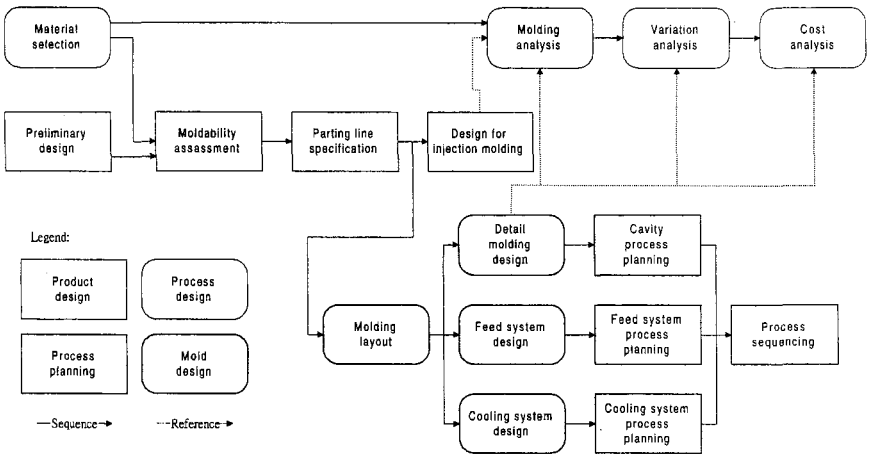


Fig. 5. Sequence of tasks of molding product and process development.

The process design includes selecting the material, as well as analyzing the molding, molding part dimension variation, and manufacturing cost. According to the mold geometry characteristics, planning the mold manufacturing process is classified into the cavity process, feed system process, cooling system process, stock machining process, heat treatment process, post process, and process sequencing.

Value analysis is first performed on each task to ensure its contribution to the entire development process. Basically, the above tasks are required for the development process. The next step is to adjust the sequence, as performed by analyzing the inputs and outputs involved in each task. Linking the inputs and outputs forms the process sequence. Figure 5 also shows the new development sequence.

The preliminary design is a product model, which is the input to assess the moldability. The parting lines are initially determined when assessing the moldability and specified after confirming it. The locations of parting lines also affect the cavity layout, subsequently impacting the detailed design for injection molding. Therefore, the input to molding layout and detailed design for injection molding is the preliminary product model with parting lines. Designs of the feed system and the cooling system are based on the molding layout. Molding analysis, variation analysis, and cost analysis are performed on product geometry or molding geometry based on the selected material(s). Cavity process planning, feed system process planning, and cooling system process planning are performed on the mold cavities, feed system, and cooling system, respectively. Finally, process sequencing is applied to the cavities, feed system, and cooling system.

The interactions or relationships among the development tasks are identified by analyzing the input, output, constraints, and knowledge to, from and employed in each task. Figure 6 shows an interaction matrix that indicates the interactions among the molding product and process development tasks. The interaction symbol at the upper-left corner of each column-row intersection refers to a situation in which

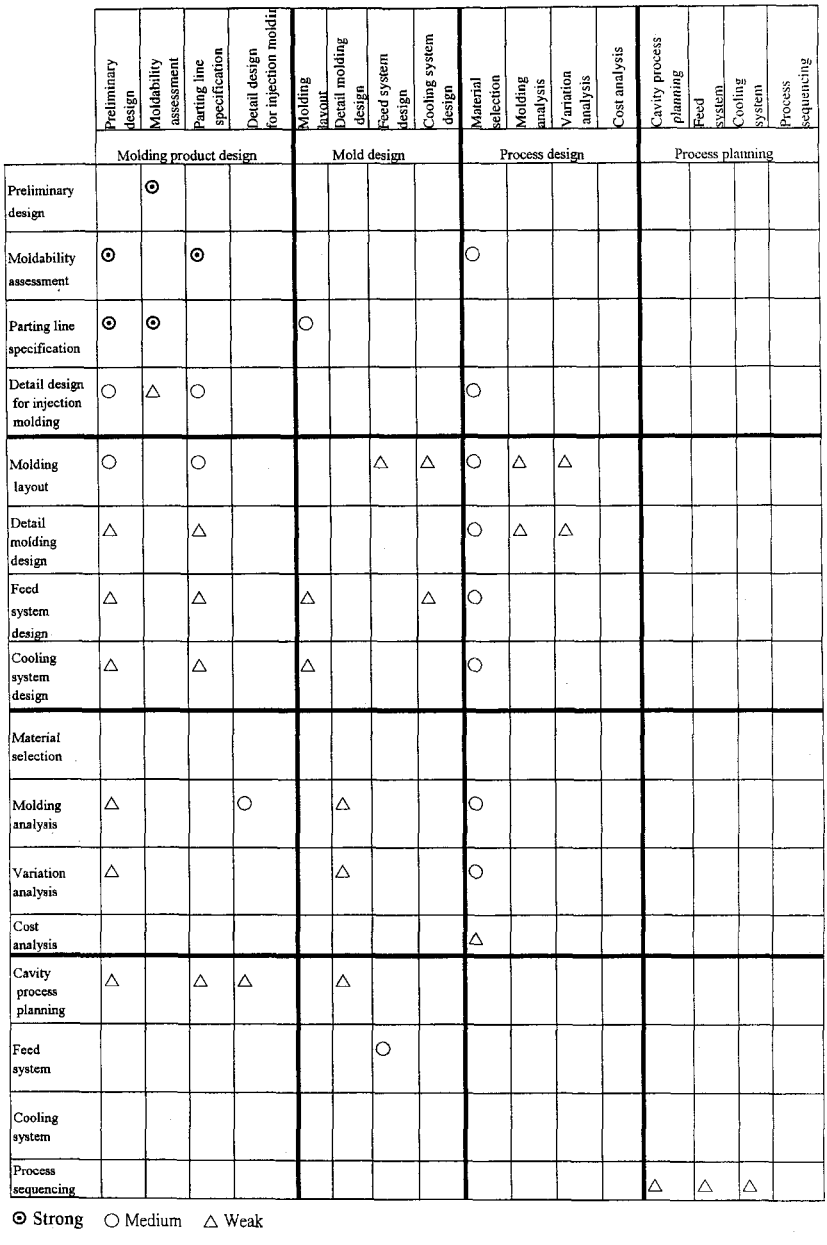


Fig. 6. Interaction matrix for some of the product and process development tasks.

the task of this column influences the row task. The symbol at the bottom-right corner of each intersection refers to a situation in which the row task influences the task of the column. A dotted circle denotes a strong influence between tasks, a blank circle represents a medium one, and a triangle is a weak one.

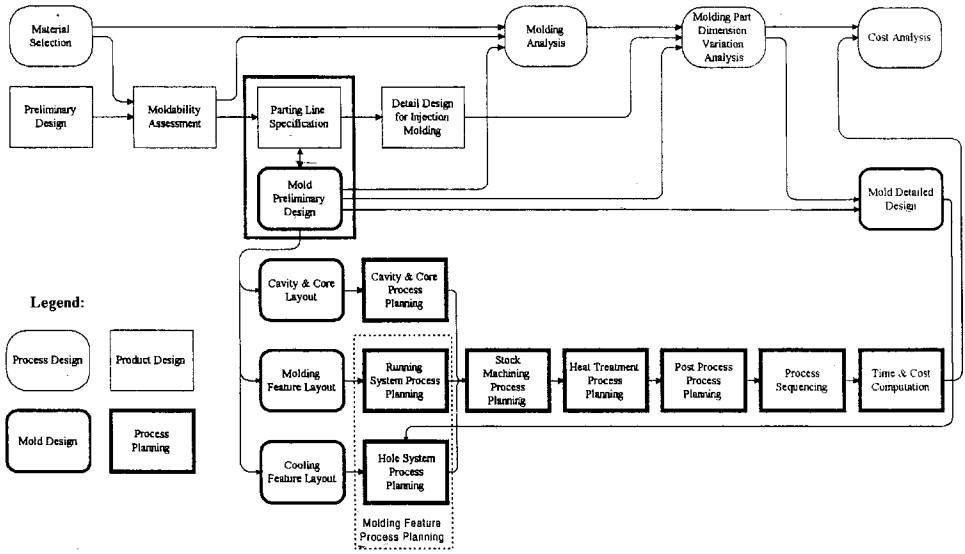


Fig. 7. Concurrent molding product and process development process.

The interaction matrix classifies the relationships between the tasks as either concurrent or collaborative. Tasks with concurrent relationships are the successors of a certain task and do not interact with each other, thereby allowing them to be performed concurrently. Tasks with collaborative relationships should be performed collaboratively owing to their strong interactions. Examples of concurrent tasks are detailed mold design, feed system design and cooling system design. Examples of collaborative tasks are preliminary design and moldability assessment. Figure 7 depicts the concurrent net shape product and process development model developed based on the task sequence and interaction matrix. As parting line specifications affect the cavity layout and moldability assessment, it should be performed collaboratively by the product designer and mold designer.

One of the characteristics of this concurrent process is the possibility of reducing development cycle time. For example, instead of waiting for a complete mold design, cavity process planning, feed system process planning, and cooling system process planning can be performed right after their corresponding, preceding tasks are finished. The other characteristic of this concurrent process improves the development quality because all of the potential problems are considered, reviewed, and resolved during the collaborative development.

4. Concurrent Mold Development Process Development

This section presents the development of a concurrent process for mold development by applying concurrent engineering and engineering process improvements. Similar to that for net shape product and process development, the conventional mold design

and manufacturing process are first analyzed to identify the involved activities and their interactions. This analysis result is represented as an "as-is" model using IDEF0 function modeling method. A development process reengineering is then performed based on the as-is model. Reengineering focuses on obtaining a "to-be" model that shows the concurrent mold design and manufacturing process.

4.1. *Conventional mold design process*

Conventional mold design includes shrinkage design, determination of cavity number, cavity layout, determination of parting linen, feed system design, cooling system design, ejector design, and venting design. Figure 8 illustrates their sequence.

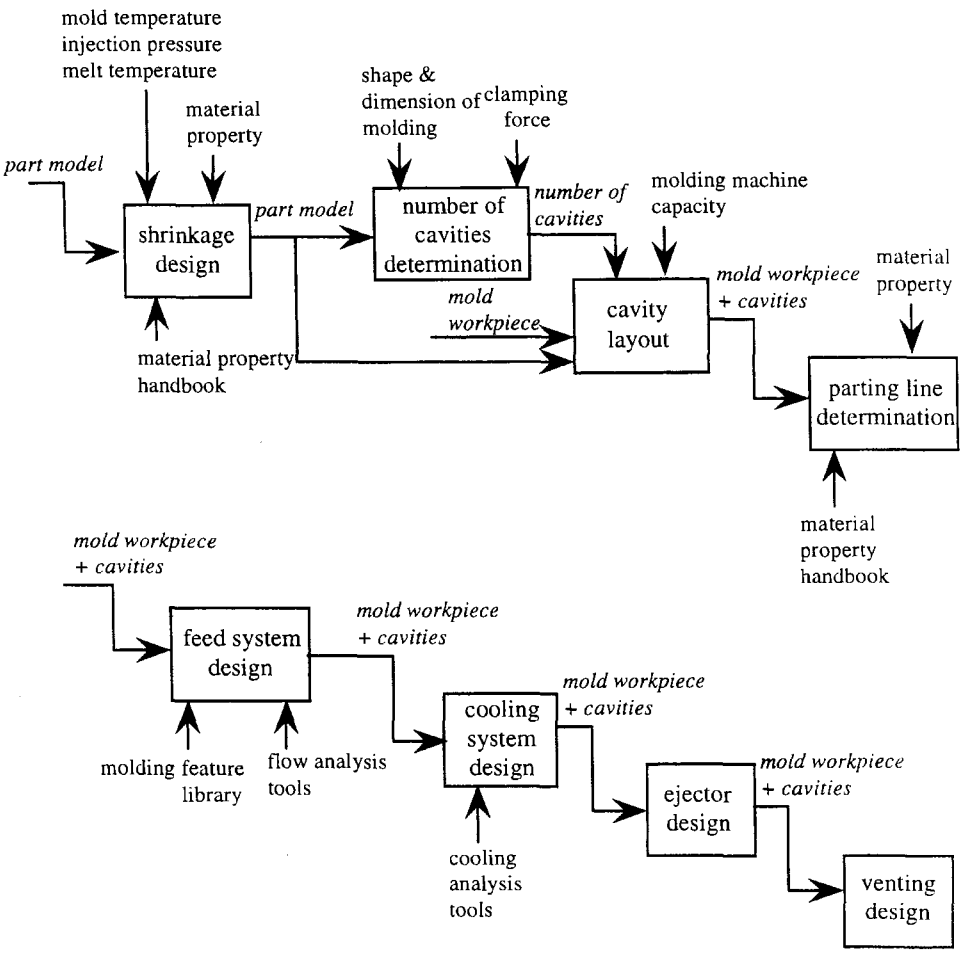


Fig. 8. Conventional mold design process.

The input to the mold design process is a product model. Before mold design, the product geometry is modified to account for material shrinkage that occurs during the processing operation. Shrinkage is the dimension to which a cavity and core should be fabricated to produce a part of desired shape and size. Basically, the shrinkage rate is a function of the material properties, mold temperature, part thickness, injection pressure, and melt temperature.

The cavity layout process begins once the product model has been modified to reflect the shrinkage allowance. In developing a cavity layout, whether or not a single- or a multiple-cavity mold should be used must be decided. Also considered are factors such as period of delivery, quality control requirements, cost of the moldings, polymer used, shape and dimensions of molding, and injection molding machine capacity.

Once the cavity layout is decided, the location and type of gating for the product is determined. A gate is a smaller cross section from which the molding can be easily separated from the runners. The positioning and dimensioning of gates are critical for achieving the required properties and appearance of the molding. Similarly, for mold construction, the gates must be located so that material fillability and uniformity are ensured. In principle, the gates are located at the thickest part of the molding, preferably at a spot where the function and appearance of the molding are not impaired. Furthermore, the gate must be located so that weld lines are avoided. Gates should always be made small at the start because they can easily be made larger, but cannot be easily be reduced in size.

Designing the runner system is the next step. Designers work to design the smallest, adequate runner system to maximize efficiency in raw material use and energy consumption in molding. Runner size is generally constrained by the amount of pressure drop and the injection capacity of the machine.

Forming the transition from the hot molten thermoplastic to the considerably cooler mold, a sprue is seen as a part of the flow length of the plastic and must be of such a dimension that the pressure drop is minimal and its ability to deliver material to the extreme out position is not impaired. The starting point for determining the sprue size is the main runner, and the outlet of the sprue should not be smaller than the runner diameter at the meeting section.

The gates, runners, and sprue are added as molding features. During molding feature design, commercial software for flow analysis is available for use to (1) minimize molded stress in local areas, (2) help determine gate location to provide the desired fill pattern, or (3) facilitate the positioning of weld lines and meld lines in areas where they impact the product function the least.[36,48,53,54]

When the number of cavities and cavity layout is optimized, the minimum mold base size required is selected to accomplish the proposed layout. Information such as the expected cavity insert sizes and other criteria such as the proposed type of runner layout may be required. The criterion generally used to establish the overall size of the mold base is that the ejector plate must completely cover, or contain within its bounds, all of the part cavity area in the

design. The standard mold bases are now available in commercial CAD/CAM systems.

Determining the parting lines is a major task in mold design. Doing so requires examining the following points: (1) flow pattern, overflows or gate area and location, (2) the presence of slides, cores, and inserts, (3) the desired shape or appearance at the parting line, and (4) critical dimensions, as well as trimming or machining operations.

To remove molding from the mold, certain geometrical considerations must be made. First, the molding should have no undercut sections that would lock if it is pulled from the mold. Undercuts cannot always be avoided from the design. Then, movable cores or slides must be used. However, changes in part design should be considered to simplify the mold construction and reduce the number of slides or cores.

All surfaces in line with the mold opening direction must have a certain draft to facilitate ejection of the molding from the mold. The draft required is determined primarily on the material properties, geometry, tolerance requirements, local shrinkage rate, and location of the parting. It also depends on whether or not a machining operation is performed on the part after the molding operation.

A sharp corner in a mold causes both stress concentration and material filling problems. Hence, rounding external corners and filleting internal corners are necessary in molding design as well as in mold design. The proper radius of the corner or fillet is based on the material properties and geometry.

An important aspect of mold design is the provision of adequate cooling arrangements. The cooling system, which is an essential mold feature requiring special attention in mold design, should ensure rapid and uniform cooling of the molding. In the design of mold components and the layout of guides and ejectors, allowances should be made for proper size and positioning of the cooling system.[18,22] Tools are also available, in the form of cooling analysis programs, to recommend a cooling system design.

Venting is done using small gapes or vents provided in the mold parting lines, or other small channels in the mold (i.e. around the ejector pins and cores). Vents must be provided at the end of the flow path(s).

Once the cooling system has been designed, remaining components such as ejector pins, support pillars, and parting line locks, may be incorporated into the mold design. Libraries of mold component parts are now available to either (1) create components, depending on their specific dimensions, or (2) retrieve components on file in the database.

4.2. *Computer applications in mold design and manufacturing*

Computer-Aided Design and Manufacturing (CAD/CAM) technology has significantly impacted mold design and manufacturing. With CAD tools, designers can create mold geometry much faster using corresponding product geometry. Moreover,

engineers can evaluate the performance and producibility of the components as well as the mold itself by assessing the relative images of alternative designs.

Although initially used only for mold design, drafting, analysis, and programming CNC machine tools, CAD/CAM is currently used also for mold manufacturing process planning, inspection, as well as quality control, machine tool control and monitoring. The merits of using CAD/CAM systems include productivity improvement, quality enhancement, turnaround time improvement, and more effective utilization of scarce resources.

With the increasing demand for capabilities to meet the requirements of complicated mold design and manufacturing activities, CAD/CAM technology has significantly progressed. Four major applications for CAD/CAM in mold design and manufacturing are as follows:

(1) *Design and drafting*: Most CAD tools are highly effective for part and mold design as well as drafting. Several modeling methods are available, including feature-based design, parametric design, and variation design. In feature-based design, a product is constructed, edited, and manipulated in terms of features with certain spatial and functional relationships. Variation design allows a designer to analytically define product geometry and engineering specifications using constraints without concern for the order in which constraints are placed or solved. In addition to offering more geometric oriented and specialized approaches to handle constraints on the geometry, parametric design also allows families of components to be represented as a single parameterized model.

(2) *Analysis*: Tools are available to determine how well the mold works in practice by simulating the flow of a material inside the cavity using analytical models. The resulting flow lines, stresses, temperatures, and forces evaluate the quality of the mold, possibly providing further insight into the life of the mold and/or reasons for failure or wear.

(3) *Inspection and verification*: Conventionally, inspection was performed by skilled technicians. Approval for the mold, or the final part or model is generally determined by producing a finite number of parts and individually inspecting them. Currently, coordinate measurement machines are used to monitor the dimensional quality of the part, or mold cavity, or model; otherwise, an electrode is used to make the mold cavity. With a connection to the CAD/CAM system, the planning of inspection and verification can be performed to significantly improve the effectiveness of product and mold quality control.

(4) *Mold making*: Manufacturing functions provided by CAD/CAM systems that are related to mold making, include NC program generation, machine tool control (either a wire or plunge EDM, or a milling machine), and robotic control. Some systems may provide functions for handling material management (bills of material), estimating the mold fabrication cost, and communicating with other business functions.

4.3. *Mold design process re-engineering*

The proposed concurrent mold design process is based on three principles: (1) a systematic development process, (2) concurrent design, and (3) computer-based design. A systematic development process provides uniform practices for mold development to increase development efficiency and consistency. Concurrent design is referred to herein as all mold life cycle issues are considered and reviewed throughout all phases of the development cycle. The merits of this practice include enhanced quality by defining the requirement earlier and operations involvement, reduced cycle time by minimizing development iterations, and reduced cost through improved producibility as well shorter schedules. Successfully implementing computer-based design would facilitate the practice of (1) and (2) by providing computer-based tools.

Based on these principles, the concurrent mold development model is developed through the following steps: (1) identifying and classifying the mold development activities and tasks into levels of detail, (2) analyzing the relationships among these activities and tasks classified in the previous step, and (3) establishing a mold development process flow based on the interactions between activities and tasks.

Figure 9 shows a matrix that denotes the interactions among the mold development tasks. Tasks in the row and column headers are mold development tasks identified from process capture and characterization. The interactions symbols at the upper-left corners of column-row intersections represent the tasks influenced by the task of the column and are termed influencing symbols. The symbols at the bottom-right of intersections, which denote the tasks influencing the task in the column, are referred to as influenced symbols. A dot circle represents a strong influence, a blank circle indicates a medium influence, and a triangle indicates a weak influence. Weights are assigned to the interaction symbols, such as strong = 5, medium = 3, and weak = 1. For each column, the influence factor is calculated by subtracting the sum of influenced value from the sum of influencing value. A task with a large influence factor is more decisive and thus performed earlier because it influences a great number of other tasks; however, it is itself less influenced by other tasks. According to the interaction matrix, a main process flow, consisting of pre-molding process activities, molding layout, feed system design, cooling system design, venting design and ejector selection, can be identified (see Fig. 10).

To ensure moldability of the product, product moldability is assessed before mold development to check whether the wall thickness and flow length are feasible, whether the part runs on available molding equipment, and what is the anticipated production cost of the product. Product geometry must occasionally be modified to make it moldable. The product geometry is then reconstructed to account for material shrinkage.

Molding layout consists of determining the number of cavities, parting line design, cavity layout, mold base selection, and detail molding design. The number of cavities, i.e. a decisive factor for cavity layout, is determined by the molding size

	Cavity layout	Parting line determination	Sprue design	Gating design	Runner design	Cooling system design	Mold base selection	Ejector selection	Venting design
Cavity layout		⊙ ⊙	○ ⊙			△ △		△ ○	
Parting line determination	⊙ ⊙		○ ○	○ ○				△ ○	
Sprue design	⊙ ○	○ ○		△ △	△ ○	△		△ ○	△
Gating design	○	○ ○	△ △		△	△		△ ○	 △
Runner design	○	△	○ △	△		○		△ ○	
Cooling system design	△ △	△	△	△	○		△ △	△ ○	
Mold base selection	○					△ △		△ △	 △
Ejector selection	○ △	○ △	○ △	○ △	△ △	○ △	△ △		 △
Venting design			 △	△			△	△	
Influence factor	23 10 **13**	16 12 **4**	14 12 **2**	10 8 **-2**	7 9 **-2**	5 9 **-4**	3 5 **-2**	8 13 **-5**	1 3 **-2**

⊙ Strong—5 ○ Medium —3 △ Weak —1

Fig. 9. Interaction matrix of mold development tasks.

and weight, period of delivery, as well as molding machine capacity and budget. As parting line determination and cavity layout are related, they should be performed collaboratively. The parting line design itself includes the operations of parting line specification and undercut detection. Parting line specification and undercut detection should be performed iteratively to ensure product ejectability. Mold base selection is determined primarily by the result of the cavity layout, and is performed after cavity layout. Detailed molding design includes draft, round and fillet design. As they are somewhat independent of each other, they can be done concurrently. Mold base selection and detailed molding design are performed concurrently for the same reason.

Feed system design includes sprue design, gating design and runner design. According to the interaction matrix, the sprue influences the gating design slightly and influences runner design moderately. The influence of the gating design on the runner design is negligible. Consequently, they are performed in the sequence of

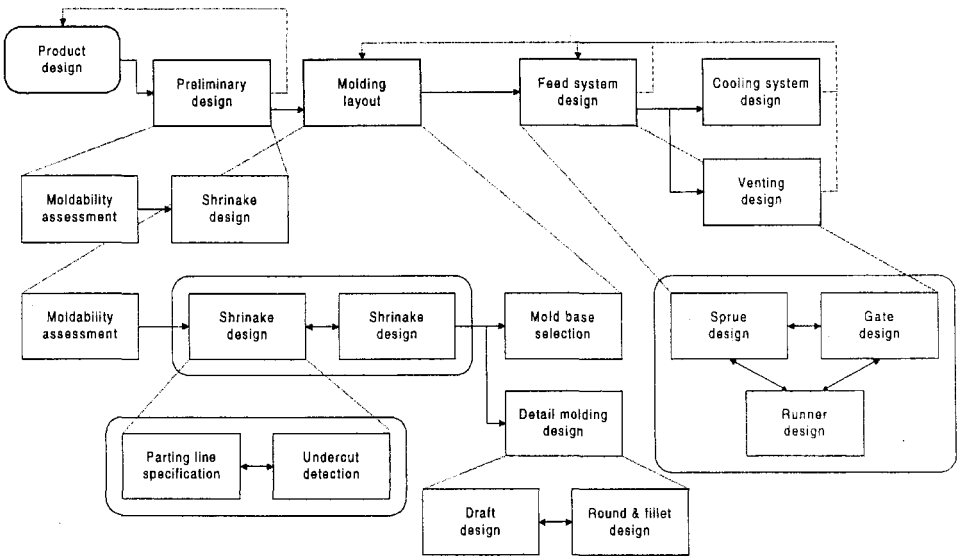


Fig. 10. Concurrent mold development process.

sprue design, gating design, and runner design. However, this sequence may not be necessarily followed because the influences are not strong enough.

The cavities should be arranged around the primary sprue so that each one receives its full and equal share of the total pressure available through its own runner system. As the molding feature design depends largely on the cavity layout, feed system design is performed after cavity layout.

According to the interaction matrix, sprue design, gating design, and runner design influence cooling system design, therefore, they are performed after feed system design.

5. Design of Computer-Aided Concurrent Net Shape Product and Process Development Environment

Due to the numerous and complicated interactions among the concurrent development activities, a computer-based environment must be developed to coordinate the tasks in an integrated manner and to support the decision making at each development stage. In this section, we propose a framework for a computer-aided concurrent engineering environment that can support the concurrent net shape product and process development process discussed in the previous section.

5.1. *Functional requirements identification*

With companies operating at multiple sites and with the increasing emphasis on internationalization, an environment must be created that allows geographically

dispersed team members to work on net shape product and process development activities both interactively and simultaneously. Satisfying this requirement necessitates communication, information and knowledge sharing, managing product and process data, and coordinating development activities across wide-area networks.

Therefore, such an environment should provide (1) tools for net shape product life cycle design and (2) the ability to empower virtual teams by enabling them to collaborate by sharing product data and process knowledge. Functional requirements of this environment are further discussed as follows.

(1) *Advisory tools to design a product through its life cycle*: Concurrent net shape product and process development require simultaneously resolving many product life cycle issues to achieve technically and economically sound product and process designs. Owing to the diversity of team members' expertise, the complexity of task interactions, and the involvement of creativity and intuition, not all development knowledge can be formalized and used in a computer-based system. Therefore, fully automating the practice of concurrent net shape product and process development would be impractical and infeasible. Consequently, it is more reasonable to facilitate the work of teams of professionals, and not to automate the role of some. This can be done by providing development systems with a better net shape product life cycle with well-integrated, special purpose advisory tools and functions.

(2) *Communication and information sharing capabilities*: Concurrent engineering is an integrated and collaborative process, where individuals in different disciplines collaborate to specify, design, and manufacture products through coordination, communication, and negotiation. Ensuring the success of the collaborative process is to completely understand the product and process information that can be shared by all members of the product development team.[37,58] Information shared in a concurrent engineering environment includes design and process knowledge, product data, and process status, which are distributed through various information repositories such as knowledge bases and databases, mail and message libraries, and file and document libraries in the form of text, graphics, images, formulas, CAD data, and multimedia data.[56]

An environment that allows information sharing should provide facilities for team members to (1) archive and exchange ideas, (2) access and send development status, and (3) relate development results to the work of other team members.

(3) *Product data management*: Product data involved in net shape product and process development include product models, tool (i.e., die and molds) models, process plans, product structures, engineering documents, and data files (e.g. IGES files). To facilitate information sharing and development control, product data management is required to provide version and access control for the product data as well as maintain the relationships between the product and related documents. Owing to the differences in representations, structures, and contents of data in different application systems, there is a need to provide an effective means of sharing and exchanging product data among applications and organizations by refining the

product models and maintaining the association between the refined models. This function involves the generation of application-specific geometry, propagation of product requirements and specifications, as well as the establishment and maintenance of the association between application models.

(4) *Project management and process control*: Concurrent net shape product and process development is conducted when a new product must be developed. Members in the development process work collaboratively on a limited project within a consensus time frame, with a common goal and well-defined responsibilities. Therefore, the development process is normally managed as a project.[8] Concurrent net shape product and process development is itself a complicated development process. Therefore, the ability to control and coordinate process activities is required. Since product data are the output of process activities, process control and product data management must be applied in an integrated manner. To properly control the process, mechanisms must be put in place to describe and manage the following: (1) how tasks and operations influence the product data and models, (2) how modifications on a product model affect the related process activities, (3) who should be notified when a product modification occurs, and (4) how to propagate changes to the affected product models.[57]

5.2. *The distributed product and process development system framework*

This section presents the requirements of the system framework for concurrent net shape product and process development. The concept of "allied concurrent engineering" is discussed in detail. A collaborative model is then proposed for allied concurrent net shape product and process development. Based on a collaborative model, a distributed system framework for net shape product and process development can be developed.

5.2.1. *Allied concurrent engineering (ACE)*

With advances in communication and computer network technologies, the "Virtual Corporation" concept[20] has emerged as a promising business strategy to maintain global competitiveness. The practice of "virtual teaming"[17] and "allied concurrent engineering"[15] are also conducted in product and process development.

Allied concurrent engineering is an effective means of integrating resources from different enterprises to overcome the difficulties faced by conventional engineering approaches. Allied concurrent engineering is a distributed and collaborative engineering process, where individuals from different disciplines and different enterprises cooperate to design a product and develop related processes through remote coordination, communication, and control. An allied concurrent engineering process possesses the properties of ***product-centered*** *and* ***dynamic-configurability,***

project-based, flexibility and heterogeneity, hierarchical *and* ***recursive structure, distributed and cooperativeness.***[15]

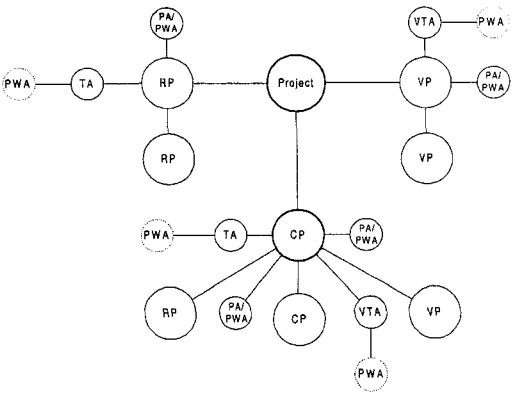
Since allied concurrent engineering is managed on a project basis, such a project may consist of one or more processes, each subsequently formed by one or more activities.

An activity can be a *real activity, collaborative activity* or *virtual activity.* A real activity is conducted by a prime enterprise that owns the process. A collaborative activity is conducted collaboratively by the process owner with one or more allied enterprises. A virtual activity is conducted entirely by one or more allied enterprises. A real activity can be a primitive activity by an individual or a team activity if it is performed by an organized group of individuals. Similarly, a virtual activity can be a virtual primitive or virtual team activity.

Processes are also classified into real processes, collaborative processes and virtual processes. A real process consists of one or more real activities. A collaborative process consists of at least one real activity and at least one collaborative activity or virtual activity. According to the above classification, an allied concurrent engineering project can be defined in terms of a hierarchical structure as depicted in Fig. 11, which reflects the characteristics of virtual enterprising.

5.2.2. *Collaborative system framework*

Based on the hierarchical process model, a concurrent net shape product and process development environment can be established in a top-down layer by layer manner. The system configuration shows the allocation of system components to nodes in a structure that reflects the physical and geographical configuration of the allied



RP - Real process
CP - Collaborative process
VP - Virtual process

RA - Real activity
CA - Collaborative activity
VA - Virtual activity
PA - Primitive activity

TA - Team activity
VPA - Virtual primitive activity
VTA - Virtual team activity
PWA - Personal work area

Fig. 11. A hierarchical structure of concurrent engineering process.

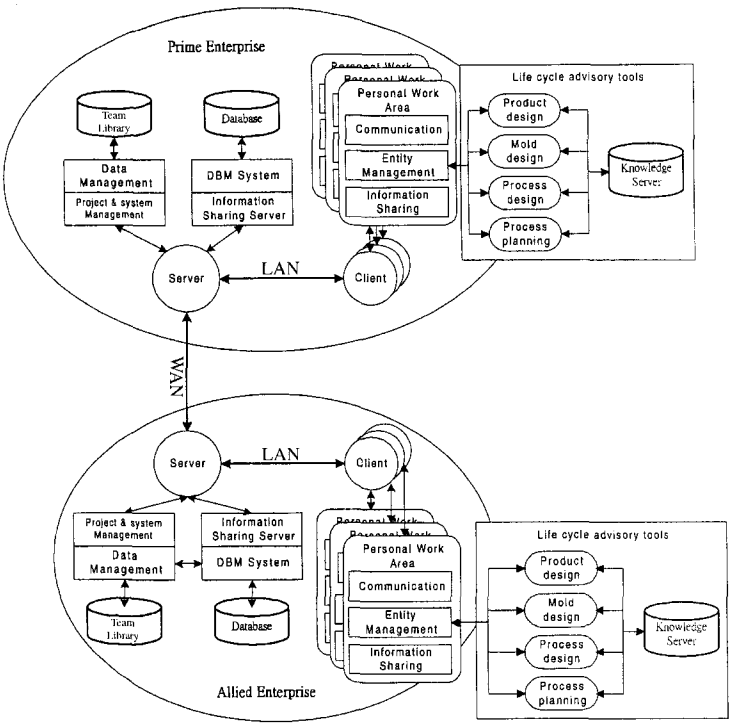


Fig. 12. System configuration for product and process development.

concurrent engineering process. The entire system is configured as levels of a client server structure that reflects the hierarchical structure of the allied concurrent engineering processes (Fig. 12).

In this configuration, a server node contains the components of a data management server, a project and system management server, and an information-sharing server. Each client node is equipped with a personal work area that contains modules for communication, entity management and information sharing as well as advisory tools for product life cycle activities. The outputs of the advisory tools are managed by the entity management module and supported by a knowledge base.

Based on this configuration, we propose an illustrative example of the system framework for a computer-aided concurrent net shape product and process development environment. Figure 13 displays the user's view of the proposed framework, mapped onto a functional system diagram. The diagram shows functions for computer aided concurrent engineering (indicated by rectangular boxes) and two kinds of containers: physical containers (indicated by plain rounded rectangular boxes) and logical containers (indicated by bold boxes with rounded corners). A *physical container* represents a product or process item that can be viewed in the operating

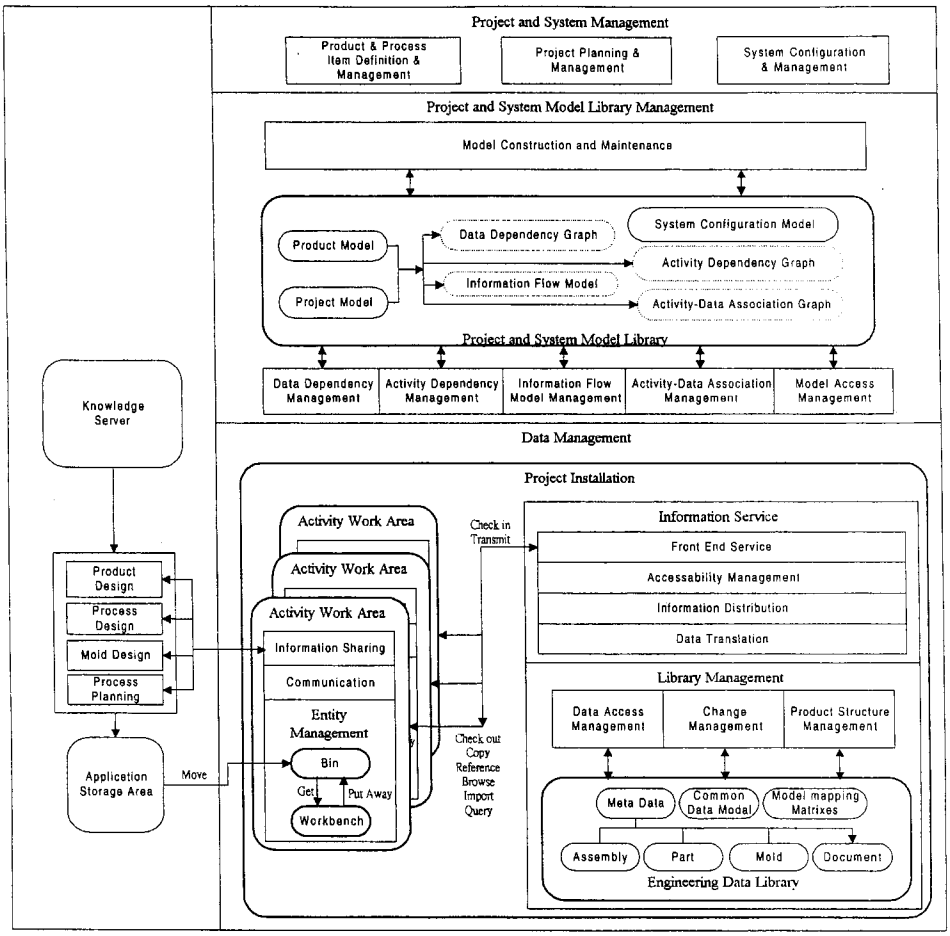


Fig. 13. The framework for a concurrent net shapes product and process development environment.

system when files are listed; a *logical container* represents a conceptual work area or location that does not appear in the operating system.

The functional ingredients of the proposed framework include functions for project and system management, as well as engineering data management as discussed below:

Project and system management

Project and system management provides functions primarily for a project manager to set up and manage an engineering project and track the product and process data items. It includes modules for *product and process item definition and management, project planning and management*, as well as *system configuration and management.*

(1) *Product and process item definition and management*: The mechanisms for product and process item definition are used to create an enhanced product structure by logically defining the components in a product and their related process data. The enhanced product structure can be managed by using management functions such as data item browsing, product structure implosion and explosion, and status viewing.
(2) *Project planning and management*: Regarding the enhanced product structure and using of project planning and management functions, a project leader can define the sequence and schedule process activities, define information flow along the process by associating product and process data items with process activities, and define the project team and authority of each member.
(3) *System configuration and management*: The system configuration and management module provides facilities to configure the overall engineering data management framework based on the process structure. This module also supports functions to specify and interface application systems with the engineering data management system.

Project and system model library management
After product definition and process configuration, a product model and a process configuration are created and maintained in a project and system model library using the *Model Construction and Maintenance* function. To support engineering data management and project management, several models can be derived from the associated product and process model using functions such as *data dependency management, activity dependency management, information flow model management*, and *activity-data association management.* A *system configuration model* is also created and maintained in the library to support system management after system configuration.

Data management
Data management handles engineering data that are shared during engineering process operations. It includes *activity work area service, information service*, and *library management* modules.

An *activity work area service* can be viewed as the activity agent for the project, which provides functions for *information sharing, communication* and *entity management. Information service* provides an effective means of sharing and exchanging product data among applications and organizations so as to meet the requirements of a heterogeneous environment. It may work independently of the implemented information repositories and collaboratively with other information management modules to form distributed information systems. *Library management* provides mechanisms for team library access and maintenance, which are supported by the underlying data access management, product structure management, as well as change management and notification functions.

Levels of logical containers

After project planning and system configuration, a *project and system model library* and *project installation* are established. The former contains engineering process models, project activity models and system configuration models of a project. The latter consists of activity work areas and data repositories such as bin, workbench, and library as the logical work areas for the project.

An *activity work area* is a logical workspace for an activity. An activity work area, which can be used by an individual actor or shared by a team or a virtual team, corresponds to an engineering process activity. Activity work areas are private, where the data within cannot be shared unless it is released to the team library.

A *workbench* is the logical space where a team member performs tasks as part of a development activity. Entity management entails controlling what is on and off the workbench. *Bins* are private storage areas for an activity of the process. A team member working in an activity can organize and control the work by taking things off the workbench and storing them in a bin. Bins allow the storage of a number of types of data. An *engineering data library* is a group storage area managed by library management functions to share data among the members of a development team. Levels of engineering data libraries can be installed to support levels of allied concurrent engineering activities.

The logical containers mentioned above may also contain physical product and process items such as process models, system configuration models, product models, application models, engineering plans, engineering documents, and data files that are shared in concurrent product and process development.

Advisory tools

The environment contains four integrated advisory tools to support concurrent net shape product and process development. They are the product design advisory tool, the process design advisory tool, the mold design advisory tool, and mold manufacturing process planning advisory tool. Each of them can perform on-line design advisory or off-line design assessment concurrently with other related tasks through task coordination and process management.

A knowledge server supports the advisory tools. A knowledge server focuses mainly on supporting the client applications and the coordination of these applications. It contains five knowledge bases: the product design knowledge base, the mold design knowledge base, the process design knowledge base, the process planning knowledge base, and the process coordination knowledge base. Each knowledge base contains several knowledge modules corresponding to the tasks of each design activity. These knowledge modules can work collaboratively and concurrently by using knowledge abstraction and conflict resolution. The function of the coordination knowledge base is two-fold: (a) coordinating the sequence of development activities and (b) resolving conflicts between the activities.

6. Computer Techniques in Development of Design Advisory Tools

Computer-based tools and systems for *information management, computer-mediated communication, process coordination and control*, and *tool integration* are commercially available to support the team-oriented product and process development. Related investigations have developed process-specific or application-specific tools capable of providing application- or process-specific guidance to help meet product life cycle requirements. Most of the tools are analogous to design for manufacturing and design for assembly.[7,11,13,41]

Development of process- or application-specific tools for concurrent net shape product and process development utilizes product and tool design rationales, net shape manufacturing technologies, concurrent engineering methodologies, and computing technologies in an integrated manner.[12,13,39,40] More specifically, it involves activities that use design and manufacturing knowledge, data and computer-based tools to generate product and tool geometry, and to perform engineering analysis. Therefore, an environment capable of supporting computer-aided concurrent net shape product and process development must be able to manipulate knowledge, data, and geometry in an integrated manner as well.

Developing such systems requires analyzing and modeling geometry, data and knowledge involved in the process- and application-specific activities of product and process development, as well as modeling mechanisms to manipulate these elements. This section presents the techniques for development such systems, including geometry modeling, data modeling, knowledge modeling, and system modeling.

6.1. *Knowledge modeling*

Most activities related to net shape product and process development are highly skill-intensive and require a wide variety of design expertises and knowledge of the manufacturing process. Owing to that automatic product and process development are still far beyond current technology, a more reasonable approach would be to provide rules or guidelines to prevent a conflict of the results of each development activity with the development constraints.

Knowledge modeling includes knowledge capturing, knowledge representation, and knowledge abstraction. Knowledge involved in net shape product and process development can be obtained by refining the constraints on the tasks of the concurrent process model developed during the process reengineering stage. Most of the knowledge captured via constraint refinement and analysis consists of universal principles and guidelines without regard to individual shops or machines. Such universal knowledge is normally presented in handbooks and textbooks. However, in industry, design and manufacturing knowledge is largely experience-based and shop- or even machine-specific. Using various methods of collection, e.g. observation, and interviews, and studying factual documents allows us to identify this type of knowledge.

Herein, we use this knowledge by applying a method to transform the data from the above knowledge collection methods into design and manufacturing rules. The proposed method depends on a form of progressive structuring of the descriptions containing the things until short, factual statements are produced. Then, a semantic network is used to generalize the facts to produce the rules. The steps of this method are as follows:

Step 1. Reduce the text into factual sentences in the form noun-verb-noun.
Step 2. Draw a semantic network with the sentences created.
Step 3. Generalize the semantic network by classifying the occurrences.
Step 4. Transform the generalized semantic network into production rules.

Most of the knowledge involved in product and process development can be represented in terms of production rules. However, to facilitate the management of knowledge, the knowledge is classified into a knowledge hierarchy, with knowledge units corresponding to the tasks or activities of the development process. Each knowledge unit in the knowledge hierarchy is represented as a knowledge object, which is an abstraction of the rules behind the knowledge unit. The hierarchical relationships between the levels of knowledge objects are defined as methods in the upper level objects. The lower level objects are instanced by activating corresponding methods in the upper level objects.

Knowledge abstraction attempts to facilitate the management of a massive number of production development rules and to integrate knowledge with data and geometry. The basic idea is to group related rules and conceal the details of these rules by representing them in terms of a concept. The concept can be defined as an object consisting of an object name, attributes, and methods. The object name is the name of the concept, the attributes are the properties of the concept, and the methods define the behavior of the concepts. Objects working towards a common goal can be further defined in terms of a higher level concept.

The knowledge objects in the lower level of the knowledge object hierarchy are used for real assessment or evaluation by running relevant production rules, while the knowledge objects in the higher levels are primarily for planning, monitoring, and controlling the lower level knowledge units via methods or planning rules. In the hierarchy, an activity knowledge object is responsible for planning, activating, and monitoring its subordinate modules. However, in addition to being responsible for planning and monitoring the subordinate tasks, a module object also plays a role in resolving conflicts between tasks. The assessments, evaluations, or planning are performed by the operations—search and evaluation, which activate relevant rules and are closely monitored by task objects.

All rules are stored in a production rule repository and controlled by their corresponding knowledge objects, which can be viewed as the abstraction of those rules. Similarly, these knowledge objects can be grouped and represented in terms of higher-level knowledge objects. The characteristics and behaviors of these knowledge objects are defined as the attributes and methods of these objects, respectively.

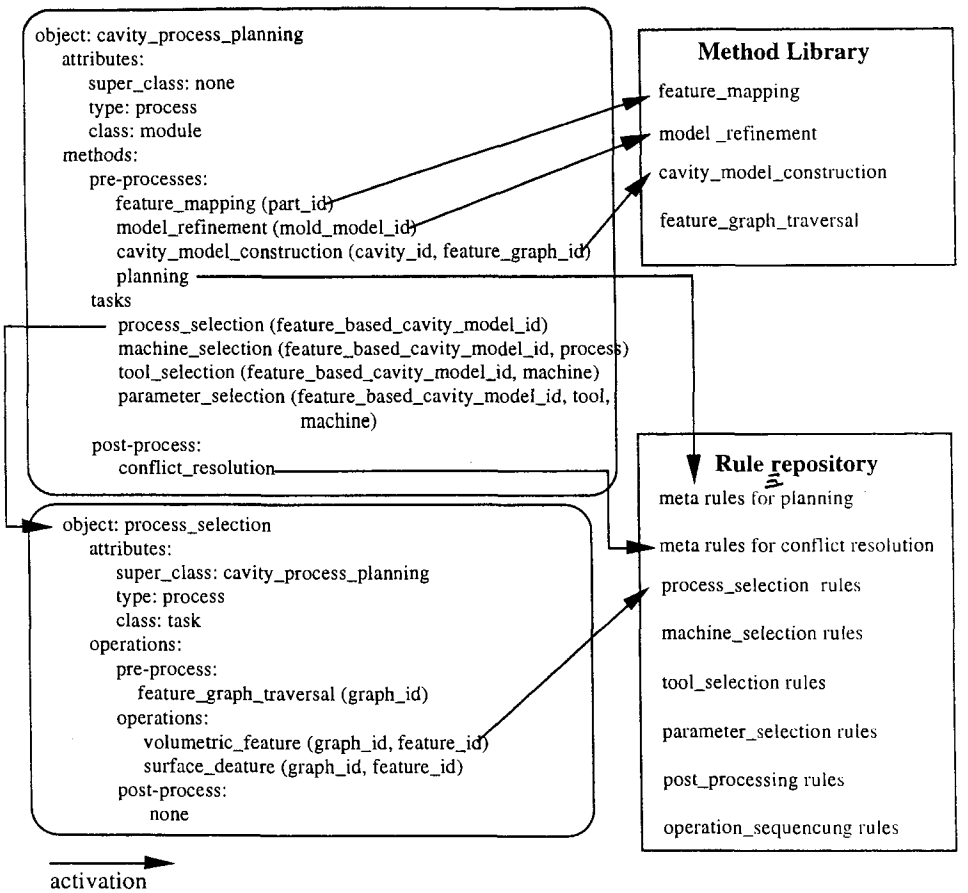


Fig. 14. An example of knowledge abstraction.

Figure 14 presents an illustrative example of knowledge abstraction (Lee *et al.*, 1998). The knowledge for cavity process planning is a knowledge object consisting of an object name, attributes and methods. The attributes define the type and the class of the knowledge object and the methods define the pre-processes, post-processes, as well as the tasks of cavity process planning. The feature_maping, model_refinement, and cavity_model_construction of pre-processes are methods performed by C++ functions and others are performed by production rules. The C++ functions exist in a method library, while the production rules are in the rule repository. Each method of the cavity process-planning object instances a corresponding knowledge object of lower level tasks when it is fired during planning. In this example, a process selection knowledge object is instanced by the process_selection task method. The operation methods of the process selection object activate rules in the rule repository to select a specific feature process.

6.2. *Geometry modeling*

Product and process development can be viewed as a means of creating a manufacturable product model and transforming the product model into a manufacturable mold model. The entire process involves geometry creation and modification. Most of the decision making in product and process development is based on an understanding of the product geometry or tool geometry. Therefore, capabilities to support geometric reasoning on product geometry and tool geometry are also required.

The geometry modeling process involves (1) identifying of the geometric elements involved in product and process development, (2) developing a semantic geometric representation for high-level reasoning, and (3) converting the semantic geometric representation into an object-oriented model.

The elements used to define a product or a tool include geometric entities, entity relationships and technical data. Geometric entities and entity relationships are used to describe the geometric characteristics of a product or a tool. Geometric entities form a hierarchical structure of a product or a tool. Entity relationships are binary relationships used to define the spatial and dependency relationships, or the connectivity between entities. The technical data are basically product or tool functional specifications.

These elements can be identified from the inputs and outputs to and from the tasks of the concurrent process model. A challenge in developing a product or a tool model is to clarify and represent the complex relationships among the geometric entities and diverse types of product and tool definition data. To support decision making based on high-level information about geometric entities and their relationships, product or tool geometric entities and their relationships must be defined in a semantic manner to provide the explicit and unambiguous high-level information required in these applications. As semantic models can explicitly and semantically represent the relationships among data,[22,49] they are employed to conceptually model a product with the following results: (1) the representation of product or tool geometric entities, their properties, relationships, and functions, (2) a wide definition of data and entity relationships, and (3) representation of semantics sufficient for all levels of application.

According to the semantic representations, an object-oriented model is developed using object-oriented techniques. The principles for converting a semantic structure into an object diagram are as follows:

(1) The geometric entities and entity relationships in the semantic structure are represented as objects in the object diagram.
(2) The "part_of" hierarchical relationships in the semantic structure become "aggregate" relationships in the object diagram.
(3) The "is_a" relationships in the semantic structure become "generalization" relationships in the object diagram.
(4) The remaining relationships in the semantic structure become associations in the object diagram.

(5) Product definition data are embedded in geometric entity objects as attributes based on the relationships defined in the semantic product definition data model.

6.3. *Data modeling*

The design of the net shape product and process is governed initially by its intended function and second by the specific limitations of the manufacturing process. The material properties to be used and the engineering aspects of the product design and tool design are added factors. Consequently, designing the net shape product and process requires not only further insight into the selected manufacturing process, product, and tool design, but also a thorough knowledge of material properties and machine capacities. For this reason, in addition to providing product design and process development knowledge, information must also be provided on material properties and machine capacities.

The steps of data modeling include: (1) data identification, (2) data analysis, (3) data classification, and (4) data representation.

Data entities and the attributes involved in product and process development are identified from resources of the concurrent process model and rules identified by knowledge modeling. Data analysis aims to identify the relationships between data entities. The analysis is also performed with the rules derived from knowledge modeling. The entities and their relationships can be represented in an Entity-Relationship (E–R) model,[14,30] where entities are represented as rectangular boxes, with the name of each entity inside the box. Relationships between two or more entities are indicated by a diamond-shaped box, labeled with the names. Lines to the entity boxes for the entities participating in the relationship in question connect each relationship box. Each line is labeled "one" or "many" to indicate whether the relationship is one-to-one or many-to-many.

Most commercially available database systems are relational databases and are effective for storing table-type persistent data. However, they cannot adequately support tasks requiring dynamic reasoning. Developing the E–R model facilitates the implementation of a relational database as a data repository that can provide information for creating data objects. Moreover, the E–R model clarifies object-oriented data model development.

The object-oriented data model is obtained from the conversion of the E–R model, where the data entities in the E–R model become the classes in the object diagram. The entity relationships become the associations between classes. Some of the classes can be further categorized into a class hierarchy.

6.4. *Integration of knowledge, geometry and data*

Based on the object-oriented modeling concept, the relationships between geometric entities and knowledge units can be established by defining methods in geometry

objects for knowledge activation. When the method is activated, a corresponding knowledge object is instanced to perform certain operations on the geometric entity through running rules belonging to the knowledge object. Similarly, data can be closely related to knowledge objects through defining methods for activating relevant data objects from knowledge objects.

A geometric object consists of product definition data attributes and methods for geometric model construction and knowledge object activation. A knowledge object consists of attributes of the knowledge unit itself and methods for data object activation and production rule activation. The attributes defined in data objects are information required by production rules belonging to knowledge objects to which the data object is associated. Methods in data objects are the mechanisms used to retrieve data attributes from database systems.

7. Other Computer Techniques in System Development

The object-oriented technique is widely employed in software development. This technique provides a novel approach to considering problems through models organized around real-world concepts. The fundamental construct is the object, in which data and associated operations that are normally performed on that data are encapsulated, in a single entity. Therefore, instead of passing data to procedures and having these procedures operate on the data, the objects can be invoked to perform operations upon themselves.

This section presents the techniques of object-oriented system design and modeling, as well as the distributed object technology. Both of these techniques are related to the development of a computer-aided concurrent engineering system.

7.1. *Object-oriented system design and modeling*

System modeling defines levels of system details in terms of a set of models. Various object-oriented analyses and modeling techniques are available for system modeling.[4,33–55] Unified Modeling Language[6,32] techniques have emerged in recent years as the notational standard for object-oriented modeling owing to its relative comprehensiveness. Using standard modeling techniques may standardize and facilitate the development process through using common concepts, notations as well as support tools, ultimately increasing compatibility with other software systems.

The system modeling phase includes steps of *use case modeling, class modeling,* and *dynamic modeling* as discussed below.

Use case modeling

Using case modeling is an attempt to (1) capture a system's functional requirements before detailed design work starts and (2) create a seamless transition from the business process to software systems. According to Refs. 33 and 34, a use case describes all of the details about a business process by viewing customers as users

and the business process as a case of how they use the business. It is therefore viewed as "a behaviorally related sequence of interactions performed by an actor in a dialogue with the system to provide some measurable value to the actor."[33,34] *Use cases* represent ways of using a system in terms of sets of scenarios, while *actors* represent roles that have specific sets of responsibilities relating to use cases.

Class modeling
Class modeling identifies the classes involved in engineering information management and their relationships. The results of class modeling are class diagrams that define the static, structural, and data aspects of the system framework in terms of classes and their relationships.

Dynamic modeling
Class modeling examines the static structure of the system by identifying the structure of the objects in the system and their relationships. Dynamic modeling, on the other hand, addresses the dynamic behavior of objects by examining the different events and associated state changes that can happen to an object through different time intervals. It includes the tasks of scenario analysis and state transition diagram development.

A *scenario* is a sequence of particular events that occur during the execution of a system. Arranging the events shown in use case descriptions in a time sequence allows us to perform scenario analysis. In addition, the scenario analysis results are sequence diagrams that show object interactions arranged in a time sequence.

The sequence diagram also indicates events triggering each object and events coming from each object. They can be identified and used to define the state transition diagram of each object class. A state which is an abstraction of the attribute values and links of an object specifies the response of the object to input events. The response to an event depends on the state of the object receiving it and can include a change of state or the sending of another event to the original sender or to a third object. A state diagram that is the link of states through events describes the behavior of a single class of objects. Each event corresponds to a method of the object.

7.2. *Distributed object technology*

Distributed object technology has been used extensively to develop software systems with inter-operability and client-server or multi-tier architecture. A distributed object is packaged as independent pieces of code that can be accessed by remote clients via method invocations regardless of the language and compiler used to create distributed objects.[47] Clients who can be on the same machine or on a machine that sits across a network, do not need to know where the distributed objects reside or on what operating system it is executed.

To develop a computer-aided concurrent net shape product and process development environment, the distributed object technology must be applied. COM/OLE

from Microsoft and CORBA from the OMG (Object management Group) are the two "standards" recognized by the industry. In this section, the concept of CORBA (Common Object Request Broker Architecture) is briefly discussed. Details of distributed object technology can be found in Ref. 47.

CORBA

Common Object Request Broker Architecture (CORBA) not only applies the distributed object technology,[47] but also extends the Client/Server architecture. *Object Request Broker* (ORB), Common *Object Services, Common Facilities*, and *Application/Business Objects*. These are the primary components of the reference model architecture proposed by the Object Management Group (OMG).[47]

The *Object Request Broker* (ORB) was the middleware that established the Client/Server relationships between the objects. Through an ORB, objects communicated with each other by transparently passing messages to or invoking methods on other objects, which could be on the same machine or across a network, without concern for object locations, programming languages, operating systems, or any other system aspects that do not belong to an object's interface.

Common Object Services that are a collection of system-level services packaged as components with predefined interfaces that can be viewed as augmenting the functionality of the ORB. Notable examples of the services include Naming, Events, Lifecycle, Transactions, and Security Services.

Common Facilities are a collection of predefined components that applications can share. Common facilities can be categorized as *horizontal* and *vertical*. The formers are end-user-oriented facilities applicable to most application domains. Examples include *User Interface, Information Management, System management*, and *Task Management* services. The latter provides predefined interfaces for vertical market segments such as health, retail, and finance.

Application/Business Objects are non-standardized application-specific interfaces. Application objects are built on top of services provided by the ORB, Common Facilities, and Object Services.

8. An Example of a Computer-Aided Concurrent Mold Design System

In this section, we present an illustrative example to display a part of the functions of the computer-aided concurrent mold development system.[39]

Step I: Assess the moldability

The mold design process starts by retrieving a part named EXAMPLE_2 (Fig. 15) from a team library. The moldability assessment, which includes feature evaluation and global shape evaluation, is performed on the part to ensure its moldability. Individual feature evaluation is performed on each feature. No individual feature error is issued for the both features (i.e. shell and boss). The global shape evaluation

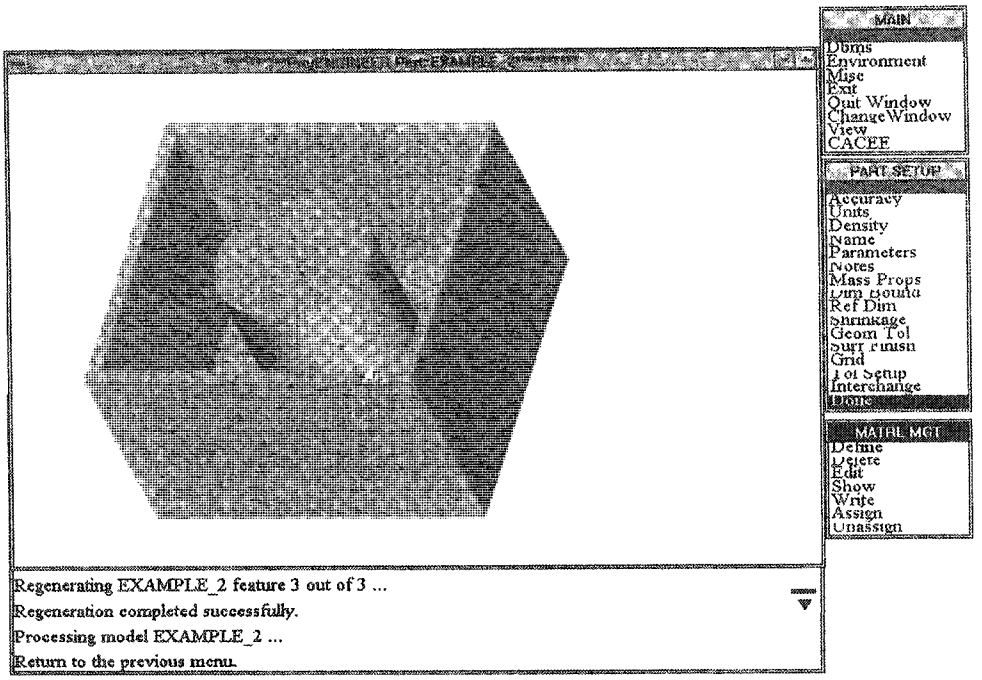


Fig. 15. An example part.

is basically the assessment on part weight and volume. One design error occurs. The part is too heavy. An effective means of reducing the weight is to place a cored hole on the boss. The evaluation is performed again on the wall thickness of the cored hole boss (see Fig. 16). The wall thickness of the cored hole boss is too thin at this time, is modified to place four ribs to strengthen the wall thickness of the cored hold boss.

Step II: Perform shrinkage design

Before starting the molding process, the material shrinkage is considered. The mold designer uses the product model as a reference model for shrinkage design. The designer is allowed to set up isotropic shrinkage for the entire model, or to specify shrinkage coefficients for individual dimensions. With the current system, suggestions are made on specified shrinkage coefficients for selected dimensions, according to the polymer entered by the designer.

Step III: Determine the cavity number

Theoretically, the cavity number depends mainly on the period of delivery, quality control requirements, cost of the moldings, polymer used, shape and dimensions of the molding, and capacity of the injection molding machine. However, in this system, a suggestion is made based on the bounding box of the product (approximate

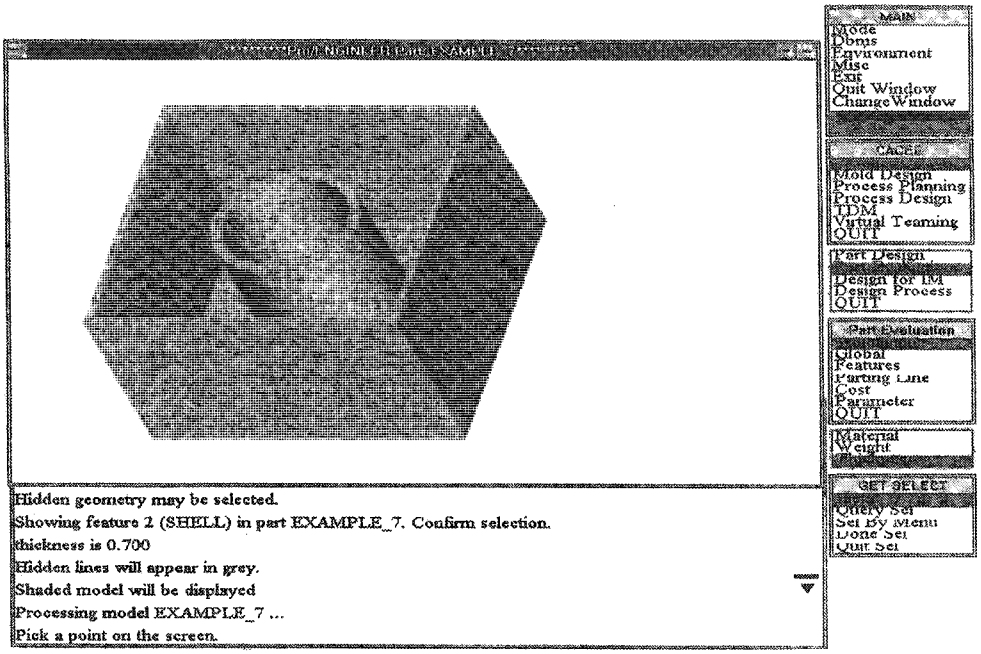


Fig. 16. The example part after moldability assessment.

to product size), the size of workpiece, and the capacity of the injection-molding machine. Therefore, the designer is required to select an injection-molding machine from the machine database and to choose a workpiece from the standard part library. Two-cavity molding is suggested in this example.

Step IV: Specify the parting line

Locations of the parting lines affect the product detail design and mold activity layout, accounting for why it would be more appropriate for the product designer and mold designer to perform the parting line specification collaboratively. The parting line specification starts with the product designer assessing the preliminary product model from teach library. The mold designer may work with product designer to determine the parting lines through remote access and control. Once the parting lines are initially specified, the user must define the mold opening direction. Undercut detection is then performed to ensure the ejectability of the product. At the same time, the mold designer may perform preliminary cavity layout based on the parting lines. A parting line re-specification is required if any undercut occurs or if any problem occurs on cavity layout. In this example, the parting lines are specified along the edges of the bottom face of the shell, and the mold opening directions are both perpendicular to the bottom face.

Step V: Perform the cavity layout

Cavity layout is performed collaboratively with parting line specification. To work on a cavity layout, the mold designer must retrieve a workpiece from the mold base library based on the number of cavities and capacity of the molding machine. The workpiece has standard overall dimensions to fit in a standard mold base, or it can be custom-made to accommodate the geometry of the design model. This product model is used as the reference model for cavity layout. The functions for cavity layout are customized primarily from the functions of Pro/Mold^{TM}. After the initial cavity layout, the knowledge base evaluates the balance of the cavities. Warnings are made if any unbalance occurs. Figure 17 displays the reference part and the cavity layout.

Step VI: Implement the detailed design for injection molding

Once the parting line and mold opening have been determined, the detection of draft faces and round edges are performed on the part. Faces that are parallel to the mold opening direction are considered to be drafted. The user can add drafts onto these faces sequentially, based on the draft angles suggested by the system or

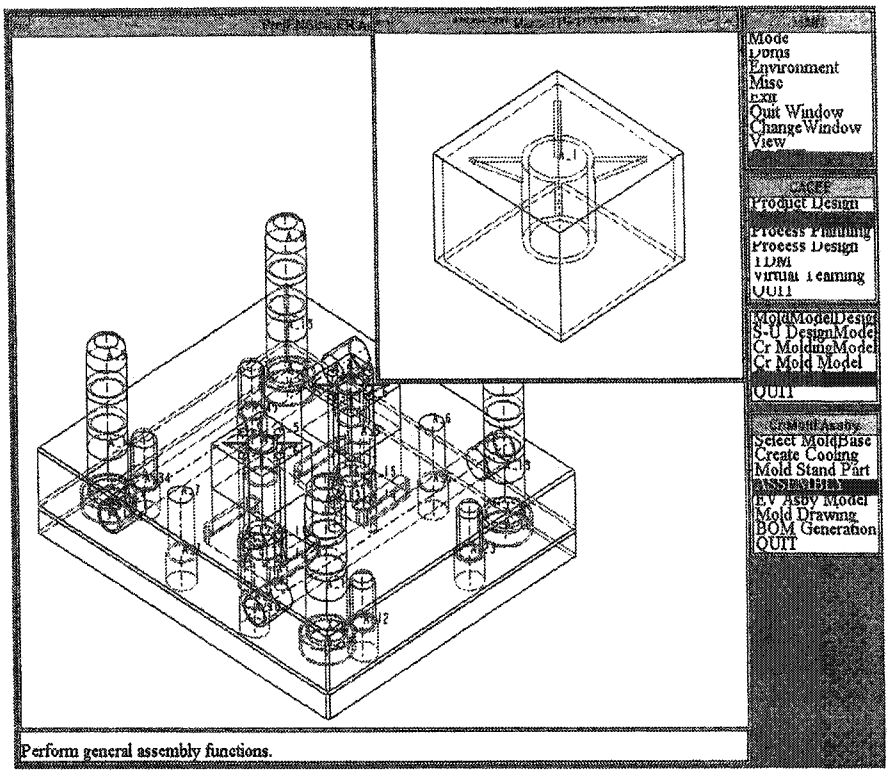


Fig. 17. Reference part and the layout of cavities.

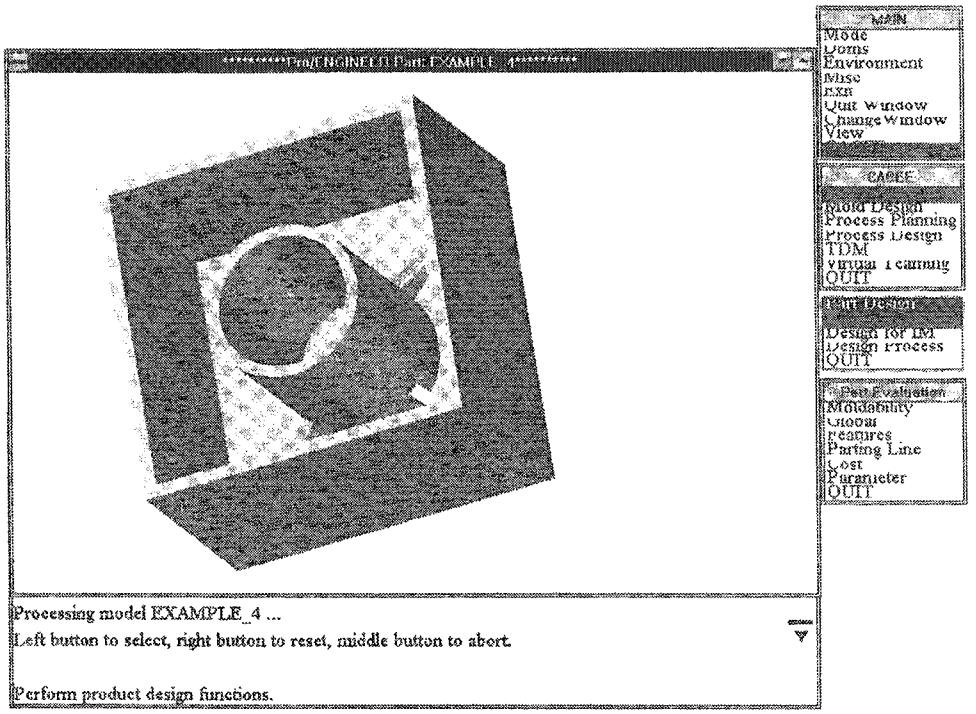


Fig. 18. The final product geometry.

allow the system to add drafts automatically. A round is required for a convex edge, while a fillet is required for a concave edge. A proper radius for the corner or fillet is suggested by the system based on the material used, geometry of the edge, and length of the edge. The edge type and length are derived and computed from the product model. Similarly, the rounding and filleting operations can be performed manually or automatically. Figure 18 depicts the final design of the product.

Step VII: Perform the detailed molding

The modeling detailed design can be performed concurrently with feed system design and cooling system design after molding layout. This design starts by referencing the final product model from the team library and is followed by replacing the preliminary product geometry in the cavity layout with the final product geometry.

After molding detailed design, the designer can split the workpiece by extruding the parting lines sketched on the cavities.

Feed system design, cooling system design, mold flow analysis, and venting design can be performed after or before the workpiece splitting. In the current system, they should be performed manually, without assistance of the knowledge base.

9. Conclusions

This chapter discusses how to apply computer techniques and concurrent engineering methodology in net shape product and process development. Mold design is used as the example for detailed discussion.

This chapter focuses mainly on developing a procedure for a computer-aided concurrent net shape product and process development environment. It includes steps of domain characterization, process reengineering, functional requirement analysis, system design and modeling. Also discussed herein are pertinent technologies and methods that may be applicable in each step.

It is believed that the development of a tool for practical applications should be based on a thorough understanding of the application domain. Moreover, we view the application of computer techniques in concurrent product and process development as a "unified" solution for engineering processes rather than a "point" of system functionality. Therefore, the activities and characteristics of net shape product and process development are first analyzed herein. Concurrent engineering, enterprise modeling and process engineering are adopted to develop a concurrent process for net shape product and process development prior to the design and development of computer-based systems.

In addition to the system development process itself, successful development of a computer-based system depends on the computer technologies employed and methodology innovations for manipulation of the elements in the application domain. In methodologies, the team data management, project management and process control, and information sharing concepts in allied concurrent engineering are introduced. Methodologies of feature-based design and design for X are also presented to develop design advisory tools. Owing to the heavy involvement of knowledge, geometry and data, the mechanisms for integrated manipulation and use of knowledge, geometry and data form the core of a design advisory tool. Therefore, techniques for knowledge modeling, geometric modeling, and data modeling are discussed as well. Since object-oriented technology has been widely employed in software development, UML-based system development methodology and distributed object concept are also discussed to facilitate the system development.

References

1. A. H. S. Al-Ashaab and R. I. M. Young, Design for injection molding in a manufacturing model environment, *Journal of Design and Manufacturing* **5** (1995) 45–54.
2. T. A. Aldowaisan and L. K. Gaafar, A framework for developing technical process reengineering designs, *Computers and Industrial Engineering* **32**, 3 (1997) 657–670.
3. T. Altan, Advances in metal forming processes, *Robotics and Computer Integrated Manufacturing* **4**, 1/2 (1988) 121–128.
4. P. Allen and S. Frost, *Component-Based Development for Enterprise Systems* (Cambridge: Cambridge University Press, UK, 1998).

5. P. Bernus and L. Nemes, Enterprise integration—engineering tools for designing enterprise, in *Modeling and Methodologies for Enterprise Integration*, eds. P. Bernus and L. Nemes (London: Chapman and Hall, 1996).
6. G. Booch, J. Rumbaugh and I. Jacobson, *The Unified Modeling Language User Guide* (Reading, MA, Addison-Wesley, 1999).
7. G., Boothroyd, P. Dewhurst and W. Knight, *Product Design for Manufacture and Assembly* (New York, Marcel Dekker, Inc, 1994).
8. M. R. Cantor, *Object-Oriented Project Management* (John Wiley and Sons, Inc., New York, 1998).
9. D. E. Carter and B. S. Baker, *Concurrent Engineering: The Product Development Environment for the 1990s* (Addison-Wesley Publishing Company, Inc., New York, USA, 1992).
10. C. Charney, *Time To Market: Reducing Product Lead Time* (Society of Manufacturing Engineers, Dearborn, MI, USA, 1991).
11. Y. M. Chen, R. A. Miller and K. Sevenler, Knowledge-based manufacturability assessment: an object-oriented approach, *Journal of Intelligent Manufacturing* **6** (1995) 321–337.
12. Y. M. Chen, Development of a concurrent net shape product and process development environment, *International Journal of Robotics and Computer Integrated Manufacturing* **13**, 4 (1997) 337–360.
13. Chen Yuh-Min and Wei Ching-Ling, Computer-aided feature based design for net shape manufacturing, *Journal of Computer Integrated Manufacturing Systems* **10**, 2 (1997) 147–164.
14. P. P. S. Chen, The entity-relationship model: toward a unified view of data, *ACM Transactions on Database Systems* **1**, 1 (1976) 9–36.
15. Chen Yuh-Min and Liang Mind-Wu, Design and implementation of a collaborative engineering information system for allied concurrent engineering, *International Journal of Computer Integrated Manufacturing* **17** in press.
16. D. Clausing, *Total Quality Development: A Step-by-Step Guide to World Class Concurrent Engineering* (New York: ASME Press, 1994).
17. K. J. Cleetus, Virtual Team Framework and Support Technology. CERC *Technical Report*, ERC-TR-RN-92-016. Concurrent Engineering Research Center, West Virginia University, 1993.
18. J. S. Colton, An intelligent design for manufacturing system, *Concurrent Engineering: Automation, Tools, and Techniques*, eds. Andrew Kusiak (New York: John Wiley and Sons, Inc., USA, 1993).
19. B. Curtis, M. Lellner, and J. Over, Process modeling, *Communication of the ACM* **35**, 9 (1992) 75–90.
20. W. H. Davidow and M. S. Malone, *The Virtual Corporation* (Harper Collins Publishers, USA, 1992).
21. Z. Dong, Design for automated manufacturing, *Concurrent Engineering: Automation, Tools, and Techniques*, ed. Andrew Kusiak (New York: John Wiley and Sons, Inc, 1993).
22. C. M. Eastman, The Construction of Data Modeling to the Future Development of CAD/CAM Databases, ASME Computers In Engineering Conference, 1991.
23. ERC, *Fourth Year Progress Report and Renewal Proposal*, Engineering Research Center for Net Shape Manufacturing, The Ohio State University, 1990.
24. M. Fox, Issues in enterprise modeling, *Proceedings of the IEEE Conference on Systems, Man, and Cybernetic*, Le Touquet, France, 1993.

25. V. Gruhn, Business process modeling and workflow management, *International Journal of Cooperative Information Systems* **4**, 2 & 3 (1995) 145–164.
26. B. Gerdes, Concurrent engineering approach to injection mold design and fabrication, *ANTEC* (1994) 1111–1116.
27. J. R. Hartley, *Concurrent Engineering: Shortening Lead-Time, Raising Quality, and Lowering Costs* (Cambridge, Massachusetts: Productivity Press, 1990).
28. K. Himasekhar, J. Lottey and K. K. Wang, Design of a cooling system in injection molding: use of CAD tool, *ANTEC* (1990) 1103–1106.
29. Y. J. Huh and S. G. Kim, A knowledge-based CAD system for concurrent product design in injection molding, *Computer-Integrated Manufacturing* **4**, 4 (1991) 209–218.
30. R. Hull and R. King, Semantic database modeling: survey, applications and research issues, *ACM Computing Surveys* **19**, 3 (1987) 201–259.
31. R. K. Irani, B. H. Kim and J. R. Dixon, Automatic injection mold gating design system, *ANTEC* (1989) 1265–1269.
32. I. Jacobson, G. Booch and J. Rumbaugh, *The Unified Software Development Process* (Reading, MA: Addison-Wesley, 1999).
33. I. Jacobson, M. Christerson, P. M. Jonsson and G. Overgaard, *Object Oriented Software Engineering: A Use Case Driven Approach* (Reading, MA: Addison-Wesley, 1992).
34. I. Jacobson, M. Ericcson and A. Jacobson, *The Object-Advantage-Business Process Reengineering with Object Technology* (Reading, MA: Addison-Wesley, 1994).
35. H. H. Johansson, P. Mchugh, A. J. Pendlebury and W. A. Wheeler, *Business Process Reengineering* (John Wiley and Sons, New York, 1993).
36. W. R. Jong and K. K. Wang, Automatic and optimal design of runner system in injection molding based on flow simulation, *ANTEC* (1990) 385–389.
37. R. Karinthi, V. Jaganna, V. Montan, J. Petro, R. Raman and G. Trapp, Promoting concurrent engineering through information sharing, *Proceedings of ASME Winter Annual Meeting*, Anaheim, California, 8–13 November, 1992.
38. A. T. N. Kusiak, T. N. Larson and J. Wang, Reengineering of design and manufacturing processes, *Computers and Industrial Engineering* **26**, 3 (1994) 521–536.
39. R. S. Lee, Y. M. Chen and C. C. Lee, Development of a concurrent mold design system: a knowledge based approach, *Journal of Computer Integrated Manufacturing Systems* **10**, 4 (1997) 287–307.
40. R. S. Lee, Y. M. Chen, H. Y. Cheng and M. D. Kuo, A framework of a concurrent process planning system for mold manufacturing, *Journal of Computer Integrated Manufacturing Systems* **11**, 3 (1998) 171–190.
41. S. Y. Liou and R. A. Miller, Design for die-casting, *International Journal of Computer Integrated Manufacturing* **4**, 2 (1991) 83–96.
42. R. J. Mayer, M. K. Paintec and P. S. Dewitte, IDEF family of method for concurrent engineering and business re-engineering applications, Technical Report, Knowledge Based Systems, Inc, 1994.
43. K. G. McIntosh, *Engineering Data Management: A Guide to Successful Implementation* (New York, McGraw-Hill Book Company, 1995).
44. L. C. G. Miller, *Concurrent Engineering Design* (Dearborn, Michigan, Society of Manufacturing Engineers, 1993).
45. R. Mills, B. Beckert and L. Carrabine, The future of product development, *Computer Aided Engineering*, October (1991) 38–46.
46. E. Molloy and J. Browne, A knowledge-based approach to design for manufacturing using features, *Concurrent Engineering*, eds. Hamid R. Parsaei and G. William (Sullivan, New York, Chapman and Hall, 1993).

47. R. Orfali, D. Harkey and J. Edwards, *The Essential Distributed Objects: Survival Guide* (New York, John Wiley and Sons, Inc, 1996).
48. I. Pandelidis and Q. Zou, Optimization of injection molding design. Part I: gate location optimization, *Polymer Engineering and Science* **30**, 15 (1990) 873–882.
49. J. Peckham and F. Maryanski, Semantic data model, *ACM Computing Surveys* **20**, 3 (1988) 153–189.
50. J. P. Pennell and R. I. Winner, Concurrent engineering: practices and prospects, *Proceedings IEEE Global Telecommunications Conference and Exhibition (GLOBCOM '89), Part I* (IEEE Service Center, Piscataway, NJ, USA, 1989) 647–655.
51. B. Prasad, *Concurrent Engineering Fundamentals, Volume I: Integrated Product and Process Organization* (New Jersey, Prentice Hall PTR, 1996).
52. B. Prasad, *Concurrent Engineering Fundamentals, Volume II: Integrated Product Development* (New Jersey, Prentice Hall PTR, 1997).
53. R. G. W. Pye, *Injection Mold Design*, 4th ed (The Plastics and Rubber Institute, 1989).
54. D. V. Rosato, and P. E. Rosato, *Injection Molding Handbook* (New York, Van Nostrand Reinhold Company, 1986).
55. J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy and W. Lorensen, *Object-Oriented Modeling and Design* (New Jersey, Prentice Hall, 1992).
56. K. Srinivas, R. Reddy, A. Babadi, S. Kamana, V. Kumar and D. Z. Zhao, MONET: A multi-media system for conferencing and application sharing in distributed systems, *CERC Technical Report*, CERC-TR-RN-91-009. Concurrent Engineering Research Center, West Virginia University, 1992.
57. D. Sriram, R. Logcher, A. Wong and S. Ahmed, Computer-aided cooperative product development: a case study, *International Journal of Systems Automation: Research and Applications (SARA)* **1** (1991) 89–112.
58. G. Trapp, Sharing information: a CALS/CITIS, concurrent engineering and PDES/STEP synergy, *CERC Technical Report*, CERC-TR-TM-91-011, Concurrent Engineering Research Center, West Virginia University, 1991.
59. F. B. Vernadat, *Enterprise Modeling and Integration* (London, Chapman and Hall, 1996).

CHAPTER 3

THE APPLICATION OF COMPUTER MODELLING TO THE PROCESS OF MANUFACTURING STRATEGY FORMULATION

D. K. HARRISON and T. S. BAINES

Caledonian University
City Campus
Cowcaddens Road
Glasgow G4 OBA

A successful manufacturing industry can make a significant contribution to the prosperity of a nation. For companies to consistently realize success in manufacturing, they must have an effective manufacturing strategy. An effective manufacturing strategy is realized through a formal planning process which takes proper account of all relevant factors. This chapter provides a uniquely broad view in developing a manufacturing strategy.

Keywords: Computer modeling; manufacturing strategy; formal planning processes.

A successful manufacturing industry can make a significant contribution to the prosperity of a nation. For a manufacturing company to consistently realize success invariably requires the organization to seek and achieve congruence between internal manufacturing capabilities and external market and financial environments. This approach to organization design is often expressed as a manufacturing strategy, and there is a close association between the existence of an intended manufacturing strategy within a business and prosperity.

A manufacturing strategy can be formed by a number of methods, but a particularly successful approach is practising managers being guided through strategy formulation by a formal planning process. Usually, such a process is a sequence of activities that secure recognition of a company's existing manufacturing capabilities, structure an expression of the associated financial and market environments, and stimulate the evolution of a sequence of actions to overcome any deficits that may exist.

During manufacturing strategy formulation it is usual to evaluate the effect of proposed actions on the capabilities of the manufacturing system under consideration. Such evaluation can be made through judgement of individual personnel, refined through bargaining between a number of personnel, and supported by analytical methods. One such analytical method is modelling. A model can be created of a manufacturing system, a number of modifications can be made to the model to reflect the strategy under consideration, and the ensuing model behavior treated as a prediction of future manufacturing capabilities.

Modelling is often used in detailed design of manufacturing systems. However, manufacturing strategy formulation is different from detailed manufacturing system design, and hence demands specific characteristics of a modelling approach. Therefore, to promote the application of the manufacturing strategy concept, this chapter investigates modelling in the evaluation of a manufacturing strategy.

This chapter provides a uniquely broad review of modelling techniques that can be used for manufacturing strategy evaluation. Section 1 explores the concept of manufacturing strategy, and hence sets the context for the modelling techniques discussed in this chapter. Section 2 builds on this foundation, and describes the modelling challenge from the perspective of an industrial practitioner. Section 3 forms a taxonomy of models, and uses this to illustrate the wide range of modelling techniques available to the practitioner. Finally, Sec. 4 explores the suitability of these techniques to manufacturing strategy evaluation.

1. The Challenge of Manufacturing Strategy Formulation

This chapter is concerned with a unique challenge for computer modelling, namely how to help in the process of manufacturing strategy formulation. To understand this challenge, it is necessary to explain the meaning of manufacturing strategy, and to explore how it can be formed.

1.1. *The concept of manufacturing strategy*

Manufacturing strategy is a concept; it is a general notion about the organization of a company's manufacturing activity. The word "strategy" has a Greek origin from around 550 BC. Initially, the word referred to a role, for example, a General, and later came to mean "the art of the General".[30] More recently, Chandler,[16] whilst discussing the planning and growth of an organization, is generally accredited with probably the first definition of strategy in business. Chandler saw strategy as:

"... the determination of the basic long-term goals and the objectives of an enterprise, and the adoption of courses of action and the allocation of resources necessary for carrying out these goals."

Later, Ansoff,[3] in a business context, identified strategy as:

"Strategy guides and directs a firm's growth and change."

Skinner[76] is seen by many as being the first to introduce the concept of manufacturing strategy. Skinner however, actually gives this credit to McLean in 1946, along with Miller and Rogers a decade later. According to Skinner, McLean observed that a number of companies may compete within an industry using entirely different approaches to manufacturing management. In this work, Skinner refers to strategy as:

"... a set of plans and policies by which a company aims to gain advantage over its competitors."

The contribution of researchers such as Chandler and Skinner, amongst many others, can be clarified through viewing strategies at three tiers in an organization. Hayes and Wheelwright,[39] define these levels as a hierarchical structure. The roles that each strategy takes are summarized as:

(1) Corporate strategy: Definition of the businesses in which a corporation will participate, and the acquisition and allocation of key corporate resources to each of those businesses.
(2) Business strategy: The basis on which a business unit will achieve and maintain competitive advantage, in a way that links the strategy of the business to that of the corporation as a whole.
(3) Functional strategies: Providing support to the competitive advantage being sought by the business strategy.

Although the form of functional strategies other than manufacturing are outside the scope of this chapter, a brief insight assists in setting the context of manufacturing strategy. Functional strategies can be coarsely grouped into marketing, financial and manufacturing. Each of these strategies will have goals associated with their function. For example, the goals of a financial strategy can include Return On Investment (ROI) and profitability measures, while marketing goals include market share and growth. The functional strategies combine to form the basis of a company's business strategy.

Since Skinner first promoted manufacturing strategy within an organization, there have been numerous attempts to give a fuller and more precise definition of this specific functional strategy. Some differences in definitions appear to be semantic, whilst other definitions represent a real alternative emphasis. There is however a general agreement in the literature that manufacturing strategy has a long range thrust, and that there should be some competitive advantages defined. Platts[65] provides a useful definition of strategy, that incorporates the views of many authors, namely a manufacturing strategy is:

"A pattern of decisions, both structural and infrastructural, which determine the capability of a manufacturing system and specify how it will operate in order to meet a set of manufacturing objectives which are consistent with overall business objectives."

This definition of manufacturing strategy mentions "manufacturing objectives", and decisions about the "structure" and "infrastructure" of a manufacturing system. The specifics of what a strategy contains in each of these areas is generally referred to as the "content". The content focuses on the specifics of what was decided, whereas "process" addresses how strategic decisions are reached. To develop an appreciation of the nature of manufacturing strategy it is necessary to explore the generally accepted forms of content.

The manufacturing task, referred to by Platts as the manufacturing objectives, is a statement of what the manufacturing function must accomplish. Platts defines the manufacturing objectives in terms of "competitive criteria", as shown in Table 1. This is similar to New[58] who uses "competitive edge criteria"; Hill,[42] who uses "order winning and order qualifying criteria"; and Minor *et al.*[54] who use "competitive priorities". In each case these criteria are used to assess the contribution that a manufacturing activity makes to the saleability of a product.

The span of changes to a company that a manufacturing strategy is generally accepted to address is broad. As Skinner[77] points out:

"The entire factory must be planned and renovated as a unit lest any one element undermine the entire structure."

Likewise, Buffa[13] considers that:

"All the activities in the line of material flow — from suppliers through fabrication and assembly and culminating in product distribution — must be integrated for manufacturing strategy formulation."

Such changes can be grouped as either structural or infrastructural. Furthermore, the notion of "policy areas" can be applied to provide a detailed categorization of structural or infrastructural changes (Table 2). Decisions concerning policy areas and competitive criteria have mutual implications and constraints. There have to be trade-offs in a manufacturing strategy. Compromises are necessary in the goals and decision areas of a production system.

In conclusion, the content of a manufacturing strategy can be viewed in terms of changes to the structure and infrastructure of a company, made with the intention of

Table 1. *Competitive criteria (Source: Platts, 1990).*

Criteria	Function
Product features	Adding capability to the product, or choice to the customer.
Quality	Producing a product that performs well to specification.
Delivery lead time	Delivering the product within a short lead time.
Delivery reliability	Always delivering on schedule.
Design flexibility	Having the ability to produce products to customer specification.
Volume flexibility	Having the ability to supply fluctuating volumes without compromising lead time.
Price	Selling at the lowest price.

Table 2. Policy areas (Source: Platts, 1990).

Policy Areas	Description
Facilities	The factories, their number, size, location, focus.
Capacity	The maximum output of the factory.
Span of process	The degree of vertical integration.
Processes	The transformation processes (metal cutting, mixing, assembly, etc.) and most critically the way in which they are organized.
Human resources	All the people-related factors, including both the personal and the organizational level.
Quality	The means of ensuring that product, process and people operate to specification.
Control policies	The control policies and philosophies of manufacture.
Suppliers	The methods of obtaining input materials at the right time, price and quality.
New products	The mechanisms for coping with new product introduction, including links to design.

fulfilling manufacturing objectives. The manufacturing objectives can be categorized in terms of competitive criteria, and are focused at forming links between functional strategies. Changes to a company's structure and infrastructure, associated with manufacturing strategy, can be broadly categorized into policy areas. However, there are inevitable trade-offs that have to be made in decisions about competitive criteria, policy areas, and time schedule applied to realize the associated manufacturing capabilities.

1.2. *Manufacturing strategy formulation process*

The objective of this section is to establish how a manufacturing strategy can be formed in an organization. This section explores the meaning of process.

1.2.1. *Strategy formation*

To fully appreciate the mechanisms of strategy formation it is necessary to consider a broad definition of strategy; as is given by Mintzberg[55] who defines strategy as:

"...a pattern in a stream of decisions."

Mintzberg[55] argues that such a definition enables an appreciation that a strategy maker may formulate a strategy through a conscious process before making a specific decision, or a strategy may form gradually, perhaps unintentionally as decisions are made one by one. A similar view of strategy forms is implied by authors such as Ansoff,[3] from a business strategy perspective, who saw that strategic change takes place in most firms, with or without explicit strategy formulation by management. Porter[67] agrees and says that every firm has a competitive strategy which has either been developed explicitly through a planning process or it may have evolved implicitly through the activities of various functional departments.

Mintzberg[55] specifies this distinction between conscious and unintended actions more precisely. He identifies that a "realized" strategy may have "intended" or "emergent" origins, and that an emergent strategy can be observed when a non-intentional pattern can be recognized in past actions. Mintzberg[57] also reasons that even when a strategy is deliberately intended, the resulting real-world strategy is often, because of influences such as learning, a mix of intended and emergent strategies. Quinn[69] reinforces this view and argues that there is likely to be a significant difference between an intended strategy and a realized strategy because strategy deals with "unknowable" factors.

The action of consciously forming a strategy can only be associated with an intended strategy and is termed formulation. In this sense strategy formulation can be thought of as a subset of strategy formation. Likewise, Mintzberg[57] argues that only with an intended strategy does the distinction exist between strategy formulation and implementation, along with the notion of tactical actions, and a potential for dislocation of tactical and strategic thought. Tactics are short duration, adaptive, action-interaction re-alignments used to accomplish limited goal.

1.2.2. *Strategy formulation*

The depth of consideration given to the content of a strategy can vary in strategy formulation. At one extreme, strategy formulation may be achieved through strategy formulators applying a relatively shallow decision making process with the resulting strategy content being largely adopted — this may be termed a prescriptive or generic strategy. Alternatively, a very detailed, full and lengthy consideration of strategy content may be performed. This second case is more usually associated with formal planning processes.

The role of formal planning processes in strategy formulation, particularly in a corporate context, appears to be open to contention. Andrews[2] suggests that a common form of strategy formulation is an individual executive responding to environmental pressure, competitive threat, or environmental opportunity. Hofer and Schendel[43] stress that formal planning systems are not always required for effective strategy formulation. Likewise, Mintzberg[57] sees that most successful strategies have been based to a large extent on formulation through mental synthesis. He argues that the appropriate process of strategy formulation is based on forms of synthesis, with formal planning processes supporting this role.

1.2.3. *Formal planning process*

A process is applied through a mechanism of company based meetings of personnel, typically orchestrated by a number of worksheets. Similar and associated mechanisms are offered by, for example, Hill,[42] Fine and Hax,[31] Pendlebury,[62] and DTI.[27] Generally, the application of formal planning processes is intended to follow, though not necessarily procedurally, a number of stages. Hofer and Schendel[43] identified

seven stages that are included implicitly or explicitly in major strategy formulation processes, these are:

(1) Strategy identification: Assessment of current strategy.
(2) Environment analysis: Identification of opportunities and threats.
(3) Resource analysis: Assessment of principal skills and resources available.
(4) Gap analysis: Comparison of the organization's objectives, strategy and resource against the environment opportunities and threats to determine the extent of change required in the current strategy.
(5) Strategic alternatives: Identification of the options upon which a new strategy may be built.
(6) Strategy evaluation: Evaluation of the strategic options to identify those that best meet the values and objectives of all stakeholders, taking into account the environmental opportunities and threats and the resources available.
(7) Strategic choice: Selection of the options for implementation.

Each of these stages may be addressed differently, depending on whether an internally or externally supportive process is being applied.

Hofer and Schendel[43] see that stages 1–4 (inclusive) are concerned with establishing the value of competitive factors external to an organization and contrasting these against internal manufacturing performance. The scope of this analysis is extended to explore opportunities and threats that may drastically alter the environment within which a company is competing. The anticipated gap between external factors and internal manufacturing performance, forms a platform on which strategy formulation can commence. The objective of the ensuing formulation activity is to create a strategy that achieves congruence between internal and external factors.

Hofer and Schendel associate stage 5 with the identification of strategic alternatives. This activity is where formal planning and synthesis processes converge. The literature generally agrees that strategy formulation will take into account factors other than those presented in formal planning. However, information from earlier stages, through a mechanism such as a manufacturing audit, can stimulate and guide this activity by providing an analytical base from which new strategies can be synthesized. The outcome of this stage is a number of alternative manufacturing strategies.

The motive at stage 6 is seen by Hofer and Schendel as establishing a ranking in the suitability of strategies, such that stage 7, strategy choice, can take place. Stage 6 is concerned with assessing the impact, in terms of manufacturing objectives, of proposed strategic alternatives. It is important to note that this activity seeks to establish how a strategy will effect a systems performance, and is distinct from evaluating the success of a strategy that has been applied.

Hofer and Schendel consider that stage 7 is concerned with strategy choice and is intertwined with the previous stage of strategy evaluation. If strategy evaluation is comprehensive and complete, such that all appropriate factors are considered, then the preferred strategy should be obvious. Mintzberg *et al.*[56] argue that the evaluation-choice routine may be considered in three modes, namely:

(1) Judgement; one individual makes a choice in his own mind with procedures that he does not, perhaps cannot, explain.
(2) Bargaining; selection is made by a group of decision makers with conflicting goal systems, each exercising judgement.
(3) Analysis; factual evaluation is carried out, generally by technocrats, followed by management choice by judgement or bargaining.

It is possible to extend the evaluation-choice routine to feedback into the activity of generating strategic alternatives. This could occur when the new knowledge gained from strategy evaluation is used to stimulate idea generation or refinement of strategic options, analysis supporting a debate and discussion between decision makers through providing assessment of various alternatives, and providing a starting point for the generation of new alternatives.

1.3. *The modelling challenge*

In summery, an explicit strategy could be produced by mental synthesis alone, but it appears that formal planning processes offer significant support in this role. Such processes have seven common steps. These steps span from the assessment of the current strategy of an organization, to choosing the most appropriate strategy for implementation. A crucial step in formulation is strategy evaluation prior to implementation. It is this assessment activity where modelling can play a valuable role, and hence, this is the focus of this chapter. There is a need to be clear about the specific form of the modelling considered here.

There is a wide variety of work of an apparently analytical nature. Some work is concerned with pseudo-analysis tools that focus on providing structured enquiry, judgement and problem solving about a real world object or system under study. These tools are occasionally referred to as models, though this chapter will use the term "methodology". This somewhat pedantic step is necessary to enforce a distinction from "models" that are an abstract representation and emulation of a real system.

Many tools that can be termed methodologies appear to have originated from corporate and business planning literature, for example, the "Boston Consulting Group, growth share matrix"[67]; the "General Electric McKinsey, industry attractiveness-business strength matrix"[40]; and the "product/market evolution portfolio matrix".[81] An appropriate term for these particular methods is considered to be "traditional strategy tools". Traditional strategy tools can provide assistance in overall formulation as well as specific strategy evaluation. For example,

"product life cycle" diagrams can aid in the identification of product families and in focusing discussions about evolution of product sales. Other tools can directly assist in evaluating the effect of a manufacturing strategy, for example, "price of non-competitiveness matrix", "learning curve" and "product-process matrix".

The potential value of modelling appears to be high, for example, Copacino and Rosenfield[22] see that decision support models are useful both for measuring the impact of proposed plans, as well as for determining the most efficient way to support the corporate plan. Likewise, as previously introduced, Ansoff and Brandenburg[4] advocate a modelling approach. The preference for models appears to have occurred as they provide both prediction about, and insight into, the behavior of a manufacturing system.

2. Understanding the Requirements of Modelling in Manufacturing Strategy Evaluation

This section focuses on the role of a model and modelling tool in manufacturing strategy evaluation. This role is considered from a practitioners perspective, and four categories of requirements of modelling are identified.

2.1. *Establishing the general requirements of modelling*

An appropriate starting point is to consider how a model and modelling tool could operate in practice within the general concept of manufacturing strategy. A suitable modelling approach for manufacturing strategy evaluation would commence with a modelling tool that will allow a manufacturing system to be represented by a model. Such a model should be capable of modification to reflect the implementation of a manufacturing strategy. The resulting model behavior could then be assessed against the performance measures that are appropriate to manufacturing strategy formulation.

From this description of model operation, a number of requirements of a model become apparent. Firstly, some model flexibility is necessary in order to accommodate the range of manufacturing system developments that are associated with the concept of manufacturing strategy. As identified in Sec. 1, these developments can be viewed in terms of changes to the structure and infrastructure of a manufacturing system. Secondly, a model should provide performance measures that are consistent with the manufacturing objectives associated with manufacturing strategy formulation. Thirdly, the time taken to realize a strategy is an important aspect of the strategy concept, as time will be necessary to execute changes to a manufacturing system and for the associated effect to be experienced. Therefore, a model should enable an assessment of the transition of the capabilities of a manufacturing system as a strategy is implemented. Hence, three categories of requirements of modelling are immediately apparent.

Absent above is a measure of the practical viability of modelling. Even if modelling fulfils the three sets of requirements so far established, other issues may still

inhibit application, such as the ability of a model to accurately predict the effect of a change, along with the costs and benefits of using a model. Therefore, a fourth category of requirements exist, termed here the serviceability of modelling.

On the basis of this brief analysis four initial categories of requirements of modelling are established. These categories are:

(1) Assessment of structural and infrastructural changes to a manufacturing system.
(2) Indication of performance in terms of manufacturing objectives.
(3) Assessment of system transition.
(4) Serviceability.

The following section will examine the content of each of these categories in more detail.

2.2. *Assessment of structural and infrastructural changes to a manufacturing system*

There are a number of views in the literature regarding what constitutes the structural and infrastructural changes to a manufacturing system that a strategy has the jurisdiction to affect. Platts[65] however, has been credited for reviewing the intentions of various authors and subsequently providing a platform of terminology and categorization in this situation. This platform has been adopted by this chapter, along with the term "policy areas" and the categories given in Table 2. The categories in Table 2 represent the span of changes to a manufacturing system, across which a strategy formulator is likely to require evaluation to be performed.

In summary, a modelling tool is required that will enable a model to be constructed that allows changes to a manufacturing system to be evaluated, across the structure and infrastructure represented by the policy areas.

2.3. *Indication of performance in terms of manufacturing objectives*

There are a number of issues that need to be explored within this category, namely, gaining an external and internal view of model performance, along with providing measures of product features, design flexibility and quality. This section addresses each of these issues in turn.

2.3.1. *An external view of model performance*

The manufacturing objectives that are generally associated with the manufacturing strategy concept have been explored in Sec. 1. From this investigation, a set of manufacturing objectives have been adopted and these are termed the competitive criteria (Table 1). However, it has also been noted that this set of criteria may be incomplete, as an explicit link to the financial performance of a business is

vague. The current set of criteria focus on the contribution that the manufacturing activity makes to the saleability of a product or product family. Therefore, further consideration is necessary as to the range of manufacturing objectives that should be provided by a modelling tool.

In this case, the terms business and financial strategy are being treated as synonymous, as it is a link with the financial component of business strategy that is in question. Platts and others, only view the financial implications of a manufacturing strategy in a financial justification activity, late in their strategy formulation process. Such justification would usually be constructed using Discounted Cash Flow (DCF) or Internal Rate of Return (IRR).[38]

These financial measures can be redefined into a more basic set of performance indicators. Growth in sales, if measured from a financial perspective, is a view of how the financial turnover of a company is changing over time. Growth in return on assets can similarly be viewed as a change in Return On Investment (ROI) against time. Likewise, growth in return on sales is a statement of profit relative to turnover. Therefore, a group of measures that relate the contribution of manufacturing to the financial performance of a business consists of turnover, ROI, profit and cash flow.

There are, however, many other measures and financial ratios that can complement this set. However, this set is considered here to be the basic variables to assess consistency between financial and manufacturing strategy, and will hence be added to the manufacturing objectives. A model, modelling tool and modelling technique, are expected to support generation of these manufacturing objectives.

2.3.2. *An internal view of model performance*

The manufacturing objectives highlighted above ensure links between manufacturing, financial and marketing functional strategies at an early stage in strategy formulation. These variables can be considered to give an external view of manufacturing system performance. The practitioner, faced with formulating a manufacturing strategy, needs clues to the manufacturing processes or resources within a manufacturing system that are inhibiting overall performance. This may be the case particularly where a large, complex system, is being studied.

The strategy formulator can be appeased to some extent, and without compromising the argument above, if a user is allowed to explore the performance of the principal elements and sub-systems that make up a model. For example, a model should allow a strategy formulator to investigate the effect that a particular manufacturing process, or department within a factory, has on product lead time or other manufacturing objectives. This capability is considered here to be a desirable characteristic of a model, and hence will be adopted as a requirement of a modelling technique. This chapter will use the term "contribution" when referring to the provision of manufacturing objectives at a sub-system level within a model. However, this measure will not illustrate the intensity of activity that is occurring at a resource or sub-system level.

The measure of "utilization" of a manufacturing process or resource is frequently dismissed within the literature as a stimulus of piecemeal development. A measure of utilization can however reveal important information about why a manufacturing system is performing in a certain manner. For example, a delay in product lead time may be caused by a manufacturing resource being heavily utilized. In this sense utilization is not used as an operational management measure, rather, it can be one of a set of measures to assist in the analysis of a manufacturing system. Therefore, to complement the contribution measure given above, utilization of a manufacturing processes or resources is also considered to be necessary.

Finally, to emphasize that the internal measures of utilization and contribution are used to support analysis, and are not intended to be strict manufacturing objectives in the operation of a facility, the term given to this category of requirements will be changed from manufacturing objectives to "performance measures". This terminology change is applied throughout the remainder of this chapter.

2.3.3. *Providing measures of product features, design flexibility and quality*

A question that has been posed is how can a model provide values for each of these three measures? An initial retort from some practitioners was to provide an "index" value for each measure. In this manner a model could generate a value for each product family, such as, "1" being equivalent to poor, "2" being equivalent to fair, and "3" being equivalent to good quality or flexibility. Such an approach is often applied in the strategy literature. However, this approach could in practice cause contention between strategy formulators when attempting to designate such values, and there is a preference for a more factual method.

Investigating first the performance measures of product features and design flexibility further, DTI[27] gives the following definitions:

Product features: Adding capability to the product, or choice for the customer.
Design flexibility: Having the ability to produce products to a customer's specification (customization).

It is apparent that under some conditions there is duplication in these definitions, as by providing a large variety of choice in product features, a product can be matched to customers specifications. The distinction that is believed to be intended above, is that there is a difference between an intended and planned variation in a products specification, and the capability of a manufacturing system to react to an unexpected and unforeseen requirement to modify the design of a product. This shows that strategy deals with unknowable factors. Clearly, if a strategy formulator has little or no conception on the products that a company will need to provide in the future, then a high value of design flexibility is necessary. Such is the case in a "jobbing" environment.

Product modifications concerned with product features may be considered to reflect a proactive intention of a company to offer a variety of product

characteristics, whereas design flexibility can be thought of as a reactive method of offering such variety. Clearly, there are trade-offs associated with choosing a manufacturing strategy that provides high design flexibility rather than product features, and vice versa. For example, Hayes and Wheelwright[39] see that an inflexible assembly line generally promises lower costs, whereas a company that chases demand generally has higher production costs. On this basis, while a direct measure for product features and design flexibility may not be possible, the effect of a company choosing to pursue each of these objectives could be assessed indirectly. Product features and design flexibility could be treated as an input into a model, and as conditions under which a model should operate, with the effects being measured in terms of product cost, lead time, etc.

Dealing with product features and design flexibility in this manner means that a model may be configured to offer a range of planned and intended product variants, commensurate with the intended product features, or tested against an ability to deal with requests made for unplanned product variants as is commensurate with design flexibility. The effect of the model configuration can be measured indirectly in terms of measures such as product cost, lead time, etc.

Quality can be considered in a number of ways, but in particular, capability, reliability and conformance will be discussed here. Quality capability is a statement about the features of a product, whereas quality reliability is concerned with the reliability of a product in service. Quality capability is synonymous with product features, and reliability is a function of quality capability and conformance. Quality conformance is a measure of how well an actual product specification, as determined by manufacture, compares to the specification promised to a customer. Therefore, it is the measure of quality conformance that is of interest here.

To assess quality conformance, some measures are needed for the number of products that do not attain the required specification during manufacture. In practice, a number of quality conformance indicators are available internally and externally to a manufacturing system. Internally, quality conformance can be measured in terms of defective components and scrapped products. This however demands some form of inspection activity to compare manufactured products to their required specification, and all products will need to be assessed in some manner for an absolute measure of quality conformance. If complete quality conformance does not occur and defective products are produced and released into a market, then external effects of quality conformance may manifest in the form of returned products and reduced customer demand. However, this external effect appears to be more difficult to assess than internal measures, as customer response to poor quality conformance may vary considerably and be difficult to measure accurately. Some form of internal quality conformance indicator is therefore favored within a model.

To negate any need for external performance measures, only products that fulfil the desired specification, as determined by the product features and design flexibility discussed above, should be treated as valid production within a model. Quality conformance can be treated as a nominal specification that products must match

or exceed in order to be registered as production. Any substandard products are not recorded as viable manufacturing production, although the associated manufacturing costs must still be accounted for. This approach will allow, for example, the assessment of alternative quality assurance strategies. Furthermore, it supports a strategy formulator who will probably make an assessment of market potential on the basis of products being supplied to the customer at the specification promised.

To summarize, this section has considered at some length how performance measures can be assessed by a model, and has established that product features and design flexibility can be dealt with as a specification input into a model. The effect of producing this specification can then be evaluated in terms of lead time, cost, volume, etc. Quality conformance can be managed by recording as output from a model, only the production of those products that attain an acceptable specification. Finally, the consequence of this reasoning is that none of the three criteria are treated as direct measures of manufacturing system performance. A modelling tool must be capable of configuring a model to manage these criteria, and these can be directly related to the requirements of a modelling technique.

2.4. *Assessment of system transition*

When considering the change of a system over time, a question that arises is whether a model ought to be based on a company's existing manufacturing system and then modified to reflect a strategy, or should the model be concerned with some futuristic manufacturing system and then a strategy sought to connects the future and current state. A benefit of the latter approach is that a "green field" model may help to stimulate creativity amongst strategy formulation. However, there is a risk that such an approach may threaten the credibility of a model because an association with a green field site may be perceived as being idealistic amongst practising managers. Furthermore, it does not take into account that a link between current and future states may not be possible within the resources available to a company. Although the former approach potentially inhibits creativity, it is favored here because of the pragmatic association. Therefore, a model should be based on the existing state of a manufacturing system, and then provide a prediction of manufacturing performance during the transition to a future state.

To place in context what can be expected of the predictive capabilities of a model it is interesting to examine an ancient, but highly appropriate, view of "the future". Pirsig[64] gives an ancient Greek perspective of the future as:

"They saw the future as something that came upon them from behind their backs with the past receding away before their eyes."

As Pirsig points out, knowledge of the future can only be established from a projection of the past. Hence, some authors, such as Thomas and Schwenk,[80]

recognize the implications of future predictions and emphasize that evaluation processes should focus on a comparison of alternatives and checks on the sensitivity of these alternatives to error, mis-estimation and future surprises. In this chapter the emphasis has been moved from attempting to establish what "will" happen to what "could" happen, because this second approach is more realistic when discussing the predictive capability of a model.

Authors such as Danzyger[25] build on the transitional changes to the behavior of a manufacturing system and strongly associate manufacturing strategy with a detailed implementation schedule. This schedule is intended to orchestrate the transition of a manufacturing system from a current state to one that attains some desired manufacturing objectives. Danzyger[25] and Greenhalgh[38] suggest the use of Program Evaluation and Review Technique (PERT), a time-phased network diagram, as an aid in this situation. Such tools provide a time-based continuum against which the implementation of a strategy's content can be viewed, along with the users predicted effects of a strategy on the performance of a manufacturing system. It is important to note however, that such tools contain no analytical evaluation capabilities, and are totally reliant on the users estimates of the impact of a change on the performance of a manufacturing system. In contrast, a valid requirement of a model is felt to be the prediction of the transitional performance of a manufacturing system under such transitional situations.

To summarize, a modelling tool is required that will enable a model to be constructed of the existing state of a manufacturing system, and will then predict the effect on manufacturing system performance as the transitional changes associated with manufacturing strategy content are brought about. These requirements can be directly related to the desired capabilities of a modelling technique.

2.5. *Serviceability*

A strategy formulator will be concerned with the viability of applying a modelling tool in practice. As pointed out by Banerjee and Basu,[10] the selection of a modelling approach depends on the resources available, such as funds and human skill. This point is taken further by Balci[9] who argues about modelling, that:

"A technique whose solution is estimated to be too costly or is judged to be not sufficiently beneficial with respect to the study objectives should be disregarded. Among the qualified ones, the technique with the highest expected benefits/cost ratio should be selected."

These statements emphasize an issue of serviceability that is related to resources, and the consumption of resources may be summarized as a cost for model construction. It can be argued that the primary cost of applying a modelling tool, is the purchase price of the modelling tool, and the cost associated with the time taken to apply the tool by the user. They argue that, as the number of applications increase within a company then, assuming no further purchases are necessary, the user's

application time and associated cost become dominant. Hence, the concern is with purchase price, application time and associated cost.

Purchase price is dependent on the expertise and resources that have been invested to produce a modelling tool, and the characteristics of the market within which the modelling tool is sold. A high market demand may allow the vendor to set a purchase price that is significantly higher than the cost of producing the modelling tool. Likewise, the relationship between purchase price and production cost may vary dependent on market conditions. Therefore, to remove market factors from consideration, this chapter will focus on the resources invested and the cost associated with producing a modelling tool, rather than the price at which vendors choose to sell a modelling tool.

The cost associated with the time taken to apply a modelling tool is a function of the expertise and resources required during this time. If a user requires a high level of expertise to apply a modelling tool, then typically such a user is likely to command a higher value on their working time, than a user who requires no particular skill or training. Hence, the expertise and resources associated with applying a modelling tool, may influence its adoption and application in practice, and should be accounted for in the requirement set.

The expertise required in applying a modelling tool is a function of the complexity inherent to the modelling technique. To some extent this complexity also affects the cost associated with producing a modelling tool. Therefore, a variable of application cost will be used here to represent the expertise and resources required in both construction and application of a modelling tool. This amalgam is based on an assumption that if a modelling approach exhibits a low complexity it will require less expertise and resources to construct and apply a modelling tool, than a more complex modelling technique.

Cost is only one element of the wider issue of model serviceability, users are also concerned with whether models and the information derived from them can be used with confidence. The activities of "model verification" and "model validation" are then presented as mechanisms for developing confidence. Verification is concerned with ensuring that a model performs as intended, whilst validation ensures that the intended model is an accurate representation of the real system being modelled. In this case accuracy as an insurance of confidence, and the relevance of model accuracy is stressed by many authors. However, the use of accuracy can be qualified by also considering credibility. Credibility is concerned with how believable the results of a model are. Accuracy and credibility are both independent statements about model validity. Intriguingly, a model may be accurate and not credible, or more dangerously, not accurate but credible.

On the basis of the arguments presented above it is apparent that a strategy formulator will require a modelling tool to provide an accurate and credible model quickly and inexpensively. A modelling technique should enable the construction of such a modelling tool.

3. The Modelling Techniques

The objective of this chapter is to establish a comprehensive foundation of knowledge about modelling techniques. This objective is realized, and the chapter structured, to first develop a taxonomy of models. This taxonomy is then used to expose the variety of models available, and a number of representative modelling techniques are chosen to illustrate each sub-class of the taxonomy.

3.1. *Foundation to model taxonomy*

A taxonomy is a framework of classification. Provision of an unambiguous framework requires a clear definition of scope and consistent terminology. This section provides a suitable foundation for a taxonomy of models by addressing both these issues.

This chapter is concerned with models that are abstract representations and emulations of real world objects or systems. Such models exhibit at least one distinctive quality that pertains to the real object or system, for example, visual impact, geometric dimensions, or behavior. These are distinctly different from, and this taxonomy does not include "methodologies" that are approaches for structured enquiry, judgement and problem solving about a real world object or system.

Unfortunately, this categorization is not definitive as some techniques can be applied as a model and a methodology. For example, the operation of a machine tool could be illustrated by an Ishikawa cause and effect diagram even though this technique is more usually applied to establishing relationships between specific factors and their respective causes. A more usual application of this technique would be, for example, in an investigation into causes of poor product quality.

Furthermore, some modelling techniques, such as Soft Systems Methodology (SSM)[18,19] incorporate both a methodology and modelling approach, in this case a Rich Picture (RP) as a model around which an enquiry is conducted. Indeed, within the field of decision support literature generally, authors often support their description of modelling with an application methodology. Examples of this are Carrie[14] and Law and Kelton.[48] Therefore, it is important to recognize that a distinction between methodologies and models is not definitive and needs to be cautiously applied.

Used in construction of the model taxonomy are the terms "model instance", "model type", "modelling technique" and "modelling tool". Each of these terms requires a fuller explanation.

Banerjee and Basu[10] provide a suitable definition of model instance and model type. They see a model instance as a specific formal representation used in addressing a particular problem, whereas a model type is a possibly infinite collection of model instances characterized by a set of rules and/or properties that distinguish instances of that model type from those of other model types. Hence, when the term "model" is used in isolation in the literature, and also within this chapter, this is usually an implicit reference to a model instance.

The properties and rules associated with defining a model type need not be constrained to a particular grouping of modelling instances, and can be applied to group together models at various levels within a hierarchical taxonomy. For example, physical models may be considered as one model type, within which a subset of analogue model type may be found.

When discussing a model type that is directly involved in model construction this chapter applies the term modelling technique. This terminology enforces a distinction between varying definitions of model type in a hierarchical taxonomy, and the principal mechanism that provides a basis for actual model construction. In this sense a modelling tool is the means through which a modelling technique can be applied, and the modelling technique can be associated with a set of distinguishing properties and rules.

Some modelling techniques may be applied in practice using computer based modelling tools, and a number of tools are seen to exist for various modelling techniques. The actual modelling tools are not as important to this chapter, as it is the underlying technique that is seen to characterize the capabilities of a modelling tool.

3.2. *Forming a taxonomy of models*

In this chapter a taxonomy is required to thoroughly establish the range of modelling techniques that support manufacturing strategy evaluation. Unfortunately, the literature does not provide a consensus on a form of model type taxonomy. For example, Ackoff and Sasieni[1] refer to three categories of models, namely, iconic, analogue, and symbolic models, whereas Mihram[53] refers to replication, quasi-replica, analogue, descriptive, simular and formalization models. Schmidt[74] has attempted to address this situation by identifying that models can be classified according to the dimensions of:

(1) The manner in which the model describes the system.
(2) The purpose of the model.
(3) The description of the time dependent behavior of the system.
(4) Description of the random behavior of elements of the system.
(5) The description of system change as a discrete or continuous phenomena.

Each of these dimensions offers a potential taxonomy framework. Unfortunately, no one taxonomy is free of limitations. Taxonomies based on both modelling medium or model purpose are popular, but lack a consensus on form and terminology, whilst distinctions based on static, dynamic, stochastic, deterministic, discrete, or continuous characteristics can be imprecise. This section uses knowledge of existing model type taxonomies to define a framework that will provide a contemporary categorization of modelling techniques.

A taxonomy based on modelling medium is most popular in the literature and provides a particularly comprehensive framework. Mihram[53] in particular gives a

detailed and comprehensive taxonomy, but incorporates some distinctions based on static, dynamic, stochastic, and deterministic characteristics that are considered to be imprecise. Moreover, the work of Mihram is not universally adopted in the literature by later researchers, and is relatively early and in danger of being out of step with current semantics. Therefore, to develop an acceptable taxonomy at this point, the material based classification of Mihram is taken as a coarse foundation, and then refined using common views and terminology in the literature.

Consider first, the category of models that Mihram termed "physical". This terminology is strongly supported in the literature, for example, Shannon,[75] Carrie,[14] Law and Kelton.[48] However, there are some varying opinions in the literature on the appropriate sub-classes of physical models. Shannon[75] and Schmidt[74] consider iconic and analogue models, whereas Carrie,[14] and Law and Kelton[48] see iconic and physical models to be synonymous. Mihram is effectively subdividing models that are often termed iconic, into two distinct classes, of replication and quasi-replica, and in doing so, enhancing the precision of the terminology. However, an amendment can be made to the definition of quasi-replica models given by Mihram, in that a dimension need only be modified in a model rather than be missing altogether. Therefore, these two sub-classes have been adopted with definitions based on Mihram[53]:

- Replication model: A spatial transform of an original physical object in which the dimentionality of the modelling is retained in the replica.
- Quasi-replica model: A physical model in which one or more of the dimensions of the physical object are missing or modified.

Some authors recognize the existence of analogue models. Whilst this is not universal, no case is apparent to formally dismiss such models. Hence, this sub-class of model types has also been adopted here, with a suitable definition again taken from Mihram:

- Analogue model: A model which bears no direct resemblance to the modelled phenomena.

Consider now, the category of models that Mihram termed "symbolic". Whilst the term symbolic is popular in the literature, a few authors, such as Law and Kelton,[48] do see mathematical models as the appropriate name for this category. Furthermore, opinions also vary considerably on the appropriate sub-class of symbolic model types. As a foundation, Mihram considers these sub-classes to be formalizations, simular and descriptive.

Consider first formalizations, which Mihram felt to be a class of symbolic models in which the symbols are manipulated by a well formed discipline. There is strong support that one subset of symbolic model types is some form of mathematical representation of a system. Likewise, authors such as ElMaghraby and Ravi[29] recognize models forms that use mathematical variables and explicit expressions to represent

the physical qualities and behavior of an actual system. Carrie[14] stresses that often these equations can be solved to give the optimum solution for the problem encountered, and that these models are optimizing in the sense that they yield the one best value for the function concerned. There are however instances where, due to the complexity of the system involved, no optimum solutions can be found directly, and suitable conditions need to be established through a number of iterations. Law and Kelton[48] point out that in this case a mathematical model can be studied by numerically exercising the input question to see how the output is effected.

The association of explicit expressions with mathematical models, is also consistent with the definition of formalizations given by Mihram. However, the term mathematical model is very popular in the literature whereas formalization is not. Therefore, this sub-class will adopt the term mathematical model along with an associated definition provided by Carrie[14]:

- Mathematical model: Explicit analytical formulae describing known relationships.

Consider simular models that Mihram defined as a sub-class of symbolic models whose component symbols are not entirely manipulated by the operations of well formed mathematical disciplines. Mihram introduced the term simular models as he recognized that the word simulation was being loosely used. Even so, it can still be argued that physical models are all examples of simulations in as much as they imitate reality. Therefore, it may be thought to be imprecise of authors such as Mihram to incorporate an explicit model type distinction on simulation, at one specific level in a hierarchical taxonomy, when simulation can be used implicitly to relate to other categories in the taxonomy.

Whilst the term simulation is weak because it can be generally applied, it is also vague for the sub-class that it is intended to represent. As Schmidt[74] points out, the distinction between mathematical and simulation models is not one that can be easily drawn. Some of this difficulty appears to have occurred because both model forms rely on mathematics, inherent to which, are numerical and logical expressions. A demonstration of this contention is provided by Pidd,[63] who describes System Dynamics (SD) as a form of simulation that is expressed in a mathematical form. However, some distinction between these model forms becomes apparent when the process of model construction is investigated.

Carrie[14] considers simulation modelling as describing the behavior of a system as a whole, by defining in detail how various components interact with each other. For example, simulation modelling with SD consists of tracing, step-by-step, the actual flow of orders, goods, and information, and observing the series of new decisions required. Hence, a more precise term for the simulation model sub-class would be "implicit mathematical models". Unfortunately, adoption of such new phraseology risks misinterpretation of the research contribution in this chapter. Therefore, to reflect popular usage in the literature, this chapter will cautiously adopt the

Table 3. Taxonomy of model types.

Main Class	Sub-Class	Definition
Physical	Replication	A spatial transform of an original physical object in which the dimension of the modelling is retained in the replica.
	Quasi-replica	A physical model in which one or more of the dimensions of the physical object are missing or modified.
	Analogue	A model which bears no direct resemblance to the modelled phenomena.
Symbolic	Schematic	A graphical representation of a system using symbols.
	Simulation	A model of the behavior of a system as a whole by defining in detail how various components interact with each other.
	Mathematical	Explicit analytical formulae describing known relationships.

term simulation for the category of models that Mihram[53] termed simular models. A suitable definition of simulation is derived from Carrie[14] as:

- Simulation model: A model of the behavior of a system as a whole by defining in detail how various components interact with each other.

Finally, consider the category of models that Mihram termed descriptive. The term descriptive models can be confused with a model's purpose. Riggs[70] and Heizer and Render[41] use the term schematic model for a drawing or chart of reality. Therefore, to overcome the contention associated with the word descriptive this chapter will use the term schematic for a symbolic model that does not contain any manipulation of variables. Rather, it is a structured statement of a system's content, structure and interactions. A schematic model will be defined in this chapter as:

- Schematic model: A graphical representation of a system using symbols.

In conclusion, the work of Mihram[53] has provided a foundation against which the views of more recent authors, and evaluations in terminology semantics, can be contrast. The resulting taxonomy and model type definitions are presented in Table 3.

3.3. *Categorization of modelling techniques*

The previous section has established a taxonomy of models, thus providing a basic framework through which a comprehensive range of modelling techniques for manufacturing strategy evaluation can be investigated. The range of models, and associated modelling techniques, can be illustrated by identifying one or more representative modelling techniques for each sub-class in the model taxonomy. These representative techniques capture the flavor of each model type whilst respecting that the task in hand is to support the evaluation of a manufacturing strategy.

A pedantic appraisal of modelling techniques would find that some approaches do not neatly fall into the taxonomy developed here, rather they span a number

Table 4. Possible modelling techniques for manufacturing strategy evaluation.

Main Class	Sub-Class	Generic Modelling Technique	Abbreviation
Physical	Replication	Model construction using an identical mechanism to that used in the real system under study.	Replica
		Model construction using any mechanism that provides a spatially identical model to the real system under study.	Non-functional replica
	Quasi-replica	Model construction using any mechanism that provides a fully functional scaled model.	Scale
		Model construction using any mechanism that provides a scaled model that lacks functionality.	Non-functional scale
		Model construction using any mechanism that provides a two dimensional scaled model that lacks functionality.	2D non-functional scale
	Analogue	Modelling using an analog computer.	Analog
Symbolic	Schematic	Rich Picture	RP
		Integrated Enterprise Modelling	IEM
		$IDEF_0$	$IDEF_0$
	Simulation	Discrete Event Simulation	DES
		System Dynamics	SD
	Mathematical	Queuing Theory	QT
		Activity Based Costing	ABC
		Business Planning	BP

of categories. For example, "Petri-nets" is a simulation technique that provides a graphical representation as a schematic model. These concerns have been addressed by identifying the principal modelling techniques involved in each case and then cataloguing these independently. A summary of the modelling techniques chosen in each case is given in Table 4.

3.3.1. *Physical replication models*

Models in this category have been defined as spatial transforms of a real world object or system. As these models are materially not far removed from the actual real system under study, they can be said to exhibit a low level of abstraction from reality. This close association means that there is limited opportunity for various forms of model to exist. A range of models is provided because a model may, or may not, contain the complete functionality that exists in the real system.

One such model occurs when a spatial replica of the physical features, and visual aesthetics, of a system is constructed that lacks functionality. Such is the case with body work styling in the automotive industry. Alternatively, aesthetics and functionality may be combined to provide a model that is a complete replica. Indeed, this second case can be considered to exist when, for example, a machine tool is installed within a factory but is awaiting commissioning prior to being entered into production. The commissioning activity can be thought of as a form of modelling,

used to perform adjustments that ensure full operational functionality, and which ceases when the machine is entered into production.

In each of these examples given above, the model form varies to suit the purpose of the model; likewise the modelling technique varies to suit the model form. In the case of automotive body work styling, the modelling technique may be sculpture. Whereas, with a pre-commissioned machine tool the modelling technique can be considered to be the machining, casting, and fabrication processes, that have been used to manufacture the machine tool. In the former case the modelling technique may be generalized as model construction using any mechanism that provides a spatially identical model to the real system under study. Likewise, the latter case can be generalized as model construction using an identical mechanism to that used in the real system under study.

3.3.2. *Physical quasi-replica models*

These models have been defined as physical models with one dimension modified or missing. Similar to physical replication models, a distinction exists based on whether functionality is combined with a quasi replica model.

One form of quasi-replica model is where two dimensions are scaled up or down in size, the third dimension is missing, and there is a lack of functionality. An example of such a model is a photograph. An alternative model form occurs where a model is still scaled up or down in size and functionality is absent, but the third dimension is now present. An example of this second case is a static scale model. Such models are often used for factory layout planning.[14] Finally, functionality may be retained in a physically scaled model of the real system. An actual application of such a model is provided by O'Reilly *et al.*[61] who describes a 1/35 scale model of an automotive painting process that consists of 62 position sensors, 39 stops activated by solenoids, 31 pneumatically operated lift tables and 46 motors.

3.3.3. *Physical analogue models*

Physical analogue models have been defined as bearing no direct visual resemblance to the system being modelled. There is an absence in the previous sub-classes of physical models that exhibit functional, but not physical, similarity to the real system being studied. This niche is filled by analogue models.

Ackoff and Sasieni[1] cite an example of an analogue model being a hydraulic system representing electrical, traffic, and economic systems. An example of an analogue model of a manufacturing system can be provided by an electrical circuit model. In this case electrical elements such as transistors, resistors and capacitors are connected in such a way as to represent a manufacturing process. To model discrete product manufacture, a digital electrical signal generator could be used as an input to such a model. This form of modelling technique appears to have been popular when analog computers are applied.[53] Of the modelling techniques in this

category, an approach to modelling using analog electrical circuiting appears to be suited to manufacturing system.

3.3.4. *Symbolic schematic models*

Such models have been defined as a symbolic graphical representation. A common example of a schematic model is an engineering drawing. Such a drawing makes use of an abstract graphical representation of a component. In this example a change in symbolic representation, say from third to first angle projection, would effectively invoke the use of a second modelling technique. Therefore, as this example illustrates, there is considerable opportunity for modelling technique diversity.

This diversity has been harnessed through a process of considering the range of formality, in other words complexity of graphical syntax, associated with modelling techniques. This has been coupled with a search for techniques that are advocated within the literature as particularly capable, or providing a bias towards, manufacturing system modelling.

The most involved modelling syntax appears to exist within systems analysis techniques. Within this category are such techniques as, Data Flow Diagrams (DFD) (Downs *et al.*[26]; Johansson *et al.*[45]); Input/Output Analysis (Olsmats *et al.*[60]; LUCAS[50]); CORE (Kehoe *et al.*[47]; LUCAS[50]); $IDEF_0$ (Brovoco and Yadav[11]; LUCAS[50]; Williams and Pontin[85]; Johansson *et al.*[45]); Structured Analysis and Design Technique (SADT) (Marca and McGowan[51]; Ziya Aktas[87]); and Structured Systems Analysis and Design Methodology (SSADM) (Kehoe *et al.*[47]).

A common approach is $IDEF_0$ (Baines and Hughes[6]; Brovoco and Yadav[11]; Brovoco *et al.*[12]; Baines and Colquhoun[7,8]). The $IDEF_0$ technique produces a model which is essentially a flow diagram that illustrates the activities within a manufacturing system. This technique is characterized by both a specific syntax and decomposition of a system content to various levels of detail.

The origins of $IDEF_0$ lie in the development of SADT by, amongst others, D. T. Ross (Marca and McGowan[51]; LUCAS[50]). Marca and McGowan describe $IDEF_0$ as a standardized subset of SADT which has been promoted by the United States Department of Defence under their ICAM (Integrated Computer Aided Manufacturing) programme. The IDEF acronym is formed by taking the "I" from ICAM and "DEF" from definition (Brovoco and Yadav[11]). Where SADT has been developed to address all phases of a systems development (Ziya Aktas[87]), $IDEF_0$ is intended purely for representing the functional relationships in a manufacturing system (Baines and Colquhoun[8]). However, $IDEF_0$ is the name given to one standard and there are at least three IDEF variants discussed in the literature.[a] $IDEF_0$ views a system as the set of functions it performs. $IDEF_1$ views a system by studying information it contains. $IDEF_2$ views the time dependent behavior of a system

[a]Though not supported in the literature, discussions with other researchers and software companies have established the existence of fourteen IDEF standards each addressing a particular application.

(Brovoco and Yadav[11]). Due to the support given to $IDEF_0$ in the literature, it has been chosen as a generic modelling technique for this study.

As presented above, $IDEF_0$ is believed to be typical of a classical approach to structured system analysis and design. Such techniques are characterized by strict rules and considerable abstraction from the system being modelled. As an alternative to this classical approach, a modeller may use techniques such as "material flow charts" (Currie[23]; Johansson *et al.*[45]). These are common operational research data capture techniques that provide a graphically similar symbolic representation to a system being modelled by $IDEF_0$.

Business Process Re-engineering (BPR) (Johansson *et al.*[45]) appears to have been a recent catalyst for a number of modelling innovations. A variety of techniques have been constructed to bridge the features of such techniques as $IDEF_0$ and material flow charts to give a comprehensive model of a business. The term Integrated Enterprise Modelling (IEM) (Mertins *et al.*[52]) can be given to these approaches. IEM is an emerging topic in the literature and there are inconsistencies in the use of terminology. However, because IEM is closely associated with BPR, and features a more relaxed syntax than $IDEF_0$, it is also chosen as a generic modelling technique.

Finally, Rich Pictures (RP) are an integral part of Soft Systems Methodology (SSM) and have no strict syntax. These have been established, through the contribution of authors such as Checkland,[18] to illustrate the content and behavior of a system as a means to promote communication between individuals. Such pictures are a cartoon-like illustration of the system under study. These pictures are intended to break down the communication barriers that are associated with written statements and technical diagrams.

3.3.5. *Symbolic simulation models*

Simulation has been defined as modelling the behavior of a system as a whole by defining in detail how various components interact with each other. A review of the literature has established that there are three principal forms of simulation modelling techniques, namely continuous, discrete event and combined. There are however some recent additions, or evolution's on a theme, that need to be considered before generic modelling techniques can be chosen.

Consider foremost Discrete Event Simulation (DES). This modelling technique is defined by Roth[71] as measurements made on variables which are affected by instantaneous changes of system state. However, a number of approaches to DES modelling are apparent.

The time varying behavior of a system can be modelled using $IDEF_2$. This technique is intended to complement $IDEF_0$ and $IDEF_1$ to provide a comprehensive modelling approach. As $IDEF_2$ is based on an instantaneous system change (Brovoco and Yadav[11]), it is a form of DES.

Schmidt *et al.*[73] discuss Object Oriented Simulation (OOS), Artificial Neural Network Simulation (ANNS), as well as DES. However, the distinction between

OOS and DES in this work is actually based on the principles of the computer language used. DES is considered to be applied using procedural computer programming languages, whilst OOS is applied using object-oriented languages which are generally seen by authors such as Graham[37] as providing improved programming efficiency through the use of polymorphism, abstraction, and inheritance. The principles of DES are however independent of the computer language used, indeed DES can be performed without a computer using "activity cycle diagrams" as illustrated by Carrie[14] and Williams and Pontin.[85]

Schmidt *et al.*[73] describe an ANNS model as basically a black box with a set of inputs that yield a set of outputs. They state that ANNS models are programmed by training, but the route to acquiring the output is difficult to relate to the system being modelled. Furthermore, they see ANNS models as the weakest in the same areas as the human brain; exact mathematical answers, computational logic and speed. Likewise, Fishwick[32] concludes that ANNS models provide a less powerful tool than traditional simulation approaches. Other artificial intelligence applications to simulation, focus on reducing expertise required for model building and interpreting simulation results with expert judgement (Garzia *et al.*[36]).

DES can be provided through the application of "Petri-nets" and "colored Petri-nets". Cecil *et al.*[15] cite several examples of the use of Petri-nets in the modelling, analysis and control of systems that can be considered to perform discrete component manufacture. They see one of the main advantages of this approach as DES being a graphical system representation. Unfortunately, Petri-nets are at a relatively embryonic stage and extensive research and development is needed to increase the scope of this technique to modelling, analysis, and control of manufacturing systems (Cecil *et al.*[15]).

Continuous simulation is a modelling technique that is concerned with constructing a model in which the state variables change continuously with respect to time (Law and Kelton[48]). Such models involve differential equations that give relationships for the rate of change of the system variables with time. Law and Kelton point out that if the differential equations are particularly simple, they can be solved analytically to give the values of the state variables, for all values of time, as a function of the values of the state variables at time zero. However, such a direct solution is rarely feasible where real systems are involved and a numerical approach needs to be adopted (Pidd[63]). Pidd points out that such an approach is incorporated into System Dynamics (SD) where the differential equations that describe the behavior of a real system are represented by difference equations.

The origin of SD, as given by Wolstenholme,[86] commenced with Forrester[34,35] who created a subject area originally known as "industrial dynamics". As pointed out by Towill,[83] the subject received this name because it was focused at work in terms of strategies for improving industrial systems performance. He goes on to say that for a brief while, the methodology became known in the UK as "managerial dynamics", and more recently the field has been named SD due to the expanding range of applications considered by Forrester, often involving the social sciences.

Wolstenholme[86] points out that SD can be considered in terms of "qualitative" or "quantitative" approaches. He states that qualitative SD is concerned with creating cause and effect diagrams that are known as casual loop, or influence diagrams. Whereas quantitative SD is defined by Wolstenholme as quantitative computer simulation modelling.

Continuous simulation modelling in a SD guise has been applied to a variety of business and manufacturing applications; Forrester[33] examines modelling of production and distribution systems; Dangerfield and Roberts[24] apply SD to investigate scenarios of the consequences of capacity requirements in the UK steel industry; Edghill *et al.*[28] and Olsmats *et al.*[60] have applied this technique to the modelling of production, inventory and distribution systems. The literature also describes a number of SD applications in financial modelling, see, for example, Ref. 82. In this second case, explicit financial equations are imbedded in a SD modelling tool, and exercised with a range of input variables. Combined simulation is a combination of discrete and continuous approaches. The capability of combined simulation can be established through a consideration of discrete and continuous approaches.

3.3.6. *Symbolic mathematical models*

Mathematical modelling has been defined as the use of explicit analytical formulae to describe known relationships. From a general perspective, a wide variety of mathematical models have been developed to describe the effect of particular relationships. In this sense equations that relate mass, acceleration and force, or price, cost and profit, are all examples of mathematical models. Modelling techniques in this instance are developed for a specific set of applications. This association is often so close, that the modelling technique is inevitably a structured expression that only requires populating with numerical values about the system being modelled. Therefore, the process chosen to review mathematical modelling techniques is to identify sets of applications of interest in manufacturing strategy evaluation, and to subsequently relate these back to identify modelling techniques.

Cited earlier, Ansoff and Brandenburg[4] summarize the task of analytical models as allowing alternative organization designs to be modelled, the outcome of each design predicted against objectives, allowing the most suitable design to be selected. In manufacturing strategy formulation, such objectives are market and financially oriented. Therefore, modelling techniques of interest are those which provide market and financial information of the effect of developments to a manufacturing system.

Market criteria consist of product based information such as lead time, volume, etc. Modelling techniques that provide such information include "control theory concepts", "metamodels", and Queuing Theory (QT).

Axsater[5] gives three forms of standard control theory methods, namely "linear deterministic", "linear stochastic", and "non-linear deterministic". However, he observes in the literature a slight decline in interest in this subject. Considerable achievements though, have been made in the recent past by Edghill *et al.*,[28]

Popplewell and Bonney,[66] and Cheema *et al.*[17] Edghill *et al.*[28] note that analytical control theory is labour intensive, and requires a degree of specialization not to be expected from potential industrial users with no previous experience.

A metamodel is often constructed to approximate the behavior of elements within a simulation model. In situations where frequent model execution is necessary, a metamodel is simpler and less costly than conducting many simulation experiments. Hence, there are a number of applications within the literature of metamodelling being applied to production control situations. However, to construct a metamodel it is invariably necessary to carry out some simulations of input-parameter combinations to obtain data from which the parameters of the metamodel are established (Law and Kelton[48]). Therefore, if a system can be adequately modelled with a simulation technique, then the arguments for applying metamodelling are significantly compromised.

QT predicts the average behavior of a manufacturing system over a medium to long time horizon, and the overall insight that QT provides is appropriate for the design and planning stage of a manufacturing system.

Financial criteria are addressed by Cooke and Slack[20] using two forms of financial models. The first is mainly concerned with conventional accounting measures and relationships, whereas the second focuses on long time scales.

A number of financial modelling techniques exist for accounting measures and relationships. These include "marginal costing", "absorption costing", Activity Based Costing (ABC), and "throughput accounting". Likewise, some models have been developed for a special purposes such as IVAN (InVestment Analysis).

ABC has been developed to overcome limitations associated with traditional accounting procedures, and the main research work on ABC has been carried out at the Harvard Business School by Cooper, Kaplan and Johnson. Kaplan[46] observed that reliance on traditional costing systems in the current competitive environment would provide an inadequate picture of manufacturing efficiency and competitiveness. However, Steeple and Winters,[78] credit Newton[59] with research that has established that few companies have successfully implemented ABC. Likewise, Cooper[21] states that implementation of an appropriate ABC can require prohibitive effort and time scales.

ABC need not however directly replace traditional accounting procedures, rather it can be used as a diagnostic tool to lay bare all overhead costs and to establish inaccuracies within an established accounting system. ABC should not be used to produce monthly profit statements, traditional systems can be used for that, rather ABC should be used for strategic decisions and profitability analysis. Furthermore, ABC has a close association with absorption costing suggesting that a company can be flexible to the extent that ABC is adopted.

Mathematical modelling techniques that consider the long term performance of a business are termed Business Planning (BP) models or "financial planning systems". The structure of such models is based on financial conventions, some of which are

legally enforced, such as tax laws, therefore few variations of models exist. Variety is only possible where a choice in convention exists, such is the case in depreciating the value of a capital investment.

4. Understanding the Limitations of Popular Modelling Techniques

Table 4 has presented a comprehensive range of modelling techniques that can be used for manufacturing system modelling. The task in hand is to establish how well such techniques can, in practice, support the evaluation of a manufacturing strategy. To achieve this, this section provides a consideration, of each of the representative modelling techniques, against each of the requirements of manufacturing strategy evaluation. This enables the limitations of these techniques to be critically identified. The culmination of this section is the identification of a set of modelling techniques that are useful to the strategy evaluation task.

4.1. *Assessment of structural and infrastructural changes to a manufacturing system*

The issue here is whether concerns exist with any generic modelling technique that supports modelling across policy areas associated with a manufacturing strategy. Physical models offer some concerns, this being particularly the case with scale and analog models, as the following discussion reveals.

Using a scale model, it is possible to build a dimensionally smaller or larger model of a modelling facility, and for the model to provide some form of functionality. For example, O'Reilly *et al.*[61] has formerly been cited for describing a 1/35 scale model of an automotive painting process that consists of 62 position sensors, 39 stops activated by solenoids, 31 pneumatically operated lift tables and 46 motors. Saunders *et al.*[72] present a scale model of a production system, incorporating materials handling, based on an electric model train, to test alternative production and repair schedules. Law and Kelton[48] cite an example of a table-top model of a materials handling system.

Scale models appear to be principally restricted where human resource issues are concerned. Unlike machinery, a human cannot be physically scaled down, a human icon could be included in a model but will lack functionality. This can be overcome to some extent by a person interacting with a model as if working with the real system. If, however, such an approach is adopted, then strictly an amalgam is being formed with a replica model.

It is important to mention that both non-functional scale models, and 2D non-functional scale models are not affected by the human resource issue. This is the case because there is no functionality offered by each of these approaches, and representation of a human resource by an inert icon is acceptable.

Recent literature on analog computer models, or analogue[b] models in general, is extremely scarce. Authors such as Mihram[53] explain that analog computer models are constructed by connecting electrical elements such as transistors, resistors and capacitors, in such a way so as to represent continuous process systems. Pritsker[68] points out that during the 1950s and 1960s, analog computers were the primary means for performing continuous simulations. He also states that analog computers lack the logical control functions and data storage capabilities of the digital computer.

As well as limitations of the analog computer hardware, there is also a concern that the user will need to be conversant in control theory to program such a machine. Edgehill *et al.*[28] have been previously cited for arguing that analytical control theory is labor intensive, and requires a degree of specialization not to be expected from potential industrial users with no previous experience.

Considering symbolic models, there are a number of assertions in the literature as to the flexibility of mathematical models in particular. A previously referenced example is Pidd[63] who states:

"...queuing theory models ... cannot cope with many types of problem."

However, such an assertion is countered by Suri and Diehl[79] who see such approaches as the right tool for the planning and preliminary evaluation of a manufacturing system design. Likewise, there are similar debates in the literature contrasting System Dynamics (SD) and Discrete Event Simulation (DES) modelling techniques. An example of this case is Love and Barton[49] who argue that the SD approach introduces approximations that undermine the accuracy or even the utility of the results generated. In a similar manner, Towill[84] counters this with an argument that:

"We feel that many critics of System Dynamics as a methodology have failed to distinguish between the general concepts and one particular approach to modelling and system performance."

The contentions associated with the existence of a debate, and the particular nature of the flexibility required in manufacturing strategy evaluation, undermines confidence in discounting such techniques on the basis of this form of evidence.

In summary, although a number of concerns exist, only scale and analog computer models can be confidently dismissed when considering the span of changes to a manufacturing system associated with a manufacturing strategy.

[b]The term "analogue" refers to a category of physical models in the model taxonomy, whilst an "analog" computer model is the modelling technique that has been chosen to represent the analogue category of models.

4.2. *Performance measurement*

Internal and external measures of manufacturing system performance are necessary. The external measures allow consistency to be maintained with marketing and financial strategies, whereas the internal measures support strategy formulation through providing analysis of the activities within a manufacturing system.

Unfortunately, a comparison of the required performance measures, against the capabilities of a generic modelling technique, is difficult to conduct through the literature. This is because in most articles the performance measurement capabilities of modelling techniques are presented implicitly, and as a consequence, there is little confidence that absent performance measures are true limitations rather than simply omissions.

Accepting this concern, a coarse review of performance measures is still possible. Each of the required performance measures have in common the fact that some form of numerical capability is necessary. This is to reflect a need of modelling to provide both insight and prediction about the effect of a manufacturing strategy. On this basis, the literature is adequate to support a review on whether or not a modelling technique supports performance measurement, and hence, an investigation can proceed by establishing whether or not a modelling approach can provide numerical information. If a generic modelling technique satisfies this requirement, then, assuming all other requirements are satisfied, a more critical enquiry into the performance measures supported by a modelling technique will be conducted during the experimentation in the following chapter.

On this basis, a number of both physical and symbolic models have distinct limitations. The physical modelling techniques of non-functional replica, non-functional scale and 2D non-functional scale, by definition, do not contain the necessary functionality to provide numerical capabilities. The symbolic modelling techniques, of Rich Pictures (RP), Integrated Enterprise Modelling (IEM) and $IDEF_0$, are also limited in this instance for a similar reason. The focus of RP, for example, is on gaining consensus amongst personnel involved in the problem solving process. This view is directly supported by Checkland[18] who states that no matter how the models are used for comparison with the real world, the aim is not to "improve the models" but to find accommodation between different interests in a situation.

Although IEM and $IDEF_0$ are characterized by a lack of numerical capability, recent innovations in modelling tools are making provisions to overcome these limitations. For example, DESIGN/IDEF allows numerical attributes to be assigned to an activity. To be strict, such functionality is actually provided by enveloping a mathematical modelling facility within the $IDEF_0$ modelling tool. However, such a combination is distinct in the manner in which an $IDEF_0$ model will allow, through the decomposition facility, a mathematical model to be constructed. To acknowledge the numerical capabilities, both $IDEF_0$ and IEM will be retained for further study.

Finally, a potential exists to combine a non-functional replica, a 2D non-functional scale or a non-functional scale model, with symbolic models to provide the required numerical capabilities. These approaches could be combined with either a simulation or mathematical modelling technique. The potential benefit in each case is an improvement in model credibility through the realism associated with physical models. However, combining physical and symbolic models is likely to be difficult. Furthermore, the nature of symbolic models is such that they can be computer based and provide 2D graphical animation, and this negates some of the perceived credibility benefits of physical models. Finally, there is a high probability that model construction costs will be higher with a combined modelling approach.

4.3. *Assessment of system transition*

It has formerly been established that modelling should allow both the behavior of a manufacturing system, along with changes to the content of the system, to be evaluated as time advances. The issue here is whether there are any generic modelling techniques unable to support the construction of such models.

There is a concern that RP, IEM and $IDEF_0$ forms of symbolic models are static illustrations of the content, interactions and structure, of a manufacturing system. Such static illustrations can be thought of as "snap-shots" of a manufacturing system at an instant in time. A combination of either IEM or $IDEF_0$, with some form of mathematical model, may provide a numerical capability. Furthermore, adopting such numerical capabilities may also coarsely overcome some concerns of only providing a snap-shot of a manufacturing systems performance. This may be accomplished, to some extent, by using time averaged values or linking a number of models to represent phases in the evaluation of a manufacturing system.

4.4. *Serviceability*

A review of the literature has revealed that both application cost and time are prominent concerns with some forms of physical models. For example, Hogg *et al.*[44] consider the construction of a flight simulator, the dynamic behavior of which is well matched to an aircraft. They state that such a model can be classed as a replica of the aircraft system under consideration, and in this case the cost advantage of the simulator compared with experimentation with the real system is given as being in the order of 10 to 1. However, the cost associated with experimentation with an aircraft is very high and thus the cost of the model is also high. This concern is reinforced by ElMaghraby and Ravi[29] who argue that the major disadvantage of physical simulators is their cost. They point out that the construction of these simulators is tedious and time consuming, along with them being relatively inflexible after construction. Therefore, it can be deduced that a physical replica of a manufacturing system will also exhibit these limitations of high cost and inflexibility once constructed.

There are, however, various generic modelling techniques within the physical category. Whilst the concerns highlighted above are undoubtedly true of a replica model, this is not the case for 2D non-functional scale models. An example of the latter is a photograph. Likewise, there is little confidence that non-functional replica, non-functional scale, and analog computer models, can be discounted on the basis of application cost and build time.

A scale model, however, does include functionality and may be expensive to provide. Consider the form that such a model of a typical factory would take, containing products, materials handling, etc. Furthermore, human resources could be included by forming an amalgam with a replica model. On this basis, such a model is more closely associated with a replica than the non-functional forms of model given above.

4.5. *Useful modelling tools in strategy evaluation*

Initially physical models have been considered for the task of manufacturing strategy evaluation, and for these models the following conclusions were drawn. Physical models have distinct limitations. A physical replica requires excessive resources to apply, and can also be expensive to modify once constructed. Non-functional replica, non-functional scale, and 2D non-functional scale models are limited because, by definition, no numerical capabilities are available. Scale models are restricted because, although they contain functionality, it is not possible to directly model human resources. Also, embedding functionality into such a model is likely to be expensive. Finally, analog computer models have a limited flexibility.

Symbolic modelling techniques are more suited to the task of manufacturing strategy evaluation, and though limitations do exist, the following capabilities have been established empirically for the representative modelling techniques of Discrete Event Simulation, System Dynamics, Queuing Theory, Activity Based Costing, Business Planning, $IDEF_0$ and Integrated Enterprise Modelling.

Discrete Event Simulation can evaluate a wide variety of issues, to a low level of detail, with relatively high model accuracy, and good model credibility. However, the technique has a slower model build rate than System Dynamics, and the resulting model will take longer to execute.

System Dynamics has the flexibility to address a wide variety of issues, it exhibits a relatively rapid model build rate and model execution time. However, because of the inherent approximation of treating a product as a flow, the depth of model detail, credibility, and absolute level of accuracy are less than for Discrete Event Simulation.

Queuing Theory enables a reasonably accurate model to be constructed relatively quickly. The predominant concerns are that performance measures are given for conditions of steady state system behavior, and that the inherent approximations restrict the depth and breadth to which a manufacturing system can be modelled. Credibility is weak because graphical animation cannot be provided.

Table A1.1. Example of a resource and some relevant states

Resource (Entity)	States, Levels or Stocks (Queue)	Rate Variable (Activity)
Product	Raw material, work in progress, finished product stock, stock in storage, etc.	Machine

Activity Based Costing is focused at providing product cost, and has the flexibility to assess a range of strategic developments in terms of this measure. Contrary to some evidence in the literature, for example in Ref. 21, this modelling technique can be applied in a reasonably short amount of time if restricted to addressing strategic issues within a manufacturing company.

Business Planning provides a business perspective of manufacturing developments. The predominant weaknesses are that valid flexibility and credibility are limited because the manufacturing system characteristics are only superficially considered.

$IDEF_0$ is a strong mechanism for illustrating the activities in a system, and their interactions, at an instance in time. The principal limitations observed were based on an integration with mathematical modelling within the tool DESIGN/IDEF. Experimentation revealed distinct limitations with the mathematical, and hence predictive, capabilities of this modelling approach.

Integrated Enterprise Modelling provides a model that is less abstract than $IDEF_0$, as the modelling syntax was more easily related to the real system. As a consequence, models are better understood, accepted, and model building is faster. However, this relaxation in syntax does mean that a $IDEF_0$ model provides a system description that is richer in information.

References

1. R. L. Ackoff and M. W. Sasieni, *Fundamentals of Operations Research* (John Wiley and Sons, 1968).
2. K. R. Andrews, *The Concept of Corporate Strategy* (Dow Jones-Irwin, Homewood, Illinois, 1971).
3. H. I. Ansoff (ed.), *Business Strategy* (Penguin Books Ltd., England, 1969).
4. H. I. Ansoff and R. G. Brandenburg, A language for organization design: Part I, *Management Science* **17**, 12 (1971) B705–B716.
5. S. Axsater, Control theory concepts in production and inventory control, *International Journal of Systems Science* **16**, 2 (1985) 161–169.
6. R. W. Baines and D. R. Hughes, Evolutionary design of computer integrated manufacturing systems, *Proceedings of International Conference on Production Research*, Nottingham, 1985.
7. R. W. Baines and G. J. Colquhoun, An integration and analysis tool for engineers, *Assembly Automation*, August, 1990, 141–145.

8. R. W. Baines and G. J. Colquhoun, A generative IDEF0 model of process planning. *International Journal of Production Research* **29**, 11 (1991) 2239–2257.
9. O. Balci, Guidlines for successful simulation studies, *Proceedings of the 1990 Winter Simulation Conference*, eds. O. Balci, R. P. Sadowski and R. E. Nance, 1990, 25–32.
10. S. Banerjee and A. Basu, Model type selection in an integrated DSS environment, *Decision Support Systems* **9** (1993) 75–89.
11. R. R. Brovoco and S. B. Yadav, A methodology to model the functional structure of an organisation, *Computers in Industry* **6** (1985) 345–361.
12. R. R. Brovoco, S. B. Yadav, A. T. Chatfield and T. M. Rajkumar, Comparison of analysis techniques for information requirement determination, *Communications of the ACM* **31**, 9 (1988) 1090–1097.
13. E. S. Buffa, Meeting the competitive challenge with manufacturing strategy, *National Productivity* **4**, 2 (1985) 155–169.
14. A. Carrie, *Simulation of Manufacturing Systems* (John Wiley and Sons, 1988).
15. J. A. Cecil, K. Srihari and C. R. Emerson, A review of petri-net applications in manufacturing, *International Journal of Advanced Manufacturing Technology*, **7** (1992) 168–177.
16. A. D. Chandler Jr., *Strategy and Structure: Chapters in the History of the American Industrial Enterprise* (The MIT Press, Cambridge, MA, 1962).
17. P. S. Cheema, L. L. Coleman, B. Bishop and J. S. Edghill, A combined feedforward/feedback "to make" model for a multi-product machine shop. *5th National Conference on Production Research*, Huddersfield Polytechnic, 6–8 September, 1987, 101–105.
18. P. B. Checkland, Soft systems methodology: An overview, *Journal of Applied Systems Analysis* **5** (1988) 27–30.
19. P. B. Checkland and J. Scholes, *Soft Systems Methodology in Action* (John Wiley and Sons, Chichester, England, 1990).
20. S. Cooke and N. Slack, *Making Management Decisions* (Prentice Hall, 1991).
21. R. Cooper, Does your company need a new cost system? *Journal of Cost Management*, Spring 1990, 45–49.
22. W. Copacino and D. B. Rosenfield, Analytical tools for strategic planning, *International Journal of Physical Distribution and Materials Management* **15**, 3 (1985) 47–61.
23. R. M. Currie, *Work Study* (Pitman Publishing, London, 1959).
24. B. Dangerfield and C. Roberts, *Model-Based Scenarios of the Consequences of Capacity Retirements in the Steel Industry*, Department of Business and Management Studies, University of Salford, Working Paper No. 9311, 1993.
25. H. Danzyger, Strategic manufacturing plan? Your competitiveness depends on it, *Industrial Engineering* (1990) 19–23.
26. E. Downs, P. Clare and I. Coe, *Structured Systems Analysis and Design Method Application and Context* (Prentice Hall, London, 1992).
27. DTI, *Competetive Manufacturing: A Practical Approach to the Development of a Manufacturing Strategy* (IFS Publications, Bedford, England, 1988).
28. J. S. Edghill, D. R. Towill and M. Jones, A combined approach for the study of industrial dynamic systems, *Modern Science Management Systems*, ed. A. Kusiak (Elsevier Science Publishers B.V., Amsterdam, 1987).
29. H. A. ElMaghraby and T. Ravi, Modern tools for the design, modelling and evaluation of flexible manufacturing systems, *Robotics and Computer-Integrated Manufacturing* **9**, 4–5 (1992) 335–340.
30. R. Evered, So what is strategy? *Long Range Planning* **16**, 3 (1983) 57–72.

31. C. H. Fine and A. C. Hax, Manufacturing strategy: A methodology and an illustration, *Interfaces* **15**, 6 (1985) 28–46.
32. P. A. Fishwick, Neural network models in simulation: A comparison with traditional modeling approaches, *Proceedings of the 1989 Winter Simulation Conference*, eds. E. A. MacNair, K. J. Musselman and P. Heidelberger, 1989.
33. J. W. Forrester, Industrial dynamics: A major breakthrough for decision makers, *Harvard Business Review* **36**, 4 (1958).
34. J. W. Forrester, *Industrial Dynamics* (MIT Press, Cambridge, MA, USA, 1961).
35. J. W. Forrester, Industrial dynamics — After the first decade, *Management Science* **14**, 7 (1968) 398–415.
36. R. F. Garzia, M. R. Garzia and B. P. Zeigler, Discrete-event simulation, *IEEE Spectrum*, December 1986.
37. N. Graham, *Learning C++* (McGraw-Hill Inc., New York, 1991).
38. G. R. Greenhalgh, *Manufacturing Strategy — Formulation and Implementation* (Addison-Wesley Publishers Ltd., Sydney, Australia, 1990).
39. R. H. Hayes and S. C. Wheelwright, *Restoring Our Competitive Edge: Competing Through Manufacturing* (John Wiley and Sons, New York, 1984).
40. A. C. Hax and N. S. Majluf, *Strategic Management: An Integrative Perspective* (Prentice-Hall Inc., Englewood Cliffs, NJ, 1984).
41. J. Heizer and B. Render. *Production and Operations Management: Strategies and Tactics* (Allyn and Bacon, Boston, 1988).
42. T. J. Hill, *Manufacturing Strategy — The Strategic Management of the Manufacturing Function* (MacMillan Education Ltd., Hampshire, England, 1985).
43. C. W. Hofer and D. Schendel, *Strategy Formulation: Analytical Concepts* (West Publishing Co., St. Paul, Minnesota, 1978).
44. C. Hogg, A. W. Self, D. F. Pearce and D. Wilson, Steps towards total flight simulation, *Proceedings of the 8th International Conference on Systems Engineering*, 1991, 876–889.
45. H. J. Johansson, P. McHugh, A. J. Pendlebury and W. A. Wheeler, *Business Process Reengineering* (John Wiley and Sons, Chichester, England, 1993).
46. R. S. Kaplan, Yesterday's accounting undermines production, *Harvard Business Review*, July/August (1984) 95–101.
47. D. F. Kehoe, D. Little and T. M. Wyatt, An approach to FMS design for improved integration, *Proceedings of the 3rd International Conference on Simulation in Manufacturing*, November 1987, 263–278.
48. A. M. Law and W. D. Kelton, *Simulation Modeling and Analysis* (McGraw-Hill, Inc., Singapore, 1991).
49. D. Love and J. Barton, Using whole business simulation to set operational policies in an integrated manufacturing system, *The International Conference on Managing Integrated Manufacturing — Organization, Strategy and Technology*, 22–24th September 1993. The University of Keele, UK., 1993, 145–157.
50. LUCAS, *The LUCAS Manufacturing Systems Engineering Handbook* (Institution of Production Engineers, London, England, 1989).
51. D. A. Marca and C. L. McGowan, *Structured Analysis and Design Technique* (McGraw-Hill Publishing Company, New York, 1986).
52. K. Mertins, W. Sussenguth and R. Jochem, Integrated information modelling for CIM: An object-oriented method for integrated enterprise modelling, *Computer Applications in Production and Engineering: Integration Aspects*, eds., G. Doumeingts, J. Brown and M. Tomljanovich (Elsevier Science Publishers BV, North-Holland, 1991) 315–323.

53. G. A. Mihram, The modeling process, *IEEE Transactions on Systems, Man and Cybernetics* **2**, 5 (1972) 621–629.
54. E. D. Minor III, R. L. Hensley and D. R. Wood Jr., A review of empirical manufacturing strategy studies, *International Journal of Operations and Production Management* **11**, 1 (1994) 5–25.
55. H. Mintzberg, Patterns in Strategy Formation, *Management Science* **9** (1978) 934–948.
56. H. Mintzberg, D. Raisinghani and A. Theoret, The structure of "unstructured" decision processes, *Administrative Science Quarterly* **21** (1976) 246–275.
57. H. Mintzberg, *The Rise and Fall of Strategic Planning* (Prentice Hall, Hemel Hempstead, England, 1994).
58. C. New, World class manufacturing versus strategic trade offs, *Proceedings of the 6th International Conference of the Operations Management Association (UK)*, June 1991, University of Aston, UK, 1–16.
59. R. Newton, A review of activity-based costing systems, *Management Consultancy* March (1991).
60. C. M. G. Olsmats, J. S. Edghill and D. R. Towill, Industrial dynamics model building of a close-coupled production-distribution system, *Engineering Costs and Production Economics* **13** (1988) 295–310.
61. S. B. O'Reilly, K. W. Casey and S. A. Weiner, On the use of physical models to simulate assembly plant operations, *Proceedings of the 1984 Winter Simulation Conference*, eds. S. Sheppard, U. Pooch and D. Pegden, 1984.
62. A. J. Pendlebury, Creating a manufacturing strategy to suit your business, *Long Range Planning* **20**, 6 (1987) 35–44.
63. M. Pidd, *Computer Simulation in Management Science* (John Wiley and Sons, 1988).
64. R. M. Pirsig, *Zen and the Art of Motorcycle Maintenance* (Vintage, London, England, 1974).
65. K. W. Platts, *Manufacturing Audit in the Process of Strategy Formulation.* PhD Thesis, University of Cambridge, 1990.
66. K. Popplewell and M. C. Bonney, The application of discrete linear control theory to the analysis and simulation of multi-product, multi-level production control systems, *International Journal of Production Research* **25**, 1 (1987) 45–56.
67. M. E. Porter, *Competitive Strategy — Techniques for Analyzing Industries and Competitors* (The Free Press, New York, 1980).
68. A. A. B. Pritsker, *Papers — Experiences — Perspectives* (Systems Publishing Corporation, Indiana, USA, 1990).
69. J. B. Quinn, Strategic change: "Logical incrementalism", *Sloan Management Review* (1978) 7–21.
70. J. L. Riggs, *Production Systems: Planning, Analysis, and Control* (John Wiley and Sons, New York, 1970).
71. P. F. Roth, Discrete, continuous and combined simulation, *Proceedings of the 1987 Winter Simulation Conference*, eds. A. Thesen, H. Grant and W. D. Kelton, 1987, 25–29.
72. D. Saunders, C. N. Bancroft and M. F. Croockett, Linking simulation to an emulation facility, *Proceedings of the 8th International Conference on Systems Enginering*, 1991, 692–698.
73. D. Schmidt, J. Haddock and W. A. Wallace, Modeling the manufacturing process: An assessment of three approaches to simulation, *1991 International Industrial Engineering Conference Proceedings*, Detroit, 1991.

74. J. W. Schmidt, Introduction to systems analysis, modeling and simulation, *Proceedings of the 1985 Winter Simulation Conference*, eds. J. Wilson, J. Henriksen and S. Roberts, 1985.
75. R. E. Shannon, *Systems Simulation* (Prentice Hall, 1975).
76. W. Skinner, Manufacturing — Missing link in corporate strategy, *Harvard Business Review* (1969) 136–145.
77. W. Skinner, *Manufacturing — The Formidable Competitive Weapon* (John Wiley and Sons, New York, 1985).
78. D. Steeple and C. Winters, A review of alternative approaches to traditional costing methods, *Advances in Manufacturing Technology VII: Proceedings of the Ninth National Conference on Manufacturing Research*, eds. A. Bramley and T. Mileham, September 1993, University of Bath, England, 1993, 196–200.
79. R. Suri and G. W. Diehl, A precursor to simulation for complex manufacturing systems, *Proceedings of the 1985 Winter Simulation Conference*, eds. D. Gantz, G. Blais and S. Solomon, 1985, 411–420.
80. H. Thomas and C. R. Schwenk, Decision analysis as an aid to strategy, **MD22**, 2 (1984) 50–60.
81. A. A. Thompson Jr. and A. J. III. Strickland, *Strategy Formulation and Implementation — Tasks of the General Manager* (Business Publications, Inc, Dallas, Texas 1980).
82. R. Thompson, Understanding cash flow: A systems dynamics analysis, *Journal of Small Business Management* April (1986) 23–30.
83. D. R. Towill, System dynamics — Background, methodology, and applications, Part 1: Background and methodology, *Computing and Control Engineering Journal*, October (1993a) 201–208.
84. D. R. Towill, System dynamics — Background, methodology, and applications, Part 2: Applications, *Computing and Control Engineering Journal* December (1993b) 261–268.
85. G. Williams and T. Pontin, Machine shop loading — A planning board approach, *Proceedings of the 5th International Conference on Simulation in Manufacturing*, 13–14th June 1989, Berlin, Germany, 1989, 111–121.
86. E. F. Wolstenholme, *System Enquiry: A System Dynamics Approach* (John Wiley Inc., Chichester, England, 1990).
87. A. Ziya Aktas, *Structured Analysis and Design of Information Systems* (Prentice-Hall Inc., 1987).

CHAPTER 4

ECONOMIC OPTIMIZATION OF MACHINING OPERATIONS IN COMPUTER AIDED MANUFACTURING SYSTEMS

JUN WANG

School of Mechanical, Manufacturing and Medical Engineering,
Queensland University of Technology,
GPO Box 2434, Brisbane,
Queensland 4001, Australia

A manufacturing firm must be competitive as assessed by the level of profit, both locally and on a global basis in order to survive. About 40% of the selling price of a product can be manufacturing costs, and thus maintaining a high level of profit depends on reducing manufacturing costs. For this reason, the manufacturing industry has led the revolution in production technology. This has resulted in the development of highly effective Computer Aided Manufacturing (CAM) techniques which are treated rather comprehensively in this chapter.

Keywords: Computer aided manufacturing systems; manufacturing costs; machining operations; computer numerical controlled (CNC) machine tools.

1. Introduction

Manufacturing is one of the major wealth-generating activities that creates the economic basis for genuine improvements in the quality of human life. A manufacturing firm must be competitive, as assessed by the level of profit, both locally and on a global basis in order to survive. Since about 40% or so of the selling price of a product is manufacturing cost,[10] maintaining a high level of profit depends on reducing manufacturing costs. For this reason, the manufacturing industry has led the revolutions in production technology as represented by the low level of automation coined at Ford's in the 1930's and programmable automation in modern Computer Aided Manufacturing (CAM) systems. It is anticipated that the advances in manufacturing technology will lead to the increase in productive (machining) times from about 5% for manual machines to about 75% in CAM systems involving Computer Numerical Controlled (CNC) machine tools.[13,22] This trend suggests that any future economic gains will have to be achieved from optimizing the various manufacturing activities or processes. The efficient and economic use of the manufacturing facilities is also necessary to offset the high investment costs in the use of

the advanced technologies and the operating costs. Of the manufacturing processes, machining is often claimed to be the most important process since a vast majority of the manufactured products require machining at some stage in their production. Consequently, efficient machining operations with lowest possible machining costs and/or highest possible production rate is ultimately the key issue of competitive manufacturing.

The optimization of machining operations or machining economics generally refers to making use of production resources most efficiently and at lowest possible costs. The variables that affect the economics of machining operations are numerous and include materials, people and equipment. In the economics of machining study, it is common to assume that the materials have been properly selected in terms of shape, dimension and properties and purchased at the lowest possible price. Similarly, the people or operators have been well trained and perform in the most economic manner. Thus, the selection and use of machine tools, cutting tools and cutting conditions become the focal points in the economic optimization of machining operations. In fact, the notion of economics of machining was recognized as early as 1907 by Taylor.[29] He noted that the selection of "low" cutting speeds in single pass turning operations resulted in undesirably low average production rates (and materials removal rates) while "high" cutting speeds were equally undesirable due to the reduced tool-life and more frequent interruption times for tool replacement. As a consequence, Taylor recognized the existence of an "optimum" or "economic" cutting speed which maximized the average production rate and was highly dependent on the tool-life—cutting speed relationship. Later researchers realized that, in addition to the cutting speed, other cutting conditions such as the feed and depth of cut have to be optimized in order to gain full economic benefits.

In a real manufacturing situation, a sequence of machining operations is normally required to make a component. Consequently, the traditional selection of economic cutting conditions for a specific operation is a sub-rather than a global optimization of the overall manufacturing system. It has been demonstrated by Ravignani,[26] however, that the cutting conditions giving optima of machining time, cost or profit rate for each specific operation yield optima of these quantities regardless of the sequence of the different operations. Thus, the optimization of any particular operation, such as turning, milling and drilling, is an essential step toward the optimization of the manufacturing systems. Furthermore, the full optimization of a manufacturing system should consider process or operation mix, production rate at each manufacturing stage, process interaction, different values of operations at different production rates, inventory, and possible variation in anticipated sales etc. Due to the complexity associated with it, a full optimization is rarely attempted, although some work has been reported on multi-stage optimization involving a number of machine tools or machining operations for processing a component.[25] Again, a procedure that is often used is to select the economic cutting conditions for each

machining operation. If necessary, these conditions may be modified after reviewing all other factors in the system.

Thus, this chapter will focus on only the optimization of cutting conditions in single- and multi-pass machining operations. The optimization of turning operations will be used to illustrate the development of optimization strategies, but the approach is generic and applicable to other operations, such as milling and drilling. With an optimization strategy, the most appropriate machine tool and cutting tool for a selected machining job can be determined on a case-by-case basis. This will be shown using a numerical case study.

2. Optimization of Cutting Conditions in Machining Operations

Traditionally, the optimization of machining operations involves the selection of economic cutting conditions, such as the feed and cutting speed, according to a variety of economic criteria such as the minimum cost per component, maximum production rate or maximum profit rate (maximum profit per unit time).[2,3,39] Studies have also been undertaken to concurrently consider multiple criteria.[9,20,21] As it has been proven that the optimum cutting conditions for one criterion are not normally the same as those for other criteria, a weighting factor is normally introduced to each criterion to determine its relative importance and for compromising. A realistic optimization study should also take into account the many technological and practical constraints which limit the feasible domain for the selection of optimum cutting conditions. These constraints are imposed by the elements of the machining system and include:

(1) Machine tool constraints:

- Maximum power
- Maximum spindle torque and low speed power
- Maximum forces
- Maximum and minimum speeds
- Maximum and minimum feeds
- Speed steps (for conventional machine tools)
- Feed steps (for conventional machine tools)

(2) Component constraints:

- Surface finish which in most cases can be represented by a feed limit or feed and speed limits
- Dimensional accuracy (may be represented by force limit)

(3) Cutting tool constraints:

- Depth of cut limit
- Force limit
- Maximum feed limit

(4) Machining system constraints:

- System stability

In addition, the manufacturing system may impose constraints on the machining operation, such as the balance of machine loading time, and reduction of work-in process (WIP) inventory. To consider the constraints in the optimization of cutting conditions has proved to be surprisingly difficult. It requires intricate mathematical analyses and computer assistance, and depends on quantitatively reliable mathematical functions of machining performance measures (such as tool-life, cutting forces, power, surface finish) and detailed specifications of the machine tools, cutting tools and components which act as constraints on the permissible cutting conditions.[33] This difficulty has resulted in some researchers using the available computer-aided mathematical programming and numerical search techniques in attempts to provide the optimum feed and speed in various practical machining operations.[11,12,15,19,23] These computer packaged strategies neither guarantee global optimum solutions nor provide clearly defined economic characteristics and solution strategies which allow for the ready identification of trends in the way in which the optimum solution can change with alternative constraints. In addition, these purely numerical search approaches require excessive long computer processing time and are not suitable for on-line application in CAM systems. As such, a number of other approaches have been developed or used for the optimization of cutting conditions.

The use of a cutting rate-tool-life characteristic function (R-T-F) for machining optimization was introduced by Friedman and Tipnis,[14] Tipnis and Friedman[30] and Ravignani *et al.*[27] They show that the tangent points of the constant tool-life curves and the constant cutting rate curves in the cutting speed-feed (or $\log V - \log f$) domain represent the optimum combination of the cutting rate and tool-life. The locus of these tangent points forms the R-T-F curve from which the optimum feed and cutting speed could be obtained.

The R-T-F approach has been successfully used to optimize the cutting conditions for situations where practical constraints were not considered. This has limited its application in developing realistic optimization strategies. In addition, more cutting variables, such as the feeds, cutting speeds and depths of cut in multipass machining, will have to be optimized in practice and it has not been shown how this approach can be applied to such complicated cases.

Artificial neural networks (ANN) are among the many other techniques that have been employed to estimate the optimum cutting conditions in machining.[16,24,34] However, it has been demonstrated that a deterministic approach which has been successfully applied to the various practical machining operations[4,6,7,18,36–38] has a number of distinct advantages over the others. It can provide clearly defined economic characteristics and solution strategies. More importantly, the resulting strategies can ensure a unique global optimum solution for a set of input conditions and are suitable for on line application in CAM systems.

Consequently, the following sections will demonstrate how the deterministic optimization approach is used in developing the strategies for selecting the economic cutting conditions in machining operations. For this purpose, turning operations will be used to illustrate the procedure, but the approach is fully applicable to the other practical machining operations such as milling and drilling. The analysis will be based on economic criteria typified by the maximum production rate and minimum cost per component and include a number of common practical constraints relevant to CNC machine tools. More practical constraints can always be introduced once the mathematical functions and the associated data become available. The single pass operation will be considered first, which will form an essential step in developing the optimization strategies for multipass operations where the surface finish requirement will need to be considered only in the final pass. A numerical study will finally be given to show the use of the optimization strategies in the selection and design of the most appropriate machine tools for a particular operation and a manufacturing environment.

3. Optimization of Single Pass Machining Operations

3.1. *Objective functions*

Based on the maximum production rate (or the minimum average production time per component) criterion, the objective function for a single pass machining operation can be given by the well-known equation[2]:

$$T_T = T_L + T_c + T_R \frac{T_{ac}}{T} \tag{1}$$

where T_L is the workpiece loading and unloading time, T_{ac} is the actual cutting time, T_c is the feed engagement time including the actual cutting time and the time for the tool empty travel for pre- and over-run outside the workpiece, T_R is the tool replacement time per tool failure, and T is the tool-life in time units. Thus, the third term in the equation represents the average tool replacement time per component.

Similarly, the objective function for the average cost per component criterion is

$$C_T = x\left(T_L + T_c + T_R \frac{T_{ac}}{T}\right) + y\frac{T_{ac}}{T} \tag{2}$$

where x is the labour and overhead cost rate, and y is the tool cost per failure (e.g. the cost per cutting edge for an indexable insert).

If a term T'_R is introduced, such that

$$T'_R = T_R + \frac{y}{x} \tag{3}$$

Equation 3 becomes

$$C_T = x\left(T_L + T_c + T'_R \frac{T_{ac}}{T}\right) \tag{4}$$

It can be noted that if the labour and overhead cost rate, x, and the tool cost per failure, y, are minimized and constant through good management and purchasing policy, Eqs. 1 and 4 are mathematically similar. Hence, the characteristics and strategies for minimizing T_T and C_T are similar although the optimum feed and speed for the two criteria are not necessarily the same under the same constraint conditions. Thus, only the analyses for the minimum time per component T_T equation will be explicitly presented based on turning operations.

Despite the dearth of tool-life data in machining operations noted in the literature, two comprehensive machining data handbooks[1] have been found to provide the tool-life function as well as the values of the empirical constants for a range of tool-workpiece material combinations suitable for economic machining studies. The tool-life was given in the typical extended Taylor-Type equation

$$T = \frac{K}{V^{1/n} f^{1/n_1} d^{1/n_2}} \tag{5}$$

where V, f and d are the cutting speed, feed per revolution and depth of cut respectively, while K is a constant accounting for the other variables. The cutting time T_c for turning of a length ℓ can be approximately expressed as:

$$T_c \approx T_{ac} = \frac{\ell D}{\mu V f} \tag{6}$$

where μ is the factor for conversion from rotational speed to surface speed. Substituting Eqs. 5 and 6 into Eq. 1 gives

$$T_T = T_L + \frac{\ell}{\mu V f} + \frac{\ell T_R}{\mu K} V^{1/n-1} f^{1/n_1-1} d^{1/n_2} \tag{7}$$

This is the fundamental form of the objective function which has to be optimized. As is usual in single pass optimization studies, only the cutting speed V and feed f have to be optimized, since it is expected that the loading/unloading time T_L and cutter replacement time T_R have been minimized using work study techniques and well-designed handling devices.

3.2. *Technological constraints*

In practice, the cutting speed V and feed per revolution f must be selected to minimize T_T in Eq. 7 without violating a number of constraints, such as those imposed by the machine tool, which in fact limit the feasible domain of speed V and feed f and result in a constrained optimum T_T. For a single pass turning operation on a CNC machine tool, the machine tool limiting force, $F_{\mathrm{p\,max}}$, spindle torque, $T_{\mathrm{q\,max}}$, maximum power, $P_{\max}$, as well as the feed and spindle speed boundaries ($f_{\min}$, $f_{\max}$, $N_{\min}$, $N_{\max}$) are considered. The component surface roughness requirement will also be included in finishing operations. These constraints can be expressed mathematically as follows.

3.2.1. *Machine tool speed and feed boundary constraints*

For CNC machine tools, any feed and spindle speed within the specified minimum and maximum limits may be considered to be available for the selection of optimum cutting conditions. Mathematically, these constraints are given by

$$\pi D N_{\min} = V_{\min} \leq V \leq V_{\max} = \pi D N_{\max} \tag{8}$$

$$f_{\min} \leq f \leq f_{\max} \tag{9}$$

3.2.2. *Machine tool power force constraint*

The power force limit is imposed by the machine tool mechanism, such as the spindle and tool post, and has to be constrained to within the machine tool maximum permissible loading. In addition, excessive machining forces will cause the machining system deformation affecting the component quality. Using the empirical power force equation in the handbook,[1] this condition can be expressed as:

$$F_p = E f^{\alpha} d^{\beta} \leq F_{p\max} \tag{10}$$

Thus, the maximum power force constraint will result in a feed limit, i.e.

$$f \leq f_F = \left(\frac{F_{p\max}}{E d^{\beta}} \right)^{1/\alpha} \tag{11}$$

3.2.3. *Machine tool maximum power and torque constraints*

The cutting conditions selected must satisfy the condition that the machining power is within the machine tool maximum power limit $P_{\max}$. In the low-speed region of a machine tool operating range, the machine tool maximum power may not be permitted since this would involve an excessive spindle torque. In this region, the "low" speed power constraint P_a due to the limiting spindle torque must be considered. This lower spindle power usually increases linearly with the speed until a critical speed V_a where the machine tool maximum power constraint $P_{\max}$ becomes relevant. Hence the combined torque (or low speed power P_a) and the maximum power constraints can be expressed as:

$$P = W V f^{\alpha} d^{\beta} \leq P_a = A_1 N = A V \quad (\text{for } V \leq V_a) \tag{12}$$

$$P = W V f^{\alpha} d^{\beta} \leq P_{\max} \quad (\text{for } V > V_a) \tag{13}$$

It is common that V_a has a constant value (dependent on the workpiece diameter) between the minimum and maximum machine tool speed limits and can be found from the machine tool specification. At $V = V_a$, $P_a = P_{\max}$ so that V_a can be found

to be:

$$V_a = \frac{P_{\max}}{A} \tag{14}$$

In addition, the low speed power constraint can be represented by a feed limit f_a that can be found by re-arranging Eq. 12, i.e.

$$f \leq f_a = \left(\frac{A}{Wd^{\beta}}\right)^{1/\alpha} \tag{15}$$

In contrast, the maximum power constraint $P_{\max}$ will limit both the feed and speed when Eq. 13 is satisfied.

3.2.4. *Component surface roughness constraint*

Although some theoretical surface roughness equations have been reported, there is a general lack of published data to support these equations so that this constraint cannot be accurately allowed for in the machining optimization until reliable surface roughness equations and associated data become available. For the purpose of the present work, the theoretical or ideal peak-to-valley height equation given by Armarego and Brown[2] will be employed and the resulting constraint expression is given by

$$f \leq f_{Rt} = R_{t\max}(\tan\psi_r + \cot\kappa'_r) \tag{16}$$

In the above constraint equations, E, W, α and β are empirical constants dependent on the tool-work material combination and can be found in the comprehensive machining data handbook,[1] $F_{\mathrm{p}\max}$, $P_{\max}$, $N_{\min}$, $N_{\max}$, $f_{\min}$ and $f_{\max}$ are constraints given in the machine tool specifications, $R_{\mathrm{t}\max}$ is the maximum surface roughness (peak-to-valley) limit, and ψ_r and κ'_r are respectively the approach angle and minor cutting edge angle of the cutting tool.

It is evident that the magnitudes of these constraints limit the permissible domain for the optimization of the cutting speed V and feed f in Eq. 7. Furthermore, the machine tool low speed power (or torque) and power force constraints as well as the component surface roughness constraint, which only limit the permissible feed and are mutually exclusive, can be generalized by a feed limit f_x, i.e.

$$f \leq f_x = \min\{f_F, f_a, f_{Rt}\} \tag{17}$$

For rough turning, Eq. 17 can be simplified as

$$f \leq f_x = \min\{f_F, f_a\} \tag{18}$$

A detailed study of the machining performance data in the handbook[1] has found that the exponents in the tool-life and the constraint equations have the following relationships: $1/n > 1/n_1 > 0$, $1/n > 1$ and $1 > \alpha > n/n_1$ while $1/n_1$ can be greater than, equal to or less than 1. Based on these common ranges of the machining performance exponents, the optimization strategy will be developed below.

3.3. *Optimization analysis and strategy*

A global minimum time per component (maximum production rate) requires that the partial derivatives of the objective function in Eq. 7 with respect to the cutting speed and feed are zero, i.e.

$$\frac{\partial T_T}{\partial V} = \frac{\ell}{\eta V^2 f}\left[\frac{T_R}{T}\left(\frac{1}{n} - 1\right) - 1\right] = 0 \tag{19}$$

$$\frac{\partial T_T}{\partial f} = \frac{\ell}{\mu V f^2}\left[\frac{T_R}{T}\left(\frac{1}{n_1} - 1\right) - 1\right] = 0 \tag{20}$$

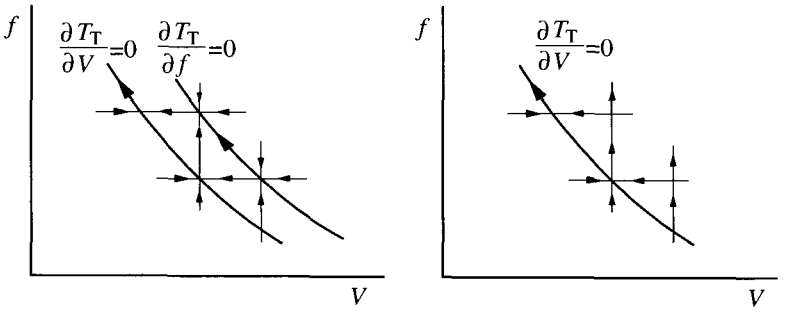
from which the economic tool-life equations with respect to the cutting speed and feed can be found and are respectively given by

$$\frac{K}{V^{1/n} f^{1/n_1} d^{1/n_2}} = T_R\left(\frac{1}{n} - 1\right) = T_V \tag{21}$$

$$\frac{K}{V^{1/n} f^{1/n_1} d^{1/n_2}} = T_R\left(\frac{1}{n_1} - 1\right) = T_F \tag{22}$$

Since $n \neq n_1$ for the common tool-work material combinations, Eqs. 19 and 20 (or Eqs. 21 and 22) cannot be simultaneously satisfied for a minimized T_R. Hence a unique pair of V and f does not exist for a global minimum time per component T_T. Therefore it is necessary to study the T_T characteristics in order to establish a strategy for selecting the V and f such that the production time per component is minimized.

Figure 1(a) illustrates the $\partial T_T/\partial V = 0$ and $\partial T_T/\partial f = 0$ loci on an f–V graph. It has been proved[33] that for the usual values of the empirical tool-life equation exponents $1/n > 1/n_1 > 1$, the $\partial T_T/\partial f = 0$ curve is above and to the right of the $\partial T_T/\partial V = 0$ curve on the $f - V$ diagram. Further, a local optimum T_T with respect to V always exists for a given f, since $1/n > 1$, which can be found on the curve described by Eq. 21, i.e. the $\partial T_T/\partial V = 0$ curve. Similarly, the optimum feed for a local minimum T_T can be obtained from Eq. 22 (on the $\partial T_T/\partial f = 0$ curve) for a given cutting speed V when $1/n_1 > 1$.



(a) $1/n > 1/n_1 > 1$ (b) $1/n > 1 \geq 1/n_1 > 0$

Fig. 1. Diagrammatic presentation of time per component characteristics.

The time per component T_T characteristics along the $\partial T_T/\partial V = 0$ locus can be found by substituting Eq. 21 into Eq. 7 from which

$$T_T = T_L + \frac{\ell}{\mu}\left[\frac{T_R}{nK}\right]^n (1-n)^{n-1} f^{n/n_1-1} d^{n/n_2} \tag{23}$$

Thus, since $n/n_1 < 1$, T_T will decrease along the $\partial T_T/\partial V = 0$ curve as f increases (or V decreases), as indicated by the arrowheads in Fig. 1(a). It can also be proved that T_T along the $\partial T_T/\partial f = 0$ locus (when $1/n_1 > 1$) possesses similar characteristics to those of the $\partial T_T/\partial V = 0$ curve, as shown in Fig. 1(a).

However, when $1/n > 1$ but $1/n_1 \leq 1$, as is possible for some tool-work material combinations noted in the handbook,[1] $\partial T_T/\partial f$ in Eq. 20 is negative and Eq. 22 does not apply. Thus the necessary condition for a local minimum with respect to f (i.e. $\partial T_T/\partial f = 0$) can never be satisfied and the minimum T_T for a given V occurs when f is as high as possible, as shown in Fig. 1(b). By contrast, Eqs. 19 and 21 still apply from which the optimum cutting speed can be found for a given f. It can be shown again that the T_T value decreases along the $\partial T_T/\partial V = 0$ locus as f increases (or V decreases).

The above T_T characteristics lead to the popular strategy of selecting V and f in the "high feed-low speed" region in the vicinity of the $\partial T_T/\partial V = 0$ and $\partial T_T/\partial f = 0$ (when $1/n_1 > 1$) loci. However, this strategy is not always valid in selecting the optimum cutting speed and feed, since in practice a number of technological and practical constraints, such as those noted above, have to be satisfied. The effects of groups of related constraints on the selection of optimum cutting conditions are considered separately below before developing the overall strategy allowing for the combined effect of all the constraints.

3.3.1. *Effects of machine tool feed and speed boundary constraints*

For CNC machine tools, the minimum and maximum feed limits define an available feed-speed domain with the upper and lower boundaries occur at $f_{\max}$ and $f_{\min}$, respectively. The minimum and maximum spindle speeds establish the cutting speed boundaries, for a given workpiece diameter, as vertical lines on the f–V diagram.

The T_T trends along the horizontal $f_{\min}$ and $f_{\max}$ boundaries, as well as the vertical $V_{\min}$ and $V_{\max}$ boundaries can be established readily by superimposing the $\partial T_T/\partial V = 0$ and $\partial T_T/\partial f = 0$ (when $1/n_1 > 1$) curves on an f–V diagram, as shown in Fig. 2. The T_T value reducing direction is again shown by the arrowheads on the feed and speed boundaries. Based on the T_T characteristics, the minimum T_T value along a feed boundary always occurs at the its intersection with the $\partial T_T/\partial V = 0$ curve. Likewise, when $1/n > 1/n_1 > 1$ the minimum T_T on a cutting speed boundary can be found at its intersection with the $\partial T_T/\partial f = 0$ locus. However, when $0 < 1/n_1 \leq 1$, the $\partial T_T/\partial f = 0$ locus does not exist and T_T will monotonically decease along the constant $V_{\min}$ and $V_{\max}$ boundaries as f increases, according to the characteristics noted earlier. Combining these trends with the T_T characteristics

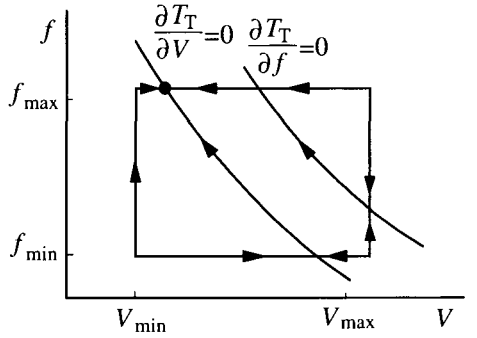


Fig. 2. T_T characteristics and feed and speed boundary constraints.

along the $\partial T_T/\partial V = 0$ and $\partial T_T/\partial f = 0$ loci will arrive at the required optimum solution.

It is apparent that the optimum solution depends on the relative positions of the $\partial T_T/\partial V = 0$ and $\partial T_T/\partial f = 0$ (when $1/n_1 > 1$) loci with respect to the feed and cutting speed boundaries, which in turn depends on the input conditions, i.e. the magnitude and relationship of the machining performance data, the values of the limiting boundaries and the time factors in the T_T equation. From a detailed study of the T_T characteristics when only feed and cutting speed boundary constraints are considered, five possible optimum solutions have been identified and the corresponding limiting conditions established. Figure 2 shows one of them, where the dot highlights the optimum solution.

3.3.2. *Effects of machine tool force and power and component surface roughness constraints*

The machine tool maximum power force, low speed power (or spindle torque) and the component surface roughness constraints have been generalized and represented by an upper feed limit f_x. In the "high" cutting speed region of a machine tool operating range, the machine tool maximum power constraint will come into play and limit both the feed and cutting speed from which a constrained optimum can be selected.

In order to establish an optimization strategy, it is again necessary to study the T_T trends on the f–V diagram while considering the effects of these constraints. As shown in Fig. 3, the maximum power P_{max} and the $\partial T_T/\partial V = 0$ curves intersect such that for the usual set of exponent values, $1 > \alpha > n/n_1 > 0$, the slope of the P_{max} curve is less than that of the $\partial T_T/\partial V = 0$ locus at the point of intersection and so the curves cross in the way shown in the figure. It can be proven that the maximum power constraint curve intersects the $\partial T_T/\partial f = 0$ curve in the same manner as the $\partial T_T/\partial V = 0$ curve on the f–V graph.

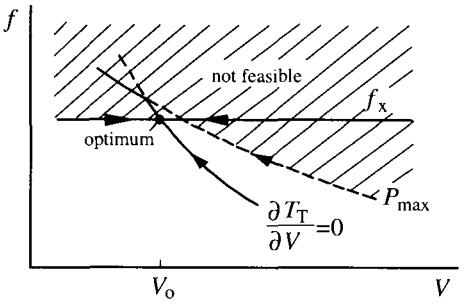


Fig. 3. T_T characteristics and force and power constraints.

The T_T trend along the $P_{\max}$ locus may be found by substituting V from Eq. 13 (with $P = P_{\max}$) into Eq. 7. This shows that for the common conditions of $n/n_1 < \alpha < 1$ and $1/n > 1/n_1 > 0$, T_T decreases along the $P_{\max}$ locus as f increases. This is shown by the arrowheads in Fig. 3. Thus if a feed f is below the intersection of $P_{\max}$ and $\partial T_T/\partial V = 0$ curves, the optimum cutting conditions will be on the $\partial T_T/\partial V = 0$ locus; otherwise, the optimal solution will lie on the $P_{\max}$ curve as the cutting conditions on the $\partial T_T/\partial V = 0$ locus are not feasible. The portions of the $P_{\max}$ and $\partial T_T/\partial V = 0$ curves on which the optimum point is likely to lie are shown by solid lines in Fig. 3.

When both the machine tool power $P_{\max}$ and the generalized upper feed f_x constraints are considered jointly, the T_T characteristics can be found by superimposing the loci of these constraints on the f–V diagram as shown in Fig. 3. The resulting optimum solution will be on the generalized f_x constraint, at its intersection with either the $P_{\max}$ curve or the $\partial T_T/\partial V = 0$ curve, depending on which intersection is at a lower cutting speed. The cutting speeds where the f_x intersects with the $P_{\max}$ curve can be found by substituting f_x into Eq. 13, which gives

$$V_x = \frac{P_{\max}}{W f_x^{\alpha} d^{\beta}} \tag{24}$$

while that at the intersection of f_x and the $\partial T_T/\partial V = 0$ curve can be found by substituting f_x into the following equation which is obtained by re-arranging Eq. 21:

$$V_V(f) = \left[\frac{K}{T_R(1/n-1)f^{1/n_1}d^{1/n_2}}\right]^n \tag{25}$$

Thus there are two possible constrained optimum (f, V) solutions depending on the relative positions of the $\partial T_T/\partial V = 0$ curve and the constraint loci on the f–V diagram. Figure 3 shows one of them.

It should be noted in Eq. 24 that when $f_x = f_a$, $V_x = V_a$ so that V_x is between $V_{\min}$ and $V_{\max}$ due to the normal range of V_a value mentioned earlier. If f_x is not equal to f_a, i.e. greater than f_a, V_x is greater than V_a. Consequently, V_x is greater than the machine tool minimum cutting speed limit $V_{\min}$.

3.3.3. *Optimization strategy for CNC machine tools*

The above study has separately considered small groups of constraints. In practical situations, the economic trends and combined effects of all the constraints have to be considered when machining on CNC machine tools. This results in a more complex strategy which benefits greatly from computer assistance for its implementation after the various possible constrained optimum solutions and the corresponding limits identifying these solutions are established.

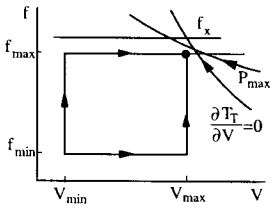
By applying the above analyses, the economic trends for the combined effects of machine tool and component surface roughness constraints can be found by superimposing the upper feed limit, f_x, and the maximum power $P_{\max}$ curve together with the feed and cutting speed boundaries as well as the $\partial T_T/\partial V = 0$ and $\partial T_T/\partial f = 0$ loci on the f–V diagram. Since the relative positions of these curves on the f–V diagram can vary depending on the magnitudes of the constraints, the machining performance data and the cut geometry (depth of cut), the active constraints on which the optimum conditions may lie can also vary. To establish the optimization strategy, it is necessary to identify the various constrained optimum solutions on the "active" constraints, and the associated limiting conditions for each solution.

A detailed mathematical study of the T_T trends on the f–V domain has resulted in 11 distinctly different solutions representing all possible relative positions of the $\partial T_T/\partial V = 0$, $\partial T_T/\partial f = 0$ and the various constraints' loci. These are shown in Fig. 4 where the arrowheads indicate the T_T decreasing direction and the "dot" highlights the optimum feed and speed. The limiting conditions for identifying each constrained optimum f and V solution are given in the captions of each diagram, based on which a computer program can be written to arrive at the required constrained global optimum f and V solution for a minimum T_T (or C_T). The inputs required are machine tool constraints ($f_{\min}$, $f_{\max}$, $N_{\min}$, $N_{\max}$, F_{pmax}, $P_{\max}$, and V_a), component surface requirement (R_{tmax}), cut geometry (d, ℓ), time and cost (if C_T criterion is considered) factors (T_L, T_R, x, y), and machining performance information in the force, power and tool-life equations. Despite the complexity of the constrained optimization strategy, the use of computers has assisted its implementation.

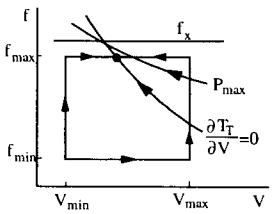
It is noteworthy that among the 11 possible solutions, one represents the situation where a single pass operation is not feasible since at least one of the practical constraints will be violated for the "input" conditions so that multipass milling operations or an alternative machine tool must be considered.

3.4. *Major features of the deterministic optimization approach*

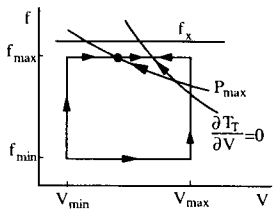
The deterministic machining optimization approach and the resulting strategies described above have a number of distinct advantages over the others. Firstly, despite the complexity of the constrained optimization analyses and strategy requiring computer assistance, the approach assisted by the f–V graphical presentation provides a means of guaranteeing and identifying the various possible global optimal solutions. For a given set of inputs, the strategy will produce a unique optimum



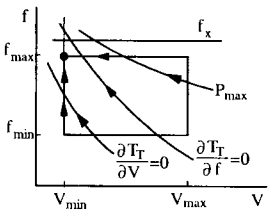
(a) $f_{max} \le f_x$ and $V_{max} \le \min\{V_p(f_{max}), V_v(f_{max})\}$



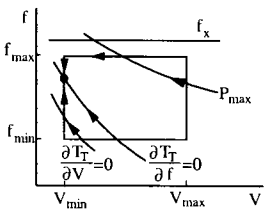
(b) $f_{max} \le f_x$ and $V_{min} < V_v(f_{max}) \le \min\{V_p(f_{max}), V_{max}\}$



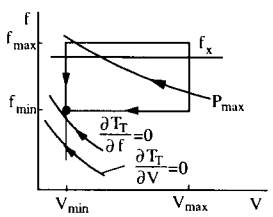
(c) $f_{max} \le f_x$ and $V_{min} < V_P(f_{max}) \le \min\{V_V(f_{max}), V_{max}\}$



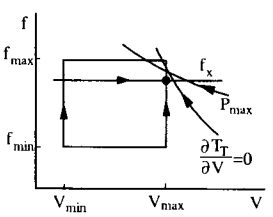
(d) $1/n_1 > 1, f_v(V_{min}) \le f_{max} \le \min\{f_X, f_F(V_{min})\}$ or $1/n_1 \le 1, f_V(V_{min}) \le f_{max} \le f_x$



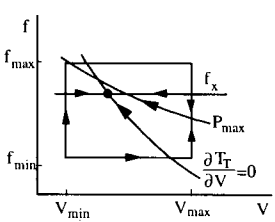
(e) $1/n_1 > 1, f_{min} < f_F(V_{min}) \le \min\{f_x, f_{max}\}$



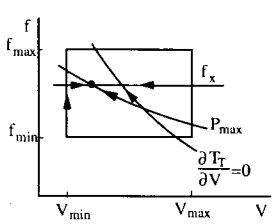
(f) $1/n_1 > 1, f_F(V_{min}) \le f_{min} \le f_X$



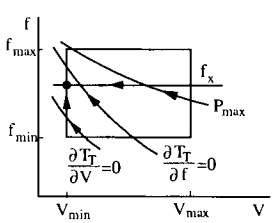
(g) $f_{min} \le f_x < f_{max}$ and $V_{max} \le \min(V_x, V_v(f_x))$



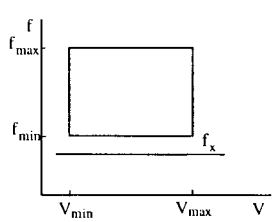
(h) $f_{min} \le f_X < f_{max}$ and $V_{min} < V_v(f_X) \le \min\{V_x, V_{max}\}$



(i) $f_{min} \le f_x < f_{max}$ and $V_{min} < V_X \le \min\{V_v(f_X), V_{max}\}$



(j) $1/n_1 > 1, \max\{f_{min}, f_v(V_{min})\} \le f_{x,} \le \min\{f_F(V_{min}), f_{max}\}$ or $1/n_1 \le 1, \max\{f_{min}, f_v(V_{min})\} \le f_x \le f_{max}$



(k) $f_x < f_{min}$ (single pass is not feasible)

Fig. 4. Various possible optimum (V, f) solutions in single pass turning operations.

feed and speed solution and the corresponding minimum time (or cost) per component. Secondly, the active constraints associated with each global solution can be identified. This information can be used to improve some of the constraints in the machine tool design for increased economic benefits. Thirdly, the resulting strategy

and the associated software involves only a simple sequential testing of the limiting conditions for identifying the appropriate global solution and thus greatly reducing the computer processing time. Numerical studies[37,38] have shown that the computer processing times required are within a hundredth second on a personal computer. Finally, additional technological constraints can always be incorporated into the optimization analysis and the resulting strategies once the mathematical equations (preferably in the empirical form) are available.

4. Optimization of Multipass Machining Operations

It has been evident from the optimization studies for the various practical operations[5,7,32,35] that due to the many constraints to be satisfied in practice, a single pass operation is not always superior to a multipass as might be intuitively assumed during process planning. Under certain constraint conditions, a single pass operation may not be feasible, as noted in the analysis for single pass operations. In such a case, the total depth of cut has to be removed by a number of cuts or passes from the workpiece, i.e. to use a multipass operation. This occurs either because of the severe limiting conditions from the machine tool (or the machining system) for the required operation, or the quality requirement of the component (e.g. surface roughness). In this section, the single pass analysis will be extended to the development of optimization strategies for multipass operations. Again, turning operations will be used to illustrate the optimization procedure while detailed analysis can be found from the work by Chia,[8] Kee[17] and Wang.[33]

4.1. *Objective function and constraints*

Based on the average production time per component equation for single pass operations, the corresponding equation for a multipass turning can be obtained by summing the machining time in each pass, the non-productive time and the tool return time per pass, i.e.

$$T_T = T'_L + \sum_{i=1}^{m} \left(T_{ci} + T_R \frac{T_{ci}}{T_i} \right) + mT_s \tag{26}$$

where T_s is the tool return time per pass, T'_L is the loading/unloading and set-up time for the operation (excluding T_s) and the other symbols are as defined in the nomenclature.

The average time per component equation can similarly given by:

$$T_T = x \left[T'_L + \sum_{i=1}^{m} \left(T_{ci} + T'_R \frac{T_{ci}}{T_i} \right) + mT_s \right] \tag{27}$$

where T'_R is given in Eq. 3. Equations 26 and 27 are mathematically similar. For the same reasons as for the single pass optimization, only the analysis for the time

per component equation will be given here. The cutting time per pass T_{ci} and the Taylor-type tool-life equation for each pass T_i are respectively given by

$$T_{ci} = \frac{\ell \pi D}{V_i f_i} \tag{28}$$

$$T_i = \frac{K}{V_i^{1/n} f_i^{1/n_1} d_i^{1/n_2}} \tag{29}$$

It should be noted that in a multipass turning operation, the workpiece diameter is not constant but reduced after each pass. However, it is expected that in modern near net shape manufacturing, the total depth of cut to be removed has been minimized and the diameter variation between passes is minimal and hence can be ignored. The analysis allowing for workpiece diameter variation is given by Armarego *et al.*[5]

Substituting Eqs. 28 and 29 into Eq. 26 gives the fundamental form of the T_T equation as

$$T_T = T'_L + \sum_{i=1}^{m} \frac{\ell \pi D}{V_i f_i} \left(1 + \frac{T_R}{K} V_i^{1/n} f_i^{1/n_1} d_i^{1/n_2}\right) + m T_s \tag{30}$$

where

$$d_T = \sum_{i=1}^{m} d_i \tag{31}$$

It is expected that the non-productive loading and unloading time T'_L, the tool replacement time T_R and the tool return time per pass T_s have been minimized. Thus, the time per component T_T is to be minimized by selecting the optimum number of passes m and the cutting conditions per pass V_i, f_i and d_i without violating a variety of practical constraints, such as the maximum allowable machine tool power force $F_{\mathrm{p\,max}}$, power $P_{\max}$, low speed power or torque limits, and the feed and speed boundary constraints. For finishing operations, the component surface roughness constraint needs to be considered in the last pass of the operation. With reference to the single pass analysis, the constraints for each pass are given as follows:

Machine tool speed and feed boundary constraints

$$\pi D N_{\min} = V_{\min} \leq V_i \leq V_{\max} = \pi D N_{\max} \tag{32}$$

$$f_{\min} \leq f_i \leq f_{\max} \tag{33}$$

Machine tool maximum power constraint

$$P_i = W V_i f_i^{\alpha} d_i^{\beta} \leq P_{\max} \tag{34}$$

Machine tool power force constraint

$$f_i \leq f_{Fi} = \left(\frac{F_{p\max}}{E d_i^{\beta}} \right)^{1/\alpha} \tag{35}$$

Machine tool low speed power or torque constraint

$$f_i \leq f_{ai} = \left(\frac{A}{W d_i^{\beta}} \right)^{1/\alpha} \tag{36}$$

Component surface roughness constraint

$$f_i \leq f_{Rti} = R_{t\max}(\tan \psi_r + \cot \kappa_r') \tag{37}$$

Since Eqs. 35 and 36 only limit the permissible feed and are mutually exclusive, they can be generalized by a feed limit f_{xi}, i.e.

$$f_i \leq f_{xi} = \min\{f_{Fi}, f_{ai}\} \tag{38}$$

If there is a surface finish requirement, Eq. 37 needs to be considered only in the final pass of the multipass operation and Eq. 38 becomes

$$f_i \leq f_{xi} = \min\{f_{Fi}, f_{ai}, f_{Rti}\} \tag{39}$$

A further study of the machining performance data from the machining data handbook[1] has found the following relationships which will be used in the current analysis: $1/n > 1$, $1/n > 1/n_1 > 1/n_2 > 0$ and $1 \geq \beta > \alpha > n/n_1 > 0$.

4.2. *Optimization analyses*

The optimization of cutting conditions in multipass machining operations to allow for integer number of passes m and discrete limits has been found to be very complicated and the use of numerical search methods is considered to be inevitable. Unfortunately, purely numerical search techniques suffer from some disadvantages. Firstly, the number of iterations required increases rapidly with the number of variables, which can result in excessive computer processing time; secondly, such methods cannot guarantee the required global optimum with the chance of the solution converging to a local optimum; thirdly, establishing a suitable initial point to start the numerical search can present difficulties; and finally, there is no evident termination point to the number of passes to be considered in the numerical search procedure.

Therefore, a purely numerical search is undesirable for use in the optimization of cutting conditions in multipass machining operations in which the number of variables to be optimized is three times of the number of passes, i.e. the depth of cut per pass d_i as well as the speed V_i and feed f_i in each pass. However, the number of variables involved can be reduced greatly if the explicit single pass economic trends and solution strategy are applied to each pass of a multipass operation so that

numerical search techniques are only required to determine the number of passes and the depth of cut for each pass.

Thus, the optimization approach becomes one whereby a numerical search method in conjunction with the single pass strategy is used to find the optimum depth of cut d_i distribution for a given integer number of passes m, while the optimum feed f_i and speed V_i in each pass are obtained by using the single pass optimization strategy. The procedure is then to search for the integer m which yields the minimum T_T. Consequently, the multipass optimization problem can be formulated to optimize the T_T function of integer m and d_i for each pass, i.e.

$$\begin{aligned} &\text{optimize} && T_T(d_i, d_2, \ldots, d_m) \\ &\text{subject to} && d_T = \sum_{i=1}^{m} d_i \end{aligned} \tag{40}$$

In addition, the depth of cut d_i should be greater than zero and within the maximum permissible limit for that pass of operation to be feasible under the constraints considered. In other words, too big a depth of cut per pass may result in that pass to be non-feasible due to the constraints. This limit can be found by jointly considering the constraints in Eqs. 32–37.[8] Other constraints are not stated here since they are not required explicitly in the analysis.

Further difficulties can arise in the numerical search procedure to optimize the number of passes m, since the range of m values to be considered should ensure that a global, not a local, minimum has been found without unnecessarily increasing the number of iterations and computer processing time. For this purpose, the economic characteristics can be established and applied to advantage to overcome this difficulty.

In this optimization analysis, the global optimum number of passes m, as well as the optimum cutting conditions for each pass (V_i, f_i, d_i), is initially found for a "continuous" m followed by a limited numerical search strategy to locate the "integer" m optimum solution. The continuous m optimization will provide a minimum "envelope" to the integer m solution and the number of passes about which the global optimum integer m can be found, thus reducing the computer processing time while guaranteeing a global optimum solution. In addition, the consideration of surface roughness constraint in only the last pass will complicate the analysis. Hence, this constraint will not be considered in the continuous m optimization, but in the numerical search stage for the integer m optimum solution. Since the additional constraint in the final pass should result in an increased T_T, the continuous m optimization so determined will still provide the minimum envelope to the integer m solution.

4.2.1. *Optimization with continuous number of passes*

The "continuous" m optimization analysis allowing for the constraint Eqs. 32–36 involves four major steps to arrive at the global optimum solution. These four steps and the major economic trends are outlined below.

(i) *Identification of various constrained optimum (V_i, f_i) solutions*
It has been shown in the single pass analysis that under different constraint and other conditions, there are 11 possible solutions including one that a single pass operation is not feasible. It is realized that a multipass operation should always be feasible since the depth of cut per pass can always be reduced to satisfy the technological constraints. It should also be realized that in a multipass operation, as the depth of cut per pass varies, the $\partial T_{Ti}/\partial V_i = 0$ and $\partial T_{Ti}/\partial f_i = 0$ loci and the various constraint curves in the V_i–f_i domain "float" with respect to each other depending on the values of the empirical equation constants and the constraint values. This floating will result in different combinations of the relative positions of the curves in the f_i–V_i domain and the resulting optimum solution for the feed and speed per pass. Thus, the various constrained optimum V_i and f_i, and the corresponding depth of cut d_i limits need to be identified and established.

It is apparent that the objective function for T_T in Eq. 30 includes the single pass case when $m = 1$ and $d_i = d_T$ where only one pair of speed V_i and feed f_i has to be optimized for a minimum T_T using the single pass strategy. In a multipass operation, a similar analysis to that used in the single pass optimization can be used to determine the speed per pass V_i and feed per pass f_i for a given d_i. A detailed analysis has found that for all possible input conditions and by varying the depth of cut per pass d_i, the 10 optimum feed and speed solutions identified in the single pass optimization still apply when the loci of $\partial T_{Ti}/\partial V_i = 0$ and $\partial T_{Ti}/\partial f_i = 0$ and constraint equations float in the V_i–f_i domain. It is noted that the optimum solutions (g) and (i) in Fig. 4 are represented by a constant cutting speed $V_{\max}$ or V_x at its intersection with the generalized feed limit f_x, and that the one with a smaller value is effective. To simplify the analysis, these two cases can be combined and the smaller cutting speed between V_x and $V_{\max}$ is used in the analysis. Consequently, nine distinctly different and feasible constrained optimum speed V_i and feed f_i per pass corresponding to nine depth of cut d_i regions have been identified.[8] These nine possible optimum solutions for each pass are regarded as sub-optimized solutions and will be further analyzed in order to arrive at the global optimum including all the passes.

(ii) *Characteristics of the nine sub-optimized T_T equations*
By substituting the nine possible pairs of the optimum (V_i, f_i) solutions into Eq. 30, the nine sub-optimized T_T equations can be established. It has been found that these equations have a common form, i.e.

$$T_{Tr} = T'_L + k_{1r} \sum_{i=1}^{m} d_i^{b_{1r}} + k_{2r} \sum_{i=1}^{m} d_i^{b_{2r}} + mT_s \quad r = 1 \text{ to } 9 \tag{41}$$

where k_{1r} and k_{2r} are constants depending on the workpiece diameter and constraint data, while the exponents b_{1r} and b_{2r} depend on the exponents in the tool-life, force and power equations, n, n_1, n_2, α and β.

It is not clear if the optimum solutions for all the pass are subject to the same constraints, i.e. only one of the nine sub-optimized solutions applies, or a mixed constraint global optimum solution. Thus the characteristics of Eq. 41 is analyzed in order to arrive at the optimum number of passes m_o and the depth of cut per pass d_i distribution.

Using Lagrangian multipliers, it has been found[33] that for the relevant range of the constants and exponents in Eq. 41, a turning point occurs at an equal d_i distribution for each of the nine sub-optimized equations under continuous number of passes m. Depending on the values of the exponents b_1 and b_2, the turning point at equal d_i per pass results in a minimum or maximum sub-optimized T_T. Thus if the turning point is a minimum, d_i is related to m by $d_i = d_T/m$ and Eq. 41 becomes a function of m, i.e.

$$T_{Tr} = T'_L + k_{1r} d_T^{b_{1r}} m^{1-b_{1r}} + k_{2r} d_T^{b_{2r}} m^{1-b_{2r}} + T_s m \quad r = 1 \text{ to } 9 \tag{42}$$

The local or sub-optimum m_{or} at the turning point where a minimum T_T occurs can be found by vanishing the first-order derivative of Eq. 42, i.e.

$$\frac{dT_{Tr}}{dm} = (1-b_{1r})k_{1r}\left[\frac{d_T}{m_o}\right]^{b_{1r}} + (1-b_{2r})k_{2r}\left[\frac{d_T}{m_o}\right]^{b_{2r}} + T_s = 0 \quad r = 1 \text{ to } 9 \tag{43}$$

By contrast, if the turning point is a maximum, the d_i for all passes have to be selected at the highest permissible depth of cut value corresponding to the sub-optimum solution under consideration, d_{maxr}. This results in yet another equal depth of cut distribution but with the optimum continuous m equal to d_T/d_{maxr}.[8,33]

Consequently, for the given constraints, each sub-optimized T_{Tr} function for the respective optimal equal d_i distribution can exhibit the T_{Tr}–m characteristics whereby the T_{Tr} either monotonically increases with m or has a turning point minimum at $m = m_o$. The T_{Tr}–m characteristics never exhibit a monotonically decreasing T_{Tr} trend with m. Furthermore, by submitting the equal d_i per pass condition in to Eq. 41, it can be shown that each of the sub-optimized T_{Tr}'s is linearly related to total depth of cut d_T, *viz.*

$$T_{Tor} = T'_L + C_r d_T \quad r = 1 \text{ to } 9 \tag{44}$$

(iii) *Possibility of mixed constraints for continuous m*

From the foregoing analysis, the depth of cut for each pass in a multipass operation may fall into one of the nine regions and result in one of the nine sub-optimum (V_i, f_i) solutions. It is not obvious whether the global optimal solution occurs when all passes are subject to the same active constraints (with the same sub-optimized T_{Tr} for all passes), or a "mixed constraint" global optimum solution. In order to study this, it is assumed that the total depth of cut d_T is divided into nine sets or sub-total depths of cut d_{Tr}, one for each of the nine sub-optimum solutions identified above. It is also assumed that within each sub-total depth of cut d_{Tr} more than one passes can occur, so that the sub-optimum for each d_{Tr} can be found by using equal cutting conditions per pass and the sub-optimum T_{Tor} is represented by Eq. 44. The

total time per component T_T for the total depth of cut can be found by summing the times for the nine sub-total depths of cut, i.e.

$$T_T = T'_L + C_1 d_{T1} + C_2 d_{T2} + \ldots + C_9 d_{T9} \tag{45}$$

where

$$d_T = d_{T1} + d_{T2} + \ldots + d_{T9} \tag{46}$$

It has been proven by Wang[8] that since T_T in Eq. 45 is a linear function of the nine sub-total depths of cut d_{Tr}'s, the global optimum T_{To} occurs when one of the nine d_{Tr}'s is equal to the total depth of cut d_T and the rest are zero, i.e. at a vertex (or a corner point) of the T_T–d_{Tr} hyperplane (i.e. a plane in the nine-dimensional space). Hence for a continuous number of passes m, the global optimal solution occurs when all passes are subject to the same constraints with the same cutting conditions in each pass and the possibility of "mixed constraint" conditions does not apply. It should be noted that this conclusion is derived when the surface finish constraint is not considered. The corresponding optimum number of passes m and the optimum cutting conditions can be found according to one of the nine T_{Tor} solutions above. However, it is not known on which one of the nine sub-optimized solutions the overall global optimum occurs and a further study is required.

(iv) Overall T_T–m curve and global optimum solution

Under different combinations of the constraint values and the constants in the tool-life, force and power equations, not all the nine sub-optimized (V_i, f_i) solutions are relevant as the depth of cut per pass d_i varies. In order to establish on which sub-optimum solution, the overall global optimal occurs with an equal depth of cut d_i distribution (i.e. $d_i = d_T/m$), it is first necessary to identify the relevant sub-optimized solutions under different conditions and establish their sequences as the equal d_i decreases or m increases.

For this purpose, the various combinations of the nine constrained optimum (V_i, f_i) loci in the V_i–f_i domain established above have been considered from which 13 different sequences of the sub-optimized solutions have been identified for all possible input conditions as d_i decreases (or m increases). These different sequences involve from as many as five to as few as two of the nine sub-optimized solutions. The nine sub-optimized solutions need never to be considered simultaneously. The limiting conditions for identifying each of the 13 sequences and the associated sub-optimum solutions have also been established for use in the software package. In addition, the nine sub-optimized T_{Tr}–m functions corresponding to the nine sub-optimized (V_i, f_i) solutions have been established for use in the analysis.

To establish the overall T_T–m curves for each of the 13 sequences of sub-optimized T_{Tr} functions from which the optimum continuous number of passes m can be identified, the "joining relationship" of all the relevant consecutive T_{Tr}–m functions at their d_i (and m) junctions need to be studied. These d_i (and m) junctions are the limits for differentiating the consecutive sub-optimized solutions

or T_{Tr}–m functions within each sequence. While many of the junctions are smooth with a common tangent for the pair of T_{Tr}–m functions (or curves), others cross each other with known relative slopes enabling the overall T_T–m curves to be established for each sequence of the T_{Tr}–m functions.

From the sub-optimized T_{Tr}–m functions, their sequences and the "joining relationships" between the consecutive T_{Tr}–m functions established above, it has been found that three types of overall "active" T_T–m curves are possible as shown in Fig. 5. The overall T_T–m curve may be monotonically increasing in the feasible m region so that the optimum m_o is the least feasible $m_{\min}$ as in Fig. 5(a), or the curve may exhibit a minimum turning point at m_o on one of the T_{Tr}–m functions that have a turning point as in Fig. 5(b), and finally the active overall curve may exhibit a minimum at a crossing junction of two sub-optimized T_{Tr}–m functions in a sequence which gives the optimum m_o as in Fig. 5(c). It should be noted that the least feasible number of passes $m_{\min}$ is determined such that each pass in the multipass operation with equal cutting conditions per pass is feasible to perform and the case shown in Fig. 4(k) does not occur.

Having established the optimum number of passes m_o for the continuous m optimization, the optimum equal depth of cut per pass can be found from $d_i = d_T/m_o$ while the constrained optimum V_i and f_i are found from the trends and analyses for the single pass optimization. As a consequence, twelve distinctly different optimum solutions for continuous m have been identified and the associated cutting and limiting conditions have been found.[3,8]

4.2.2. *Optimization strategies for integer number of passes*

It has been proven[33] that the overall T_T–m curve for continuous number of passes established above provides the minimum T_T–m envelope to the integer m solutions, i.e. the global optimum solution for integer m will be either on or above the continuous m curve. Based on the trends of the continuous m overall T_T–m curve as shown in Fig. 5, an immediate and simplified strategy for arriving at the integer m optimum solution can be proposed. In this simplified strategy, the search can start with the least feasible integer $m \geq \max\{1, m_{\min}\}$, and a numerical search algorithm

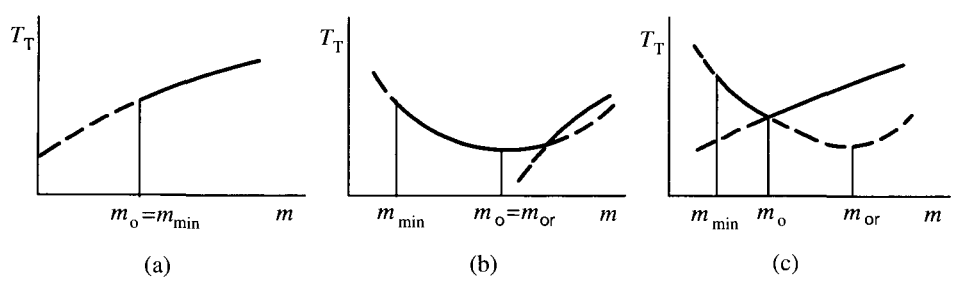


Fig. 5. The overall T_T–m functions and optimum solutions for continuous constraints.

can be employed to locate the optimum depth of cut d_i distribution while the optimum feed and speed for each depth of cut tested can be determined by using the single pass optimization strategy. The surface finish requirement, if relevant, will need to be considered in only one pass that is used as the last pass of the operation. The search continues with an increment of one pass for each iteration. When the optimum T_{To} for the number of passes under consideration is greater than previously searched T_{To}'s, the search is terminated, as T_T does not decrease with m and any further search will only arrive at an increased T_{To} value.

It can be seen that although the single pass and continuous m multipass analyses have been used to benefit the above procedure in determining the optimum cutting conditions for a multipass operation, other economic trends derived from the continuous m multipass analysis can be used to determine the starting point and locate the number of passes region within which the final integer optimum solution can be found, thus reducing the computer processing time.

Figure 6 illustrates a more comprehensive optimization strategy for integer m multipass operations on CNC machines. The analysis for rough machining is considered first where the same constraints imposed by the machine tool apply for all passes. When the continuous m overall T_T–m function is monotonically increasing, the optimum continuous m is equal to $m_{\min}$ below which at least one constraint is violated. If $m_{\min} \geq 1$ and is an integer then the final optimal solution is at $m_{\min}$ for a rough turning operation where all passes are subject to the same constraints. However if $m_{\min}$ is not an integer, the next higher integer $m_H = \text{int}\{m_o + 1\}$ is considered. If m_H is in the region where equal d_i per pass gives a minimum turning point on the T_T–m curve for continuous m, then the optimum T_T lies on the overall T_T–m curve for continuous m and m_H is the final solution with the optimum V_i and f_i found by using the single pass strategy. On the other hand, if m_H lies in the region where equal d_i yields a maximum turning point in the sub-optimized T_{Tr} function, a "penalty function" method[28] is used to find the optimal d_i distribution at m_H with the associated (V_i, f_i) found from the single pass strategy. The corresponding T_T, which lies above the T_T–m curve as shown in Fig. 6(a), is then used to find the upper number of passes limit m_{UP} and the final solution is found by comparing the T_T values for all integer m between m_{H} and m_{UP}. The numerical

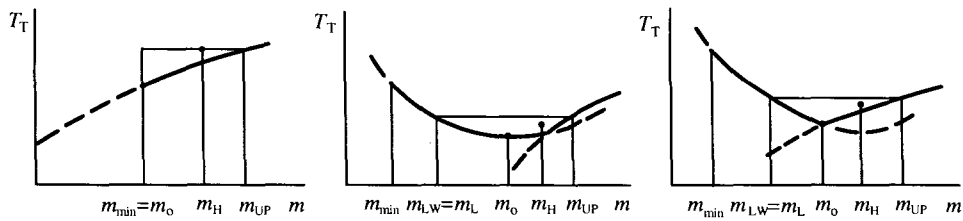


Fig. 6. The region for optimum number of passes and integer m strategy.

search can be accelerated by updating the m_{UP} limit for each improved (smaller) T_T found.

When the overall T_T–m curve for continuous m exhibits a minimum point at m_o as in Figs. 6(b) and (c), the optimum T_T at the next lower integer $m_L = \mathrm{int}\{m_o\}$, if feasible, will lie on the overall T_T curve. Any integer number of passes $m < m_{\mathrm{L}}$ need not to be considered and m_{L} is used as the lower number of passes limit m_{LW}. The optimum T_T at m_{L}, which is obtained by applying equal d_i for all passes and the single pass strategy for the optimum V_i and f_i, is used to establish the up number of passes limit, m_{UP}, which may be needed to obtain the final solution. If the next higher integer $m_{\mathrm{H}} = [m_{\mathrm{L}} + 1]$ lies in the region where an equal d_i distribution gives a minimum turning point, the T_T at m_{H} will lie on the overall T_T–m curve and the final solution is found by comparing the T_T values at m_{L} and m_{H}. However, if the m_{H} lies in the region where an equal d_i distribution gives a maximum turning point to the corresponding T_{Tr} function, then numerical search is used to find and compare the T_T values for all integer number of passes between m_{LW} and m_{UP}, whereby the penalty function method is used to search for the optimum d_i distribution for each m with the optimum V_i and f_i for each d_i found from the single pass strategy. The least T_T found is the required final global optimum solution. It should be noted that if the next lower number of passes is not feasible, i.e. $m_{\mathrm{L}} < \max\{1, m_{\min}\}$, the strategy discussed above for the situation where T_T monotonically increases with m as shown in Fig. 6(a) is used to find the optimal solutions.

The strategy for the case where the surface roughness constraint is considered in the final pass of a multipass operation, though similar to that outlined above, is more complex due to the additional constraint in one pass and the fact that the optimum T_T for any m is unlikely to lie on the T_T–m curve for continuous m established above. A rigorous strategy may be to use the penalty function method to optimize the d_i distribution for each m, with the optimum V_i and f_i in each pass determined by the single pass strategy. To simplify the analysis, an approximate and practical strategy is proposed, whereby the optimum number of passes m_o, the depth of cut per pass d_i distribution and the feed and speed per pass are first determined for a rough machining operation. The smallest d_i (or any d_i for equal cutting conditions) from the rough machining optimization is then used to determine the optimum (V_i, f_i) solution for the final pass allowing for the surface finish requirement with the single pass strategy. Based on the above analysis, a computer program has been written to implement the strategy.

5. Application of Machining Optimization Strategies to the Selection and Design of Machine Tools

While the single pass and multipass optimization strategies have been developed to select the economic cutting conditions for the minimum time or cost per component for individual machining operations, these strategies may be used on a case by case basis to select the most appropriate available machine tool in a production facility in

process planning or to deal with the machine loading problem in shop floor control. A broader approach may also be adopted, whereby the optimization strategies are used to evaluate and select appropriate machine tools for the expected population of components to be machined.

Taking the latter approach, a numerical simulation study is given here for end-milling operations using the single pass optimization strategy.[36] The reason for using end-milling in the simulation is the availability of the data. In this study, three modern CNC machining centers have been evaluated for end-milling of a plain carbon steel ($\leq$ 0.6% C) of ultimate tensile strength of 657 MPa with high speed steel (HSS) cutters (30° helix angle and 15° normal rake angle). The machine tool specifications are given in Table 1, where differences in power, torque, feed force as well as feed speed and spindle speed ranges are evident. In particular, the torque constraint in the SABRE-750 is considerably larger than those for the standard ANCA-MMC800 and the modified ANCA-MMC800-1 machines, while the feed force constraint for the SABRE-750 is considerably smaller than that for the two ANCA machines.

Based on the 108 end-milling cases described in Table 2 and the maximum production rate criterion, the results of applying the single pass optimization strategy for each machine tool are summarized in Table 3. It is interesting to note that the average optimum time per component T_{To} for the ANCA-MMC800 machine is 5.68 min, which is very similar to that for the SABRE-750 of 5.73 min, while a higher average T_{To} of 8.52 min applies for the modified ANCA-MMC800-1 machine. A further study of these results has shown that although the standard ANCA-MMC800 and SABRE-750 machines gave comparable average T_{To}, the "active" constraints in the optimization were different with the former machine being limited by its lower torque of 48 NM and the latter machine being constrained by the lower feed force

Table 1. Specifications for the three CNC machining centers.

	SABRE-750	ANCA-MMC800	ANCA-MMC800-1
Feed speed (mm/min)	3–1,200	0–5,000	0–5,000
Spindle speed (rpm)	60–8,000	0–5,000	0–10,000
Max. power (KW)	11.2	7.5	7.5
Max. torque (NM)	143	48	24
Feed force (N)	3,100	8,000	8,000

Table 2. Cutting conditions for simulation study.

Radial depth of cut a_r (mm)	3, 5, 8	12	16, 20
Cutter diameter D (mm)	16	25	40
Number of teeth z	3	3	4
Axial depth of cut a_a (mm)	20	30	40
Length of cut ℓ (mm)	150, 300, 450, 600, 750, 900		
Non-productive time $T_L = 1$ min. Cutter change time $T_R = 0.2$ min.			

Table 3. Summary of optimum T_T (in minutes) for CNC machining centre simulation.

	SABRE-750	ANCA-MMC800	ANCA-MMC800-1
Average T_{To}	5.73	5.68	8.52
Maximum T_{To}	20.18	23.84	37.69
Minimum T_{To}	1.26	1.17	1.27
T_{To} range	18.92	22.67	36.42

of 3100 N. The differences in performance of these two machines are noted in the range of T_{To} values in Table 3. The even lower torque of 24 NM for the "high-speed" modified ANCA-MMC800-1 machine was found to be the "active" constraint again. This explains why the average T_{To} increased to 8.52 min for this machine. It is also interesting to note that none of the machines utilized the available maximum power for the range of conditions tested.

In general, it appears that the SABRE-750 could be improved by increasing the feed force limit, while the two ANCA machines require considerably more torque. The latter is particularly so if face milling with generally larger cutter diameters (resulting in larger machining torque) than those of end-mills is conducted on the two machines, whereby the torque limits will severely restrict the selection of cutting conditions while the force constraint may never come into play. Thus, the use of the optimization strategy for assessing and improving the design specification and capabilities of machine tools has been demonstrated in this simulation study.

6. Conclusions

Owing to the use of Hi-Tech, high cost equipment and system and the increased proportion of available production time spent on machining in computer aided manufacturing systems, the efficient and economic use of machining operations is becoming increasingly important. A number of earlier studies.[6,33,36,37] have shown the substantial benefits in production time and cost per components incurred when using optimization strategies rather than handbook recommended cutting conditions. These studies have also highlighted the increased benefits of using optimization strategies in process planning in modern computer aided manufacture, where the proportions of non-productive times are low and continually being improved.

While a number of approaches have been used or developed to optimize the cutting conditions in the various machining operations, the deterministic optimization approach, though time consuming in developing the relevant optimization strategies, has a number of distinct advantages over the others. Using turning as an example, the procedures of developing realistic and clearly defined optimization strategies for single pass machining operations with the deterministic approach have been demonstrated based on the criteria typified by the minimum production time per component and allowing for the many practical constraints. Despite the complexity, the detailed optimization analysis assisted by the feed-speed diagrams has provided an

in depth understanding of the economic characteristics and the influence of the constraints and machining performance data, and a means of guaranteeing the global optimum solution. The resulting optimization strategies and software are suitable for on-line CAM applications.

It is now evident[5,7,33,35] that single pass machining operations are not always superior to multipass and under certain constraint conditions, a single pass operation may not even be feasible, recourse to the development and use of multipass optimization strategies. The procedures in developing the multipass machining optimization strategies have been demonstrated and outlined for turning operations. The multipass optimization analysis with continuous number of passes has been shown to be essential to establish the lower-bound sub-optimized T_T–m function, the starting point for numerical search for the optimum integer number of passes, and the region within which the global optimum solution for integer number of passes can be found. The combination of optimization analyses and limited use of numerical search techniques has again provided clearly defined strategies which guarantee the global optimum solution.

The numerical simulation study has demonstrated the importance and use of the optimization strategies in assessing and selecting machine tools in production as well as in improving the machine tool capabilities and specifications at the design stage for economic and efficient production.

Acknowledgments

The author wishes to thank Dr. E.J.A. Armarego for his contributions to the development of the deterministic machining optimization approach covered in this chapter, and Dr. R. Mahalinga-Iyer for proof-reading and his comments on this chapter.

Nomenclature

A	constant
a_a, a_r	axial and radial depths of cut in milling, respectively
C_T	average cost (excluding material) per component
d	depth of cut
D	workpiece or cutter diameter
d_T	total depth of cut
E, W, α, β	constants in cutting force and power equations
f	feed per revolution
f_a	f limit due to low speed power constraint
f_F	f limit due to power force constraint
$f_{\min}$, $f_{\max}$	min. and max. f available from machine tool
F_p, $F_{p\max}$	power force, machine tool maximum power force limit
f_{Rt}	feed limit due to surface roughness constraint
f_x	f limit due to combined F_{pmax}, P_a and R_{tmax} constraints

K, n, n_1, n_2 tool-life equation constants
m number of passes
N spindle speed
$N_{\min}, N_{\max}$ machine tool min. and max. spindle speed limits
P machining power
P_a machine tool low speed power
$P_{\max}$ machine tool maximum power constraint
R_{tmax} max. surface roughness (peak-to-valley) constraint
T tool-life in time units
T_{ac} actual cutting time
T_c cutting (feed engagement) time
T_L loading/unloading and idle time per component
T'_L T_L excluding T_s
T_R tool or cutter replacement time per failure
T_s cutter return time per pass
T_T average total production time per component
V cutting speed ($= \pi DN$)
V_a cutting speed above which $P_{\max}$ is effective
$V_{\max}, V_{\min}$ machine tool max. and min. speed limits for a given D
$V_v(f)$ V on $\partial T_T/\partial V = 0$ for a given f
V_x V at $P_{\max}$ and f_x intersection
x labour and overhead cost rate
y tool cost per failure
z number of teeth on a milling cutter
μ constant of proportionality ($= 1/\pi$)
ℓ length of cut

Subscripts:

i ith pass in a multipass operation
o optimum value

References

1. X. Ai and S. G. Xiao, *Metal Cutting Conditions Handbook*, 2^{nd} Ed. (Mechanical Industry Press, China, 1985).
2. E. J. A. Armarego and R. H. Brown, *The Machining of Metals* (Prentice-Hall Inc., New Jersey, 1969).
3. E. J. A. Armarego, Economics of machining-criteria, constraints and selection of optimum cutting conditions, UNESCO-CIRP Seminar on Prod. Tech., Singapore, 1983.
4. E. J. A. Armarego, P. K. Kee and A. J. R. Smith, Computer aided optimization of machining conditions in single pass turning, *Proc. 3rd Int. Conf. Manuf. Eng.* (I.E.Aust.) Newcastle, Australia, 1986.
5. E. J. A. Armarego, T. H. Chia and P. K. Kee, A development of computer-aided optimization strategies for multipass turning operations, *Proc. 4th Int. Conf. Manuf. Eng.* (I.E.Aust.) Brisbane, Australia, 1988, 1–5.

6. E. J. A. Armarego, A. J. R. Smith and J. Wang, Constrained optimization strategies and cam software for single pass peripheral milling, *Int. J. Prod. Res.* **31** (1993) 2139–2160.
7. E. J. A. Armarego, A. J. R. Smith and J. Wang, Computer-aided constrained optimization analyses and strategies for multipass helical tooth milling operations, *Annals of CIRP* **43**, 1 (1994) 437–442.
8. T. H. Chia, Optimization of cutting conditions for turning operations, PhD Thesis, University of Melbourne, 1984.
9. Z. J. Da, J. P. Sadler and I. S. Jawahir, Multiple criteria optimization of finish turning operations based on a hybrid model, *Proc. 1996 ASME Design Eng. Technical Conf.*, Irvine, C.A., 1996, 1–10.
10. E. P. DeGarmo, J. T. Black and R. A. Kohser, *Materials and Processes in Manufacturing*, 7^{th} Ed. (Macmillan, New York, 1990).
11. G. Draghici and C. Paltinea, Calculation by convex mathematical programming of the optimum cutting condition when cylindrical milling, *Int. J. Mach. Tool Des. Res.* **14**, 143 (1974).
12. A. M. Eskicioglu and H. Eskicioglu, Application of three non-linear programming techniques in optimizing machining conditions, *Proc. Instn. Mech. Engrs.*, Part B **206** (1992) 183–188.
13. W. Eversheim, W. König, M. Weck and T. Peeifer, Tagunsband des AWK'84, Aachener Werkzeugmaschinen-Kolloquim, 1984.
14. M. Y. Friedman, and V. A. Tipnis, Cutting rate-tool-life functions for material removal processes: Part I—Theory, *J. Eng. Ind.* **98** (1976) 481.
15. C. L. Hough and R. E. Goforth, Optimization of the second order logarithmic machining economics problem by extended geometric programming Part II Posynomial constraints, *AIIE Trans.* **13** (1981) 234.
16. C. Huynh and R. Dluzniak, Neural networks optimization for machining of cast and forged components, *Proc. 6^{th} Int. Conf. Manuf. Eng.* (I.E.Aust.), Melbourne, 1995, 353–357.
17. P. K. Kee, Optimization strategies and CAM software for rough turning operations, PhD Thesis, The University of Melbourne, Australia, 1991.
18. P. K. Kee, Development of computer-aided machining optimization for multi-pass rough turning operations, *Int. J. Production Economics* **37** (1994) 215–227.
19. E. I. Kruglov and O. I. Darymov, Optimization of face milling, *Machines and Tooling* **49**, 3 (1978) 35–37.
20. Y. H. Lee, H. M. Shun and B. H. Yang, An approach for multiple criteria simulation optimization with application to turning operation, *Computers and Engineering* **30** (1996) 375–386.
21. B. Malakooti, An interactive on-line multi-objective optimization approach with application to metal cutting turning operation, *Int. J. Prod. Res.* **29** (1991) 575–598.
22. M. E. Merchant, Industry-research integration in computer aided manufacturing, *Int. Conf. Prod. Tech.*, Melbourne, Australia, 1974.
23. V. A. Ostafiev, A. V. Globa and L. Globa, Integrated end milling optimization development, *Annals of CIRP* **33** (1984) 29.
24. S. S. Rangwala and D. A. Dornfeld, Learning and optimization of machining operations using computing abilities and neural networks, *IEEE Transactions on Systems, Man, and Cybernetics* **19**, 2 (1989) 299–314.
25. S. S. Rao and S. K. Hati, Computerised selection of optimum machining conditions for job-shop type machining systems with alternative machine tools, *Annals of CIRP* **29** (1980) 335.

26. G. L. Ravignani, A contribution to machining economics, *Annals of CIRP* **27** (1978) 619.
27. G. L. Ravignani, V. A. Tipnis and M. Y. Friedman, Cutting rate-tool-life functions (R-T-F) general theory and applications, *Annals of CIRP* **25** (1977) 295.
28. S. B. Schuldt, G. A. Barriele, R. R. Root, E. Sandgren and K. M. Ragsdell, Application of a new penalty function method to design optimization, *J. Eng. Ind.* **99** (1977) 31.
29. F. W. Taylor, On the art of cutting metals, *Trans, Amer. Soc. Mech. Engrs.* **28** (1907) 31.
30. V. A. Tipnis and M. Y. Friedman, Cutting rate-tool-life (R-T) characteristics functions for three machining variables—Theory and applications, *Proc. Third North Amer Metalworking Res. Conf.*, SME, Dearborn, 1975, 361.
31. V. A. Tipnis and M. Y. Friedman, Cutting rate-tool-life characteristics functions for material removal processes: Part II—Verification and applications, *J. Eng. Ind.* **98** (1976) 487.
32. I. Yellowley, A fundamental examination of the economics of two-pass turning operations, *Int. J. Prod. Economics* **21** (1983) 617–626.
33. J. Wang, Constrained optimization of milling operations, Ph.D. Thesis, The University of Melbourne, Australia, 1993a.
34. J. Wang, Multi-objective optimization of machining operations based on neural networks, *Int. J. Adv. Manuf. Technol.* **8** (1993b) 235–243.
35. J. Wang, Constrained optimization strategies and cam software for multipass face milling operations, *Proc. World Congress—Manuf. Technol. Towards 2000*, 7th Int. Conf. Manuf. Eng., Cairns, Australia, 1997, 107–116.
36. J. Wang, Computer-aided economic optimization of end-milling operations, *Int. J. of Production Economics* **54**, 3 (1998a) 307–320.
37. J. Wang, Development of drilling optimization strategies and software for cam applications, *J. Mater. Proc. Technol.* **84**, 1–3 (1998b) 181–188.
38. J. Wang and E. J. A. Armarego, Optimization strategies and cam software for multiple constraint face milling operations, *Proc. 6th Int. Conf. Manuf. Eng. (I.E.Aust.)*, Melbourne, 1995, 535–540.
39. S. M. Wu and D. S. Ermer, Maximum profit as the criterion in the determination of optimum cutting conditions, *J. Eng. Ind.* **88** (1966) 435.

CHAPTER 5

COMPUTER TECHNIQUES AND APPLICATIONS OF ORTHOGRAPHIC PROJECTIONS FOR THE CONSTRUCTION OF SOLID MODELS IN COMPUTER AIDED DESIGN (CAD)

BYEONG-SEOK SHIN
Bit Computer Co. Ltd.
1327-33 Bitville
Seoch - Dong Seocho - K.U.
Seoul 137-072, Korea
Email: bsshin@computer.org

As design and manufacturing processes have been automated through CAD (Computer Aided Design), CAM (Computer Aided Manufacturing), and CAPP (Computer Aided Process Planning), there are two kinds of interest in orthographic projections for the construction of solid models in CAD. One is in devising methods that manage CAD drawings efficiently and enables their reuse in the design of other products. The other is to find an efficient internal representation that can be utilized throughout the whole process from design to manufacturing. This chapter is a rather comprehensive treatment of the issues and techniques involved and their substantive significance in the manufacturing process.

Keywords: Orthographic projections; construction of solid models; computer aided design (CAD); automatic drawing recognition system.

1. Introduction

The human perception system is familiar with acquiring information from three-dimensional (3D) shape of objects. We can transmit some information more rapidly and more efficiently when using visual expression of the objects to be represented. As the computer technology advances, visualization — generating 3D view of real objects or imaginary ones — becomes the most important part of information processing in these days.

In order to visualize some information in computer systems, we should make an internal representation that can be easily manipulated by the computer, which is called *modeling*. In particular, describing 3D geometry and topology of an object in computer system is called *Solid Modeling*.[36,37] This refers to an internal representation of an object defined in 3D space as well as all sorts of operations including insertion, deletion and modification of a part of the object. Several methods have

been devised for solid modeling such as boundary representation (B-rep), constructive solid geometry (CSG), sweep representation, and spatial partitioning.

Boundary representation describes an object in terms of its surface boundary — a finite set of vertices, edges and faces based on the fact that an object is bounded by some closed faces.[27] A closed face is represented as a series of edges and an edge is defined as a pair of vertices. When we use B-reps, the internal representation of the object is composed of geometric information such as coordinates of vertices, mathematical equation of edges, and surface normal vectors of faces.

In constructive solid geometry, an object is described as a set of simple geometric primitives as well as geometric transformation and regularized Boolean set operation for the primitives.[35] Geometric primitives are simple solids that can be used as basic building blocks of the object such as cubes, pyramids, and cylinders. Geometric transformations such as translation, scaling, and rotation are used for altering the position and shape of the primitives. Boolean set operations make it possible to construct a complicated object by combining the transformed primitives. An object is stored as a tree-like structure with Boolean set operators at the internal nodes and information of geometric primitives at the leaf nodes of the tree, which is called *CSG tree.*

Sweeping an object along a trajectory through space defines a new object. Sweep representation is categorized into translational sweep (extrusion) and rotational sweep.[47] In translational sweeping, we should define contour lines and trajectory of the cross-section of an object. In rotational sweeping, contour lines and rotation axis should be described. Sweeping is adequate for representing highly symmetrical objects.

In spatial partition representations, a solid is decomposed into a collection of adjoining, non-intersecting solids that are more primitive than the original solid.[11] Although this method has the disadvantage of being able to represent an object approximately, it requires a small amount of storage for describing an object. Cell decomposition, spatial occupancy enumeration, octree,[20] and binary space-partitioning tree[7,14,43] are regarded as spatial partitioning methods.

A variety of input device and input methods are used for describing and manipulating geometric models both in solid modeling softwares and in applications that require solid models as their input. Also, they provide some facilities for manipulating the models and show the result with display devices and hardcopy devices. We call these interactive modeling techniques. Interactive modeling has some advantages in that it can directly describe a model in 3D space, and it enables us to manipulate the shape of the model with visual feedback. However, it also has some serious problems. Firstly, it is not easy to describe and manipulate the model database. Except for some special hardwares such as three-dimensional digitizers and laser range scanners,[21] most of the graphics input and the output hardwares are two dimensional (2D) devices. It is difficult to describe 3D geometric elements such as 3D lines, 3D curves, and 3D surfaces using only the 2D devices. The next

problem is that interactive modeling requires a large amount of data to describe a model. This means that it needs more storage for maintaining input data and intermediate results. Several works have been done to devise an efficient internal representation.[48,51] However, it trades-off complexity of internal representation for processing time. Although detailed internal representation reduces model manipulation and rendering time, it takes more time to input data and more storage to maintain the model. On the contrary, brief representation does not require much effort but may take a long processing time.

We can solve the problems of interactive 3D modeling by using projection methods. Projection is one of the most popular and simplest methods to describe an object. In particular, it has been common convention in engineering drawings even before computers were invented. Most CAD systems represent objects based on the projection method. In the projection method, an object is described as 2D geometric primitives such as points, line segments, and arcs generated by applying parallel and perspective projection to the object defined in 3D world coordinates.[5,33] The perspective projection gives us depth cues, so it is used for making a bird-eye view with which we can see the overall shape of the object at a glance. The most serious problem of the perspective view is that it cannot preserve the size, shape, and parallelism of geometric elements. We can solve the problem with parallel projections. Even though parallel projection does not provide depth cue effectively, it has the advantage of describing the shape of the object exactly, since it preserves the parallelism of geometric primitives. Orthographic projection which projects an object onto the planes that are perpendicular to principle axes is widely used.

We can get six views — front, rear, left, right, top, and bottom view — by projecting an object orthographically onto planes perpendicular to principle axes. Symmetric objects can be represented using only two views. Normal objects require three or more views according to their complexity. Three-view method that represents an object using front view, top view and side view has been a convention in engineering drawings. Figure 1 shows how three views are made by orthographic projections, and Fig. 2 depicts an example of engineering drawing represented by three-view method on a CAD software.

It is very easy to input geometric information in the case of using the projection method. We can easily describe an object with 2D input devices such as mouse, tablet, and joystick since each view is composed only of 2D geometric primitives. Also its internal representation is so simple since it requires a small amount of storage.

However, the projection method has some problems. Since orthographic views are not solid representation, we need considerable experience and domain knowledge for inferring the 3D shape of the object from the views. Thus, it is very difficult for non-experts to interpret the meaning of views except for simple objects. The most serious problem is loss of geometric information. Some geometric information may be lost during the projection. In addition, designers would omit some geometric

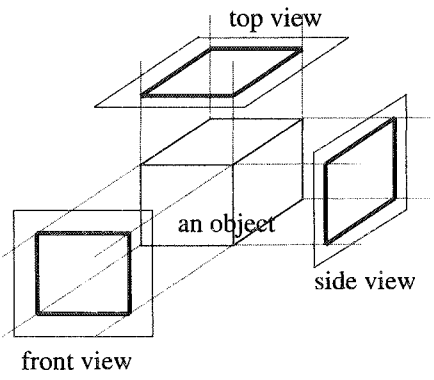


Fig. 1. Orthographic projection for an object.

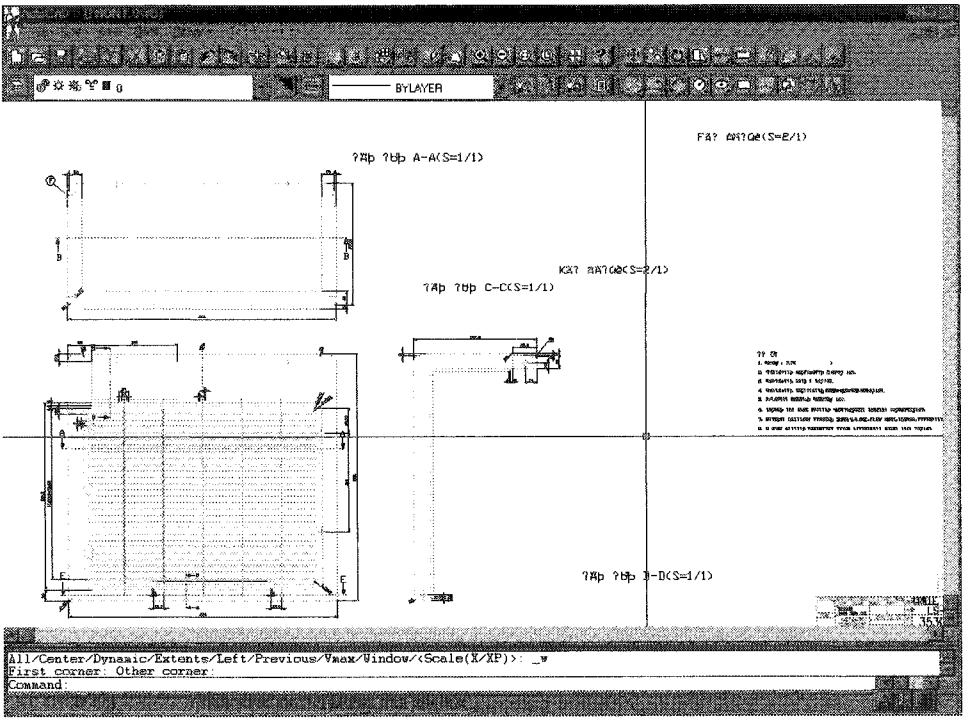


Fig. 2. An example of engineering drawing generated by using a CAD software.

primitives intentionally as convention. As a result, a set of orthographic views may be interpreted as two or more different objects. In case of overlapping a variety of shapes inside of a complex object, we may mis-interpret or fail to interpret the meaning of projections. Figure 3 shows an example of orthographic projections that can be interpreted as two different objects.

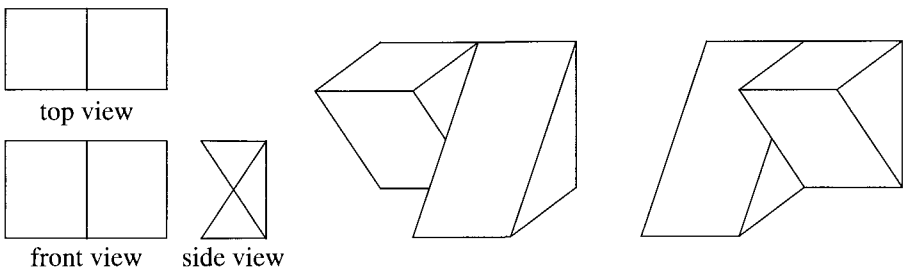


Fig. 3. An example of orthographic projections that can be interpreted as several objects.

If we take the advantages of interactive 3D modeling and projection methods, we can get the 3D representation of an object with less cost and effort. One solution is to devise an algorithm that reconstructs solid model from a set of projections. This is an inverse process of projection of a 3D object. Three-dimensional geometric elements such as vertices and edges of an object are represented as specific patterns of 2D geometric elements on each view. Therefore we can find candidate elements that may be original 3D vertices and edges by doing combinatorial search for 2D geometric elements. Some of the candidate elements can be 3D geometric primitives while others may not be. In order to distinguish the 3D primitives from invalid ones, we should reproject 3D candidate elements and check whether the images of them are represented as the same 2D geometric primitives on each view. Restoring 3D structures from 2D geometric elements is called *three-dimensional model reconstruction.*

Three dimensional model reconstruction systems require orthographic projection files composed of 2D geometric elements as their input. Projection files can be drawings from CAD softwares, free-hand drawings or hardcopies of CAD drawings. If we digitize the drawings by a scanner, and then apply some types of image processing techniques and vectorization, we can get the same results as the modeling data generated by CAD software.[22,30] Figure 4 shows the generic block diagram of solid model reconstruction system.

Reconstructed 3D models are stored in computer storage in a solid model representation form such as boundary representation or CSG, and they can be applied to a variety of areas. For example, designers can verify the robustness of drawings by visualizing reconstructed 3D models with several rendering techniques. Also 3D models can assist normal users who cannot interpret engineering drawings to recognize the shape of objects that are represented.

2. Application Areas

Three-dimensional model reconstruction methods can be utilized in a variety of applications. We can apply it to create and manipulate virtual environments in virtual reality systems. If we design a virtual environment using orthographic

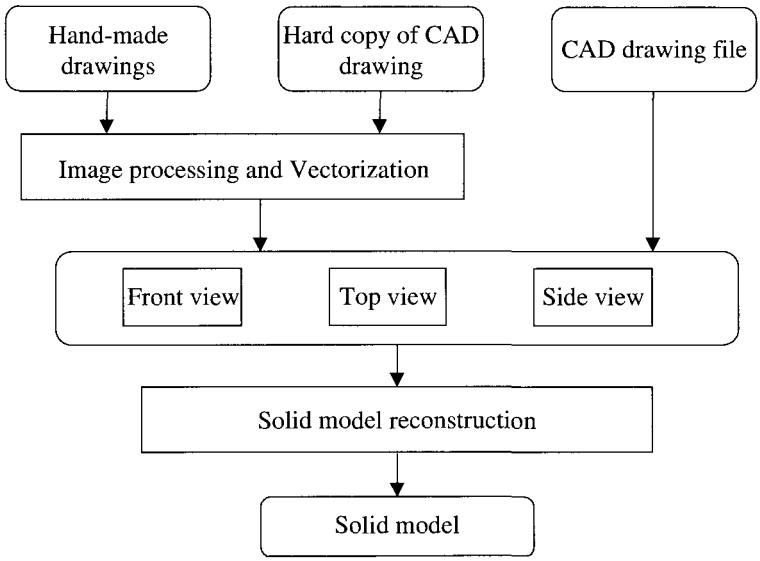


Fig. 4. Block diagram of solid model reconstruction system.

projection, then reconstruct 3D models from the views, it is possible to construct the virtual environment with a low cost. One of these approaches is to reconstruct a virtual building from architectural drawings and to walk-through or fly-through along the reconstructed virtual space.

However, there are a lot of limitations in the shape of objects that can be reconstructed. It is very hard, or even impossible to reconstruct the 3D model for a complex object since key information may be lost during the projection. Too much loss of information makes it impossible to reconstruct a valid model from the projections. For these reasons, this 3D model reconstruction has been applied to several production areas such as CAD and CAM (computer-aided manufacturing) in which relatively simple objects are manipulated. Here we introduce an automatic drawing recognition system as a typical example. This system generates the high-level representation of an object from 2D drawing, and provides the results to CAM or CAPP (computer-aided process planning) system for efficient process planning and production.

2.1. *Automatic drawing recognition system*

As computer technology advances, conventional production processes become gradually replaced by computerized processes. Creation and management techniques of engineering drawings for product design have progressed in CAD area. Automated manufacturing techniques have been improved in CAM using computerized numeric controlled machining devices and CAM softwares. In CAPP area, much

work had concentrated on devising an optimal tool selection and tool path generation methods which enable us to manufacture products more efficiently with less raw materials.[4,6] Researches on CAD, CAM and CAPP have focused on computerizing only one step of whole production process. In these days, CIM (computer integrated manufacturing) that computerize whole processes — from design to manufacturing — systemically is highlighted.[13]

As the design and manufacturing process has been automated, we are interested in two kinds of applications. One is to devise a method that manages CAD drawings efficiently and enables us to reuse them for designing other products. The other is to find out an efficient internal representation that can be used throughout the whole process from design to manufacturing. Automatic drawing recognition system satisfies these requirements. This system reconstructs 3D solid models from 2D drawings made by designers, and then transforms them into high-level representations that describe the shape of the objects to be produced. The high-level representations of drawing information might be stored in an engineering database,[8] which then can be used for drawing information retrieval with a simple query statement when we want to design another product. High-level representation can be utilized as an input of the manufacturing or process planning step without any transformation.

Automatic drawing recognition system is composed of four modules. One is for acquiring engineering drawings. The second is 3D solid model reconstruction from 2D drawings. The third extracts high-level representation from the solid model. The last is for processing query statements and retrieving the drawing information that user wants to find. Figure 5 depicts the conceptual diagram of automatic drawing recognition system.

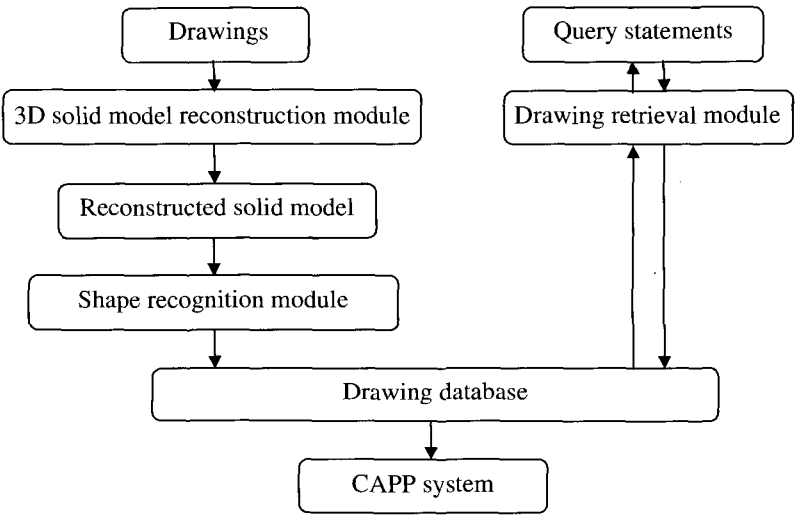


Fig. 5. Conceptual diagram of automatic drawing recognition system.

Automatic drawing recognition system offers several advantages. Firstly, we can implement efficient drawing management and process automation. CAD software has been common in product design, and most designers make tens or hundreds of drawings for the design of a product or a system since the complexity of the products has increased. Even though the number of drawings to be managed has dramatically increased, drawings are managed using primitive methods like tagging the date to be generated or by some annotation. In this environment, drawing retrieval time becomes too long, and sometimes the required drawing cannot even be found. The problem is more serious in hand-made drawings or hardcopies of CAD drawings. Consequently, designers have to repeat the same work since they cannot reuse the drawings that they have already made. Also it is impossible to refer to the meaning of drawings without designer's help when they cannot interpret the drawings. With the automatic drawing recognition system, designers can concentrate on the design process since it automatically classifies and retrieves the drawing that they want. Even when the number of drawings used for product manufacturing increases and the designers work on the distributed environment that makes them share a drawing database, it is very easy to work in that environment. Also, it reduces design time considerably since it enables us to reuse the drawings created by themselves or other designers. It helps us to construct computer supported cooperative work (CSCW) environment.[15,34]

Automatic drawing recognition system changes the basic scheme of conventional product manufacturing systems. As can be seen in Fig. 6, the conventional production process is composed of two steps. In the first step, designers make drawings for a product that they want to produce. In the next step, manufacturers interpret the meaning of drawings, and then make the product by manipulating some machining devices. In this process a mis-interpretation of the designers' drawings may result in a wrong object being produced; or production efficiency might be compromised when an inappropriate tool path is chosen.

Figure 7 shows the automated production process based on the drawing recognition system. It reduces the possibility of errors and improves the efficiency of the production process since it performs the reconstruction of 3D solid model automatically, extraction of high-level representation from the model, and determination of optimal tool path without human intervention.

The most important part of the drawing recognition system is 3D model reconstruction from input drawings.

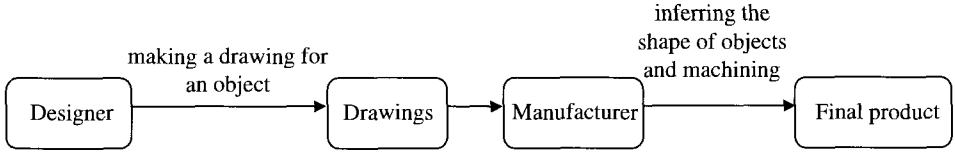


Fig. 6. Conventional product manufacturing process.

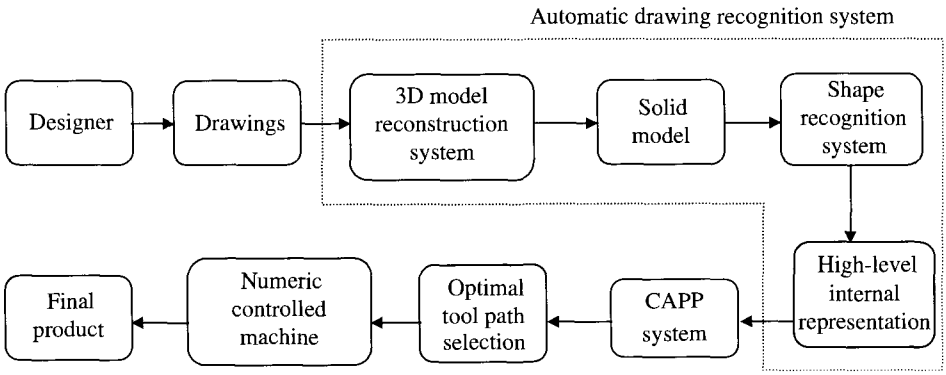


Fig. 7. Automated production process.

2.2. *Researches on 3D reconstruction using projections*

Research for reconstructing three-dimensional models from projections has been conducted in computer vision, image processing, and the computer graphics area since the 1960's.[31,46] However, it is not easy to reconstruct complete models except those that involve simple objects since a lot of geometric information may be lost during the projection. Some researchers have tried to reconstruct the lost information using heuristic search and pattern matching techniques. However these are not regarded as good solutions since they require severe restriction for input data, and they might produce wrong solutions.

As for orthographic projection, it is known that its loss of information is less than that of any other projection method. While perspective projection and oblique projection cannot preserve parallelism, shapes, and measures of geometric elements, orthographic projection maintains most geometric information except for depth information. However, even though we exploit orthographic projections, it is impossible to reconstruct all sorts of solid models perfectly using orthographic views since orthographic projections do not contain depth information. So its reconstruction process is not simple and it may cause some errors. It is impossible to reconstruct solid models for all kinds of objects with any reconstruction method that has ever been devised. Therefore, most research focused on developing methods that can reconstruct solid models for more various objects, and solve more pathological cases with greater speed.

Projection-based 3D model reconstruction methods are classified into single view methods and multiple view methods according to the number of projections to be used as their input. Also they can be categorized into the B-rep based method and CSG based method according to the type of resulting solid model. Since it is possible to transform a type of solid model into another one,[29,38,44] the type of representation for the result of reconstruction system does not matter. Boundary representation is mainly used in most reconstruction algorithms since it guarantees

maximum compatibility for representation of models and it can be applied to a variety of areas.

2.2.1. *Single view methods*

The single view method extracts the shape information of 3D objects using only one perspective projection or an oblique projection. It finds out the patterns of two dimensional geometric elements on the view that can be projected from three-dimensional geometric elements by applying some heuristics. Single view methods are mainly used in computer vision and pattern recognition. The labeling method[25,26] and gradient space method are the most popular of single view methods. In the labeling method, patterns are defined by three edges sharing a vertex that can be interpreted as valid 3D information, and then classified into four categories. Appropriate labels are tagged to an edge according to the shape of connection to the other edges on the view. We can infer the 3D shapes on that point by observing a pattern of labeled edges sharing a point. In the gradient space method, we infer the 3D shape of object using the gradients of two lines when those lines converge to a vanishing point in a perspective view. In addition, linear programming method,[42] perceptual method,[24] and primitive identification method[45] are classified as single view methods. These methods are used for inferring the shape of objects or visual inspection of products based on the scanned-data from hand-made drawings or photographs captured by a camera.

The single view method has advantages in that the amount of input data is relatively small and it can reconstruct 3D geometric information even when drawings have some erroneous elements. Also it can process hand-made sketches. However, since it can reconstruct the limited range of objects, it requires severe restrictions, and it may cause ambiguities. Therefore it cannot be adopted to applications that require accurate 3D model. Although the single view approach is very useful in solid model reconstruction and shape understanding, we will not describe the details since it is out of the scope of this chapter.

2.2.2. *Multiple view approach*

Multiple view methods generate 3D solid models using two or more views. Orthographic projection which projects an object onto the planes that are perpendicular to principle axes is mainly used rather than oblique projection or perspective projection. However, in some cases, cross-sectional views and detailed views of some parts may be used for more accurate solid model reconstruction.

Idesawa was the first to suggest a method of reconstructing a 3D model from multiple orthographic views.[18,19] He devised an algorithm that constructs limited polyhedral objects from three views. It consists of the following steps:

- Generate 3D vertices from 2D vertices on three orthographic views.
- Generate 3D edges by connecting the 3D vertices.

- Construct the closed faces from 3D edges on the same plane.
- Eliminate a ghost element that cannot be a part of three-dimensional model by applying some criteria.
- Construct 3D solid models from faces.

Although this method has some problems in that it can only reconstruct rectilinear polyhedras and may generate invalid solid models, this bottom-up approach becomes the basis of the methods proposed thereafter.

Wesley and Markowsky proposed an algorithm for reconstructing arbitrary polyhedral objects.[44] Their method is an extension of Idesawa's algorithm. Unlike the previously proposed methods that depend on heuristics and pattern matching, this method reconstructs solid models based on mathematical approach. It uses detailed views and cross-sectional views as well as two or more orthographic views to construct a more precise model. As a result, it can produce the solid model for an object composed of non-rectilinear edges and faces, and it solves problems of reconstructing invalid solid models. Also, it provides good results in pathological cases that could not be solved in the previous methods and produces all possible solutions in case that the orthographic views can be interpreted as several objects. However, this algorithm requires considerable processing time because it performs tremendous reprojections and complicated geometric operations. It is also limited to reconstructing polyhedral objects.

Aldefeld presented a method to interpret 3D models by using a pattern recognition technique based on heuristic search rather than mathematical approach such as Idesawa's method.[2,3] The key idea of the method is to find the heuristic rules that determine the type of patterns of geometric primitives found in views. The rules are applied to 2D primitives in each view and they are classified into different patterns in order to determine the shape of a 3D model. Although it is faster than Idesawa's and Wesley's methods, it can only deal with restricted objects and cannot interpret the views when parts of an object overlap each other for information loss in projection phase.

As an extension of Wesley's method, Sakurai proposed an algorithm to reconstruct objects composed of cylindrical, conical, spherical and toroidal surfaces that are parallel to one of principle axes by using the vertex type classification method.[39] In this method, vertices defined at junctions of straight lines and curved lines are classified into three categories — silhouette, tangency, and standard. Then standard vertices and edges connected to those vertices are used for reconstructing polyhedras by adapting Wesley's method, and silhouette and tangency vertices and edges connected to those vertices are used for generating curved surfaces by exploiting their own algorithm. However, because it is based on Wesley's method, it is not simple and requires as much processing time as Wesley's. Gu *et al.* provided a method to reconstruct more various curved faces by combining Wesley's and Aldefeld's methods.[16]

While some methods are not concerned about reducing the processing time, several works have been continued to reduce the processing time for solid model reconstruction. Yan *et al.* presented an algorithm for reconstructing a polyhedral object from orthographic projections based on Wesley's bottom-up approach.[52] In order to accelerate the processing speed of 3D edge generation step, the frontal-projection-based decision-tree search technique is used in this method. It uses the decision tree that represents the kind of patterns that can be a 3D edge. It enables us to generate 3D edges more rapidly than the existing method by examining patterns of geometric elements on three orthographic views that correspond to a node of decision tree. This method can also handle broken lines in the projections to get the accurate solution in case a set of views can be interpreted as several different objects. However, it does not accelerate overall speed sufficiently, since it considers 3D edge generation step. There have been some works on reducing 3D model reconstruction time using special data structures representing correlation between 2D geometric elements and three dimensional geometric elements.[40,41] In addition, there was some research for determining all possible solutions from orthographic projections that have some ambiguity using heuristic rules.[23,32] After constructing a wireframe model from three orthographic views, all solutions can be found one after the other in face generation step by applying some heuristics.

Instead of using three orthographic views, research that reconstruct solid model from two orthographic views have been continued. Wilde first attempted to reconstruct 3D model from only two views.[50] Dutta extended Wilde's work.[10] Although we can represent a symmetric object using only two views, it commonly causes much more ambiguity than when three views are used. It calculates the multiplicity of 2D vertices and edges, then reconstructs all possible solids from two views using combinatorial search. However, it is slow and it cannot generate a solid model if the third view that are not used as input has an intersection of two edges.

2.2.3. *Wireframe methods*

Several works for reconstructing 3D solid model from wireframe — an intermediate form between projections and solid models — are also interesting. Even though it does not use orthographic views as input, it can be classified as a kind of reconstruction method. Wireframe-based methods are important since a lot of CAD software provide wireframe generation facilities, and most projection-based methods can generate wireframes as an intermediate result. Markowsky proposed a method to reconstruct solid models from wireframes.[28] It became the basis of Wesley's bottom-up method that reconstructs solid models from orthographic projections. If we already have 3D vertices and 3D edges, no further processing is required for generating a wireframe. In this method, the conversion process from wireframe to solid model is very similar to that from pseudo wireframe to solid model in Wesley's method. The only difference is that Markowsky's method is easier to reconstruct solid models, since the wireframe does not have ghost elements. Agarwal proposed

another method to reconstruct solid models from wireframes. It decomposes input wireframes into atomic tetrahedras that cannot be divided any more. Then it examines whether a tetrahedral can be a part of an object or not by combining some tetrahedras one after the other. After determining the valid tetrahedras, we can reconstruct a 3D solid model by combining them.[1]

3. 3D Solid Model Reconstruction from Orthographic Views

In this section, we present the typical 3D model reconstruction method. It is based on Wesley's method and Yan's method that reconstruct polyhedral object[49,52] and Sakurai's method that produce a solid model containing curved surfaces.[39]

3.1. *Preliminary definitions*

In this section, we define the basic concept used in the 3D solid model reconstruction.

- Definition 1: A vertex v is a point in $\mathbf{R}^3$, and it is represented as a Cartesian coordinates (x, y, z) in $\mathbf{R}^3$ space.
- Definition 2: An edge e is a line segment defined by two endpoints v_i, v_j in $\mathbf{R}^3$.
- Definition 3: A face f is the closure of a nonempty, bounded, connected, coplanar subset of $\mathbf{R}^3$. The boundary of a face is the union of a finite number of line segments $e_1, \ldots, e_n$ and it is denoted by ∂f. Two adjacent edges that belong to ∂f do not share intersections except for their endpoints.
- Definition 4: An object O is the closure of a nonempty, bounded, connected subset of $\mathbf{R}^3$. It is defined as the union of a finite number of faces $f_1, \ldots, f_n$ and inside of them. The boundary of an object is the union of $f_1, \ldots, f_n$ and it is denoted by ∂O. The inside and outside of the object are represented as iO and cO respectively.
- Definition 5: A set of vertices of a face f, $V(f)$, is defined by intersections of each pair of edges comprising ∂f. That is, $V(f) = \{v | v \in e_i \cap e_j,\ e_i, e_j \in \partial f\}$.
- Definition 6: A set of edges of a face f, $E(f)$, is defined by all edges which belong to ∂f. The endpoints of an edge $e \in E(f)$ belong to $V(f)$. No interior point of the edge e belongs to $V(f)$ except for the endpoints.
- Definition 7: A set of vertices of an object O, $V(O)$, is defined by all vertices shared by three or more non-coplanar faces. That is, $V(O) = \{v | v \in f_i \cap f_j \cap f_k,\ f_i, f_j, f_k \subseteq \partial O\}$.
- Definition 8: A set of edges of an object O, $E(O)$, is defined by all edges which belongs to ∂O. The endpoints of an edge $e \in E(O)$ that are contained in $V(O)$ and no interior point of e belongs to $V(O)$ except for the endpoints. The edge should be shared by only two non-coplanar faces.
- Definition 9: A wireframe of an object O, $WF(O)$, is defined by the union of $V(O)$ and $E(O)$, that is, $(V(O), E(O))$.
- Definition 10: Let O be an object and P represent a plane in $\mathbf{R}^3$ space. The orthographic projection $\pi : \mathbf{R}^3 \rightarrow P$ is denoted by $O|P$. The images under the

projection $O|P$ are a set of points and line segments on P that are denoted by $V(O|P), E(O|P)$ respectively. An edge that is not perpendicular to P is projected as a line segment according to $O|P$. A set of those line segments is denoted by E^P. All points $p \in V(O|P)$ are endpoints of edges contained in E^P and they are shared by two non-collinear line segments. $E(O|P)$ is a set of line segments s defined by two points $p_1, p_2 \in V(O|P)$ and no interior point of s belongs to $V(O|P)$ except for the endpoints.

3.2. *Solid model reconstruction algorithm*

Figure 8 shows the block diagram of the typical 3D model reconstruction algorithm. Three orthographic views are transferred to 3D vertex and 3D edge generation step after preprocessing. In these stages, the algorithm classifies the type of edges and vertices for reconstructing curved surfaces, and then constructs wireframe by segmenting intersecting and overlapping edges. It generates planar and curved faces from the wireframe constructed in the previous steps. In case of intersecting two faces in $\mathbf{R}^3$ space, it segments them into four non-intersecting, non-overlapping faces by inserting cutting edges. It constructs virtual blocks which are small components comprising the solid model of the object by connecting the curved and planar faces. It eliminates ghost elements that cannot be used for constructing final

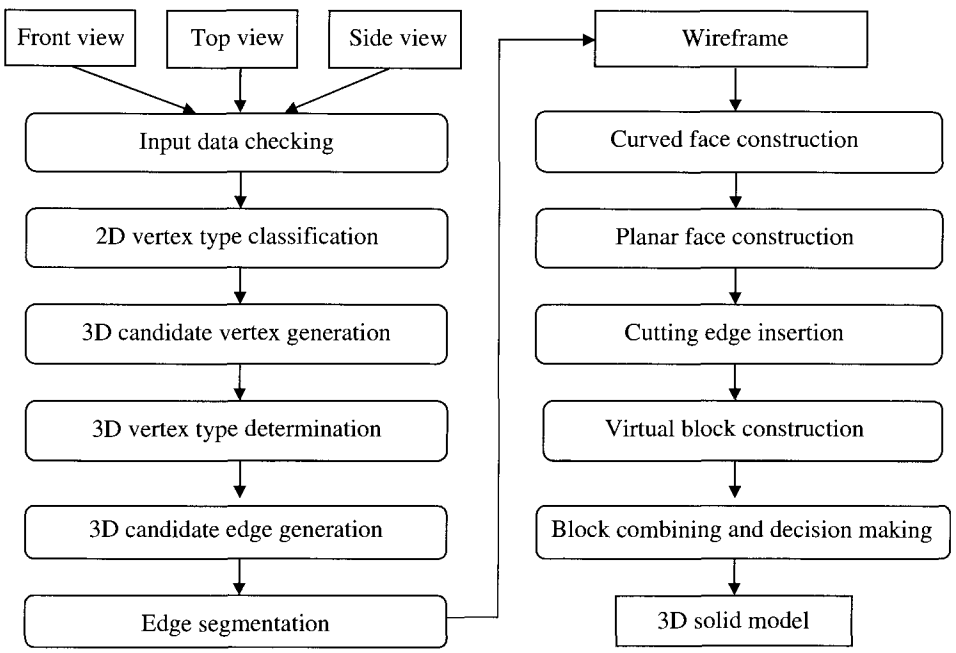


Fig. 8. Block diagram of the solid model reconstruction algorithm.

model. Finally, it completes the 3D solid model of the object by combining the remaining virtual blocks.

Among the 11 steps in Fig. 8, the major steps are 3D candidate vertex generation from 2D vertices, 3D candidate edge generation from 3D vertices, face construction by connecting 3D edges, and solid model reconstruction by connecting the faces. Whole process can be represented mathematically as follows.

Let f be the mapping function from a 3D object O and its projections P, then we can get the relationship $P = f(O)$. Three-dimensional model reconstruction is to find an inverse mapping f^{-1} such that $\bar{O} = f^{-1}(P)$, where $\bar{O}$ is a 3D solid model of the object. Inverse mapping f^{-1} can be decomposed into several functions as follows:

$$f^{-1}(P) = f_{SL}(f_{BL}(f_{PL}(f_{LN}(f_{PN}(P))))) \tag{3.1}$$

where f_{PN} is the mapping function from 2D vertices to 3D vertices, f_{LN} is mapping function from 3D vertices to 3D edges. f_{PL} is the mapping between 3D edges and faces and f_{BL} is the function from faces to virtual blocks. Lastly, f_{SL} is the function for generating final solid model by combining candidate blocks.

Consequently, we can simplify the problem by dividing the whole reconstruction process into small independent sub-steps. Also we can get more accurate solution by eliminating ghost elements efficiently in each step. Now we present the detailed description for each step of the whole reconstruction process.

3.2.1. *Input data checking*

Three orthographic views are commonly used as the input of 3D solid model reconstruction algorithms. Orthographic projection uses principle planes $XY(z = 0)$, $YZ(x = 0)$, $ZX(y = 0)$ defined in world coordinates as projection planes. We assume that each view is composed only of 2D geometric primitives such as points and line segments. Since drawings have some non-geometric elements including annotations, comments, dimensions, and measures, it is required to extract those elements from geometric primitives.[9]

When the position of endpoints of two connected line segments does not match in a view, we should let them be the same points by adjusting coordinate values for the endpoints. If the distance between two points is less than predefined threshold value, those points are regarded as the same ones.

As described in Definition 10, any 2D vertex in each view must be an endpoint of two or more line segments that do not lie on the same straight line. Otherwise the vertex and edges connected to it are regarded as ghost elements and they should be eliminated.

After preprocessing step, 2D vertices and edges in each view are stored in 2D vertex lists and 2D edge lists for each view respectively. Each item of 2D vertex list has 2D coordinates for each vertex, and each item of 2D edge list holds indices of its two endpoints.

3.2.2. *2D vertex type classification*

2D vertices on each view have one of the following attributes according to the shape of 2D edges sharing the vertex — straight line or curved line — and the topological relation of the edges.

- Definition 11:

(1) A *silhouette* vertex is an endpoint of an artificial line segment generated by projection. It is located on the point where an arc is tangent to a straight line segment parallel to a principle axis.
(2) If two arcs or a line and an arc are tangent to each other, then *tangency* type is assigned to their common endpoint.
(3) Vertices in other views corresponding to silhouette vertex are *silhouette-generated.*
(4) Vertices in other views corresponding to tangency vertex are *tangency-generated.*
(5) Remaining vertices have *standard* type.

It is necessary to classify the type of vertices in order to reconstruct curved faces in the following step. Standard vertices are used for constructing planar faces, silhouette and tangency vertices are used for generating curved-faces. Figure 9 depicts types of 2D vertices on each view.

Standard and tangency vertices appear on orthographic views, but the other types of vertices cannot be found on input views since they are artificial points. So they must be appended to 2D vertex list in this step.

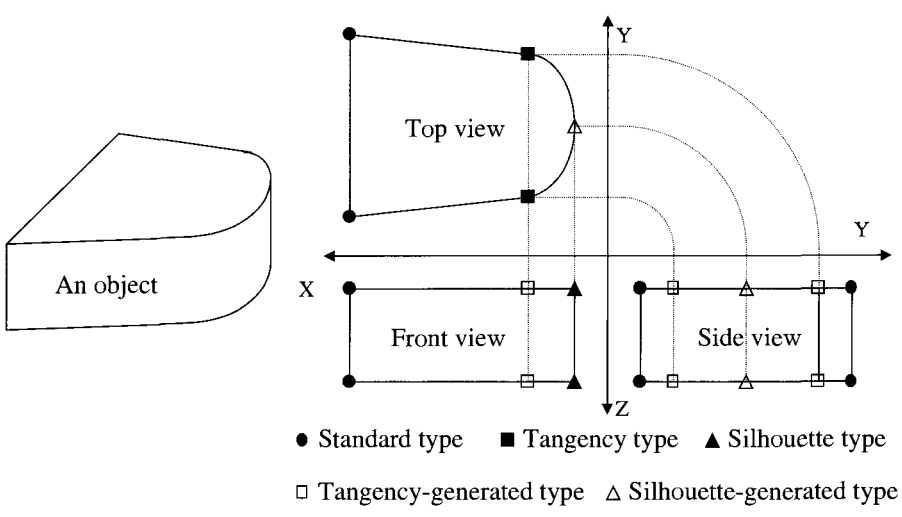


Fig. 9. Types of 2D vertices.

3.2.3. *3D candidate vertex generation*

Arbitrary plane in $\mathbf{R}^3$ space can be used as projection plane. There exists a projection with which we can correspond each 3D vertex and 3D edge to a unique 2D point and a 2D line segment, where the mapping is called *distinguishing projection.* When a 3D vertex is projected on three non-coplanar planes using distinguishing projection, we have only to find intersection of three lines that are perpendicular to those planes and pass through the 2D points which are the images of that vertex.

Non-distinguishing projection corresponds two or more vertices and edges to a single 2D geometric primitive. If a projection ray fired from the center of projection meet with two or more 3D geometric elements before it reaches projection plane, it is regarded as non-distinguishing projection. In that case, an ambiguity problem may occur. That is, we cannot recognize the correspondence between a 2D geometric primitive and 3D geometric primitives. In Fig. 10, since two 3D vertices v_1, v_2 intersect with projection ray l_1, and a 3D edge e_1 is parallel to l_1, images for those primitives are overlapped on a 2D point p_1.

Most orthographic projection is non-distinguishing, since objects are aligned so that their edge and face might be parallel or perpendicular to projection plain. Therefore, we should consider the ambiguity problem in 3D vertex generation step.

If a 3D vertex is projected onto three views, its image should be an intersection point of two or more non-collinear line segments in at least one view. So we can generate a 3D vertex from a pattern of 2D geometric elements that satisfy the above property.

A 3D vertex can be classified into two categories. One is represented as an intersection point of two non-collinear line segments in two or more views, and it is called *class I vertex.* The other is represented as intersection point of two line segments in only one view, and it is called *class II vertex.* The reason why we consider class II vertices is that there are some collinear edges that share the endpoint of edges contained to a set of edges $E(f)$. Since the set of class I vertices and class II

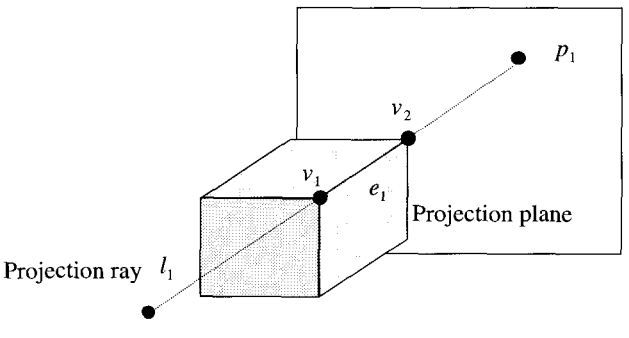


Fig. 10. Ambiguity problem in non-distinguishing projection.

vertices are disjoint, a set of 3D vertices is the sum of class I vertices and class II vertices. Figure 11 shows an example of a class I vertex and a class II vertex. In this figure, v_1 is class I vertex since it appears on three views as intersection points of two non-collinear line segments, and v_2 belongs to class II since it is represented as an intersection point of two lines only in a side view. Class I vertices are generated in this step, and class II vertices are reconstructed in edge segmentation step.

An intersection point of two non-collinear line segments on a view may be an image of a 3D vertex. Points on the remaining views that correspond to that point have the same coordinate value in spares coordinates. For example, when we compare the points on top view parallel to XY plane, and front view parallel to XZ plane, points that have the same x-coordinates can be corresponding points. In case of using three projection planes P^1, P^2, P^3, it selects a pair of corresponding points $(p^1(x,y), p^2(x,z))$ in two views and reprojects it onto the third view. If an image on the third view, $\tilde{p}^3(y,z)$, is contained in $V(O|P^3)$ or if it is an interior point of an edge $e \in E(O|P^3)$ the generated vertex $v^*(x,y,z)$ may be a 3D vertex. Figure 12 shows how 3D vertices are generated from 2D vertices.

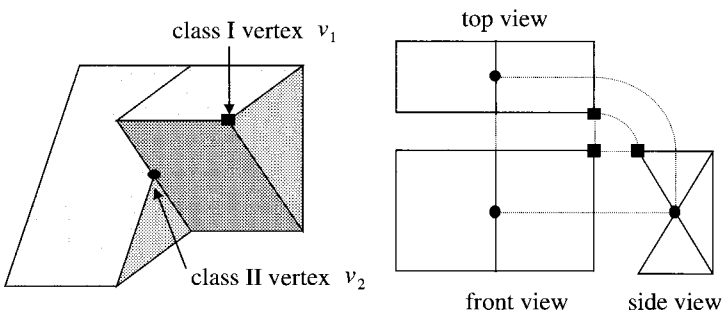


Fig. 11. An example of a class I vertex and a class II vertex.

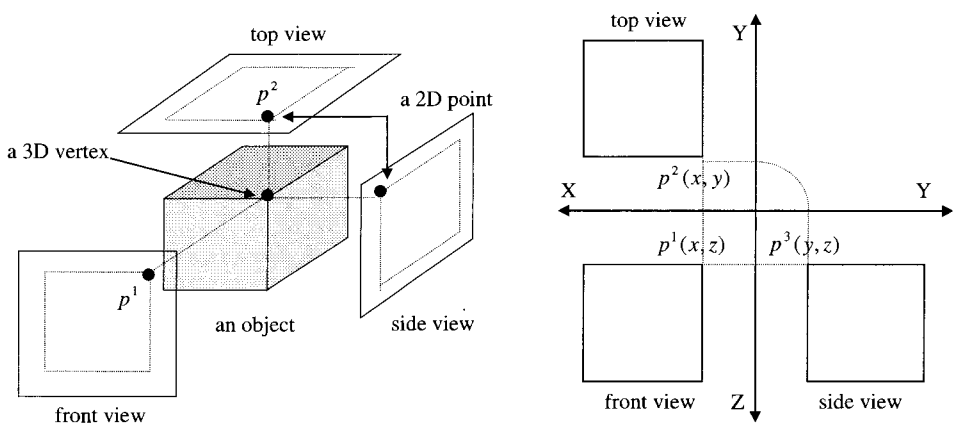


Fig. 12. An example of generating 3D candidate vertices from three orthographic views.

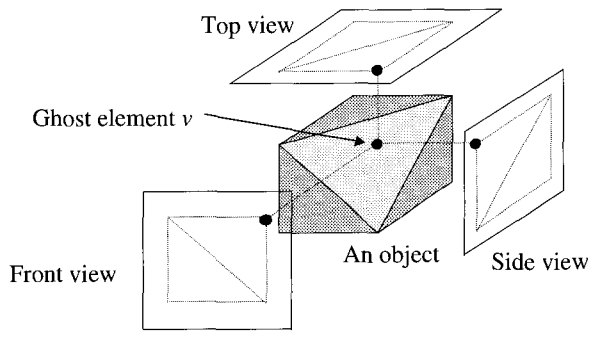


Fig. 13. Mis-interpretation of three views for information loss during the projection.

Points that are not part of an object can be interpreted as vertices of the object for information loss. In Fig. 13, even though a vertex v is not contained in $V(O)$, it looks like a real vertex when we examine the 2D geometric primitives on a view. Since the depth information may be lost during projection, a set of 3D vertices generated by the above criterion, $V^*(O)$, may contain ghost elements that are not part of an object. So we can get $V^*(O) \supseteq V(O)$. $V^*(O)$ is called the set of candidate vertices or c-vertices hereafter.

3.2.4. *3D vertex type determination*

After coordinate values of c-vertices are determined, their types are assigned according to the types of the corresponding 2D vertices. These are essential to constructing curved faces. Types of 3D vertices are determined by the following definition.

- Definition 12:
  (1) If the corresponding vertex in one view has silhouette type and the other two vertices have silhouette-generated type, then the c-vertex is silhouette type vertex.
  (2) If the corresponding vertex in one view has tangency type and the other two vertices have tangency-generated type, then the c-vertex has tangency type vertex.
  (3) If three corresponding vertices have standard type, then the c-vertex becomes a standard vertex.

Figure 14 shows the relationship between types of c-vertices and types of their corresponding 2D vertices.

3.2.5. *3D candidate edge generation*

In this section, we explain the process of generating all types of 3D edges by means of the types of their endpoints. 3D candidate edges from c-vertices are classified into standard edges, silhouette edges and tangency edges.

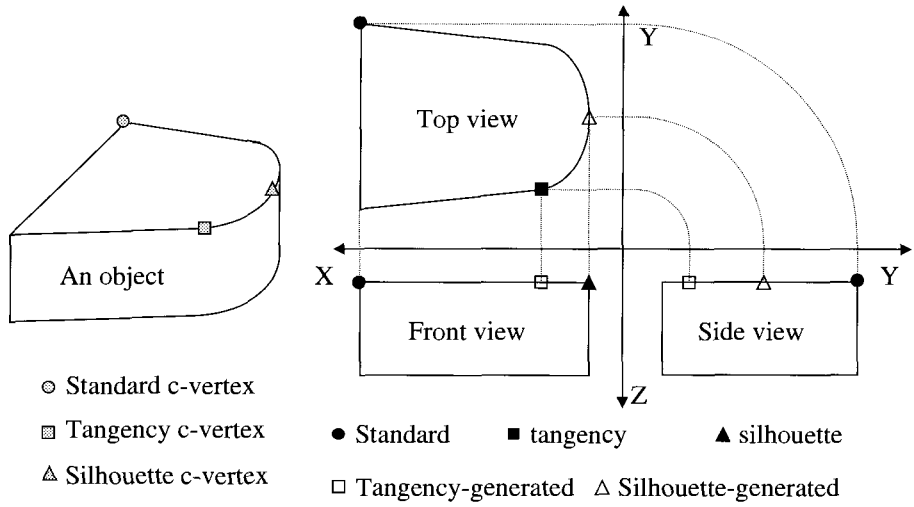


Fig. 14. Determination of types of c-vertices using the types of 2D vertices.

(1) *Generating standard candidate edges*

When 3D edges are projected on three views, those perpendicular to the projection plane are represented as 2D vertices, and the others are projected as 2D line segments. So, all vertices and edges on a view can be the images of 3D edges. We can reconstruct 3D edges based on the following property. Firstly, it selects a pair of c-vertices in candidate vertex list, and then reprojects the edge $\tilde{e}$ generated by connecting the pair onto the three views. If its images are contained in $V(O|P)$ and $E(O|P)$ in each view, then $\tilde{e}$ may be a 3D edge. Because the set of c-vertices may contain ghost elements, edges that cannot be contained in $E(O)$ may be generated. All the edges generated by the above criterion are called candidate edges or c-edges and their set is denoted by $E^*(O)$. Figure 15 shows an example of reprojecting a line segment $\tilde{e}$ obtained from connecting a pair of c-vertices (v_1, v_2). $\tilde{e}$ is projected as two line segments s_1 and s_2 in two views and a point p_3 in the third view, so it is a c-edge.

(2) *Generating silhouette candidate edges*

A silhouette edge is an artificial one corresponding to a silhouette line of a curved face. As shown in Fig. 16, it appears when a curved face is tangential to an artificial face perpendicular to a principle axis.

The following property is used to generate a silhouette edge from the information of 2D primitives on orthographic views and c-vertices generated in the previous step. A silhouette edge parallel to a principle axis is projected as a line segment of which endpoints have silhouette-generated type in one view, a silhouette type vertex in another view, and is not projected at all in the third view.

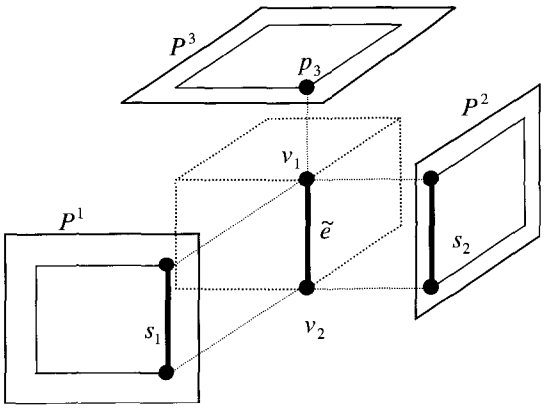


Fig. 15. An example of reprojection for candidate edge generation.

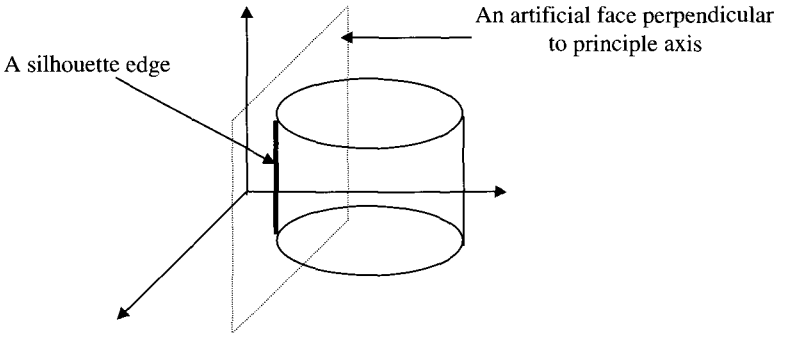


Fig. 16. An example of silhouette edge.

As shown in Fig. 17, a silhouette edge is generated when a curved face is tangent to an artificial plane parallel to principle axis. When the face and the plane are projected onto the three planes, the face is projected as an arc s and two faces f_1, f_2. The artificial plane is projected as two straight lines l_1, l_2. The silhouette edge is projected as a 2D vertex p that is an intersection point of the arc s and one of l_1, l_2. And it is also projected as a line segment s' that is an intersection of one of f_1, f_2 and the other line. p has silhouette type since it is an intersection of an arc and a line tangent to each other. s' has the same coordinate as the endpoints of p in the shared coordinates, since p and s' are the images of the same 3D edge. So the endpoints of s' have silhouette-generated type.

To generate a silhouette edge, a 2D edge whose endpoints have silhouette-generated type is selected. If there exists a pair of c-vertices which have silhouette type and the same coordinate value in shared coordinates, then the line segment obtained from connecting this pair of c-vertices may be a silhouette c-edge.

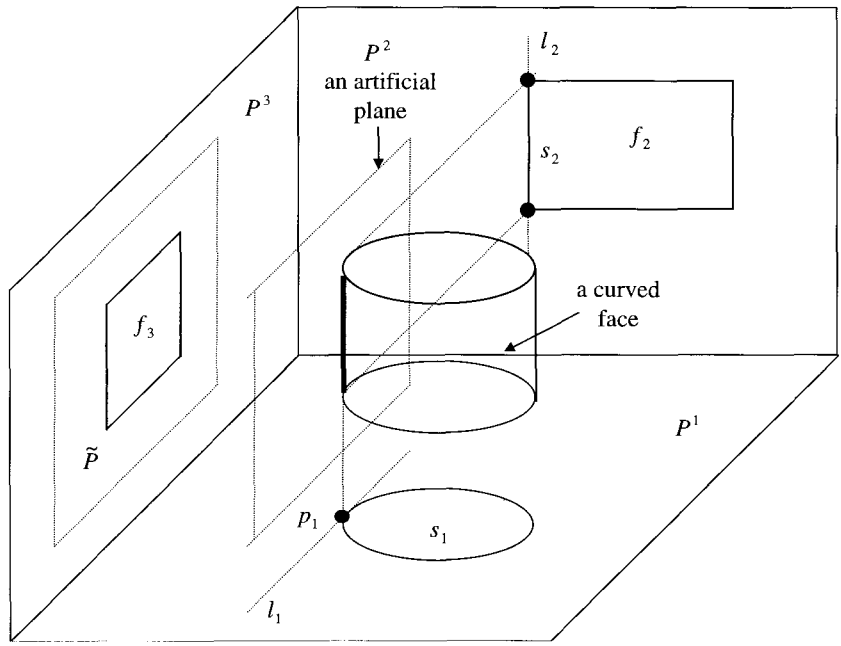
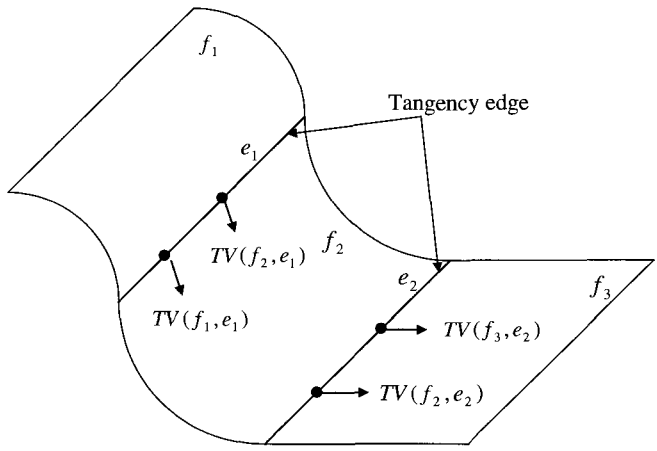


Fig. 17. Projection of a silhouette edge.



$TV(f_1,e_1)=TV(f_2,e_1)$ $TV(f_2,e_2)=TV(f_3,e_2)$

Fig. 18. Tangency edge: $TV(f,e)$ stands for a tangent vector of a face f on a edge e.

(3) *Generating tangency candidate edge*

A tangency edge is shared by two curved faces which are tangent to each other as shown in Fig. 18. Two faces sharing a tangency edge have C^1 continuity.

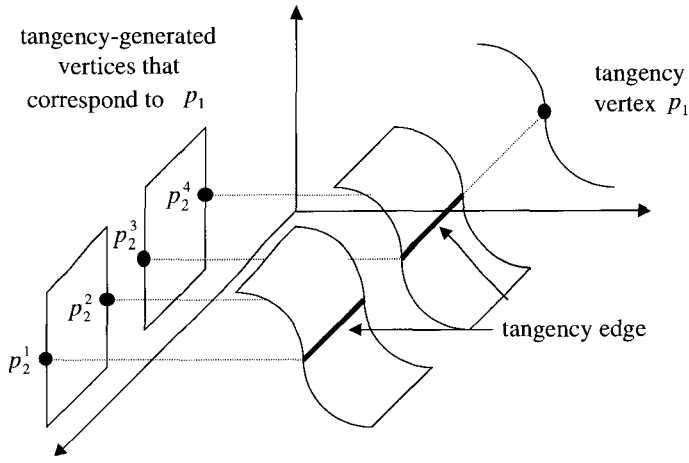


Fig. 19. An example of corresponding two or more tangency-generated vertices in a view to a tangency vertex in another view.

A tangency edge is projected as a tangency vertex in only one view. At first, it selects a tangency vertex in a view and finds tangency-generated vertices that correspond to the vertex in the other views. It should find all tangency-generated vertices that have the same coordinate values in shared coordinates in the other two views. If there exists a pair of c-vertices which has tangency type and the same coordinate value in shared coordinates, then the line segment obtained from connecting this pair may be a tangency c-edge. As shown in Fig. 19, two or more tangency-generated vertices that meet with the above condition can appear on a view. Since two parallel curved faces cannot be adjacent, the number of tangency-generated vertices corresponding to the edge is always $2n$ ($n = 1, 2, 3 \ldots$). We have only to examine 2D edges defined by a pair of vertices $(2n - 1, 2n)$. Since edges generated by pairs of vertices $(2n, 2n + 1)$ cannot generate a tangency edge, they are ignored.

3.2.6. *Edge segmentation*

Two non-collinear edges may intersect in $\mathbf{R}^3$ space and several collinear edges can be overlapped. In case of intersecting two non-collinear edges, those are segmented into four edges and a c-vertex is generated as shown in Fig. 20(c). Since overlapped edges may cause ambiguity problem, they should be segmented into smaller non-overlapped edges and duplicated edges should be eliminated as shown in Fig. 20(d).

Since class I vertices have been generated in the previous step, we can get all the vertices if we generate class II vertices. A class II vertex is an intersection point of two non-collinear c-edges except for their endpoints.

A class II vertex is projected as an intersection of non-collinear 2D line segments in only one view and interior points of line segments in the other two views. Let

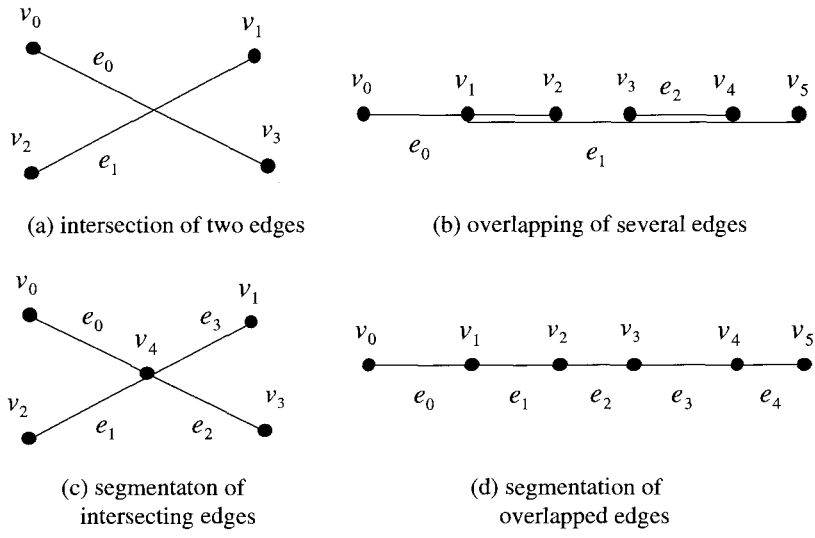


Fig. 20. Segmentation of intersecting and overlapped edges.

s_1 and s_2 be the 2D line segments. A vertex is a class II vertex if and only if all edges projected to s_1 and s_2 lie on a plane P perpendicular to the two views simultaneously. The edges should intersect at a point on P because its image on the third view should be an intersection point of non-collinear line segments. Therefore a class II vertex is an intersection point of two non-collinear c-edges.

When a class II vertex is generated by intersecting two candidate edges, the edges are segmented into four edges. The edges connected to the intersection point should be shared by only two faces which should not be coplanar. Thus a class II vertex is generated when the vertex is shared by at least four faces and two adjacent faces are not coplanar as depicted in Fig. 21. In this figure, two candidate edges e_1, e_2 of a face f are located on the same collinear line l.

If we find all class II vertices and store them into the set of candidate vertices $V^*(O)$, and segment candidate edges, and then store them into the set of candidate edges $E^*(O)$, the union of these sets $(V^*(O), E^*(O))$ contains all vertices and edges comprising pseudo-wireframe of the object.

3.2.7. *Curved face construction*

In this step, we construct quadric surfaces such as cylinders, tapers, torus, and fillet. These faces can be reconstructed by using the type of candidate vertices and edges, and 2D geometric primitives on orthographic views since those faces are parallel to a principle axis. The construction method for each curved surface depends on the shape of the faces.

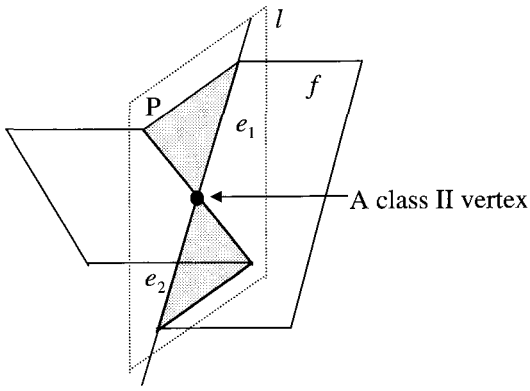


Fig. 21. Generating a class II vertex.

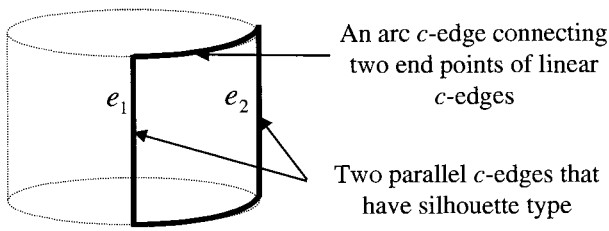


Fig. 22. Construction of a cylindrical face.

(1) *Constructing cylindrical face*
If there exists an arc candidate edge that connects endpoints of two linear c-edges whose types are silhouette, the face which contains these three edges becomes a cylindrical face. In order to construct cylindrical faces, it selects a pair of parallel edges (e_1, e_2) that have silhouette type, and then examines whether an arc connecting endpoints of the two edges on a view or not. Figure 22 shows an example of constructing a cylindrical face.

(2) *Construction of fillet face*
If there exists an arc candidate edge that connects endpoints of two linear c-edges whose types are tangency, the face that contains these three edges becomes a fillet face. Figure 23 shows an example of constructing a fillet face.

(3) *Constructing a toroidal face*
To find a toroidal face, patterns of 2D edges in each view are used. As shown in Fig. 24, a toroidal face is constructed if two arcs in different views have the same radius, the center points of the two arcs have the same coordinate value in the shared coordinates, and the distance from center points of the two arcs to the center point of an arc in the other views are equal. When a toroidal face is projected onto each

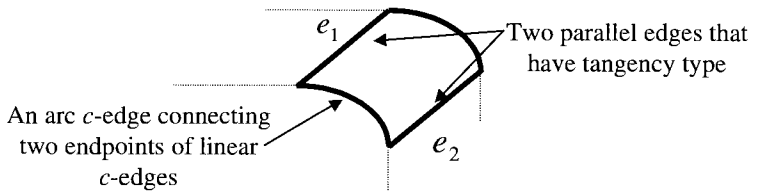


Fig. 23. Construction of a fillet face.

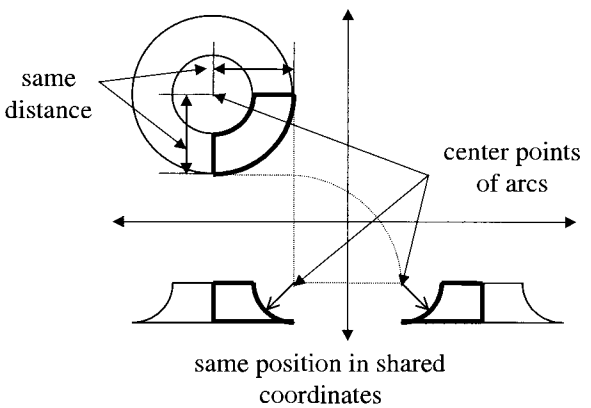


Fig. 24. Conditions to be a toroidal face.

view, arcs are presented in all views. Therefore in at least one view, two endpoints of the arc have both silhouette and silhouette-generated type. For the construction of a torus, vertices whose types are silhouette and silhouette-generated are selected, and only the related vertices and the corresponding arcs are examined.

As depicted in Fig. 25, the internal representation of a curved face is the same as that of a planar face, but the external shape of a curved face is approximated by polygon meshes. Since the main goal of this research is not the rendering of the shape of an object, an accurate surface approximation is not required. A curved surface is decomposed to same-sized polygons along with a principle axis. For example, a cylindrical face is represented as four rectangles, and a torus is represented as sixteen quadrilaterals.

3.2.8. *Planar face construction*

A set of faces that can be constructed from candidate vertices and edges of pseudo-wireframe $WF^*(O)$ is composed of real faces ∂O as well as ghost faces that do not belong to ∂O. In this step, we should reconstruct all possible faces since we cannot distinguish real faces from ghost ones. All the faces that belong to sets of real faces and ghost faces are called virtual faces. Ghost faces may appear when two faces

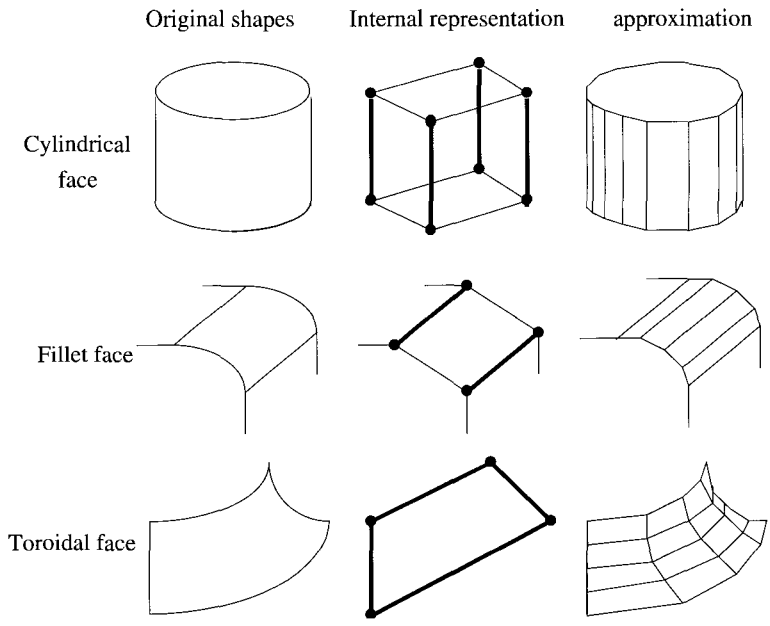


Fig. 25. Internal representation and external shapes for each curved face.

that are not actually adjacent are interpreted as adjacent faces, or a real face is regarded as an empty space for the information loss during the projection.

A 3D edge is shared by two non-coplanar faces according to Definition 8. Since two planes P_1, P_2 on which the faces f_1, f_2 are located are not parallel to each other, the 3D edge can be used for constructing only one face to which it belongs. However, an edge is regarded as if it is shared by two distinct faces on the same plane for information loss during the projection. After generating the wireframe model for an object in Fig. 26(a), only a face ABED can be a real face in Fig. 26(c). However, since we cannot find real faces, we have only to generate all possible faces. In this example, three virtual faces ACFD, ABED, and BCFE can be constructed from candidate edges located on a plane P. Although a candidate edge BE belongs to a real face ABED, it is interpreted as a part of faces ABED and BCEF. For this reason, an edge should be referred twice to the opposite direction in a face construction step.

Constructing a virtual face is to find a loop of edges comprising the boundary of a virtual face. An edge loop is a set of edges on the same plane, and it is denoted as $L = \{e_1, e_2, \ldots, e_n\}$, where an intersection point of two edges $e_i, e_j \in L$ is contained in the set of vertices of that face, and endpoints of each edge can be shared by only two edges.

After selecting a plane P to which the edges belong, it finds all edge loops on the plane. Then it selects another plane that is not coplanar to the plane P and

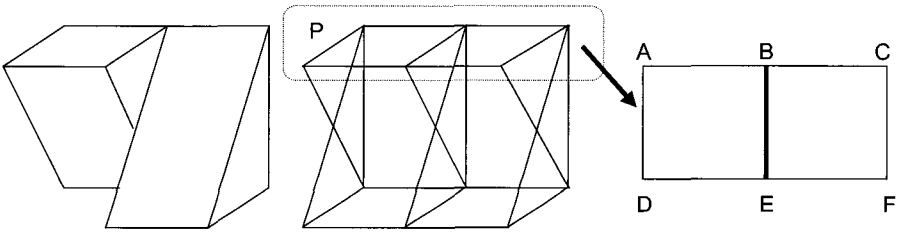


Fig. 26. An example that an edge is interpreted as shared edge of two faces.

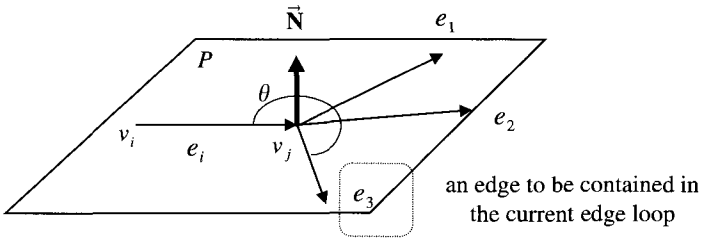


Fig. 27. Selecting an edge to be contained in the current edge loop.

contains edges of already constructed edge loops. All edge loops on a plane can be constructed by using the following procedure.

(1) Select an edge $e_i(v_i, v_j)$ not referred twice or referred once, and set it as the starting edge.
(2) Select a direction of traverse, it may be clockwise or counterclockwise.
(3) If the number of edges connected to v_j is 1, insert the adjacent edge e_j into the current edge loop L_e.
(4) If the number of edges connected to v_j is greater than 1, choose an edge e_j among them and store it into L_e.
(5) Repeat step (3)–(4) until the starting edge appears again in the current direction of traverse.
(6) Repeat step (1)–(4) until all edges are referred twice.

As shown in Fig. 27, if there are several edges connected to an endpoint v_j of an edge e_i, choose an edge e_3 which has the largest angle to e_i in the current traveling direction as a component of the current edge loop.

Figures 28 and 29 depict the process for constructing edge loops on a plane.

Bridges may occur in the edge loop construction process. A bridge is a kind of anomaly caused by information loss during the projection and occurs when an edge belongs to an edge loop twice. So, we must check whether an edge is already

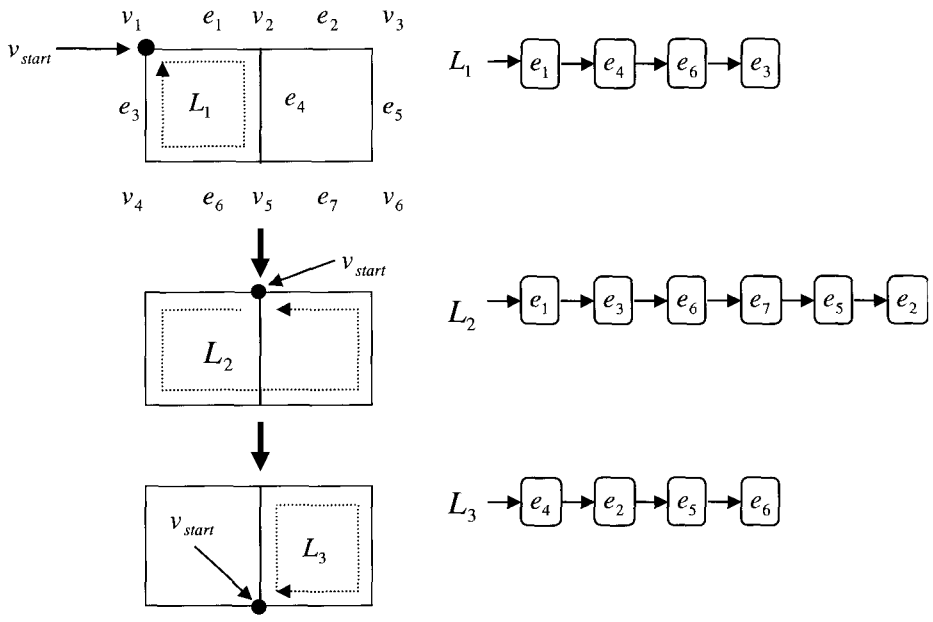


Fig. 28. Procedure of the edge loop construction.

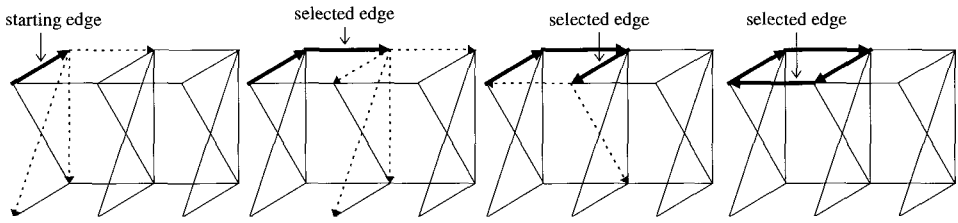


Fig. 29. The procedure for constructing L_1 in Fig. 27. Edges to be examined are drawn in dotted lines.

contained in the current edge loop whenever the edge is inserted. A bridge should be eliminated from the candidate edge list.

3.2.9. *Cutting edge insertion*

After generating all virtual faces, two non-coplanar faces may intersect as depicted in Fig. 31. In this case, one is a real face, and the other is a ghost one. Since it is impossible to know which one is real in this step, we segment them into four distinct faces by inserting a cutting edge. The segmented faces are tentatively regarded as valid ones until virtual blocks are constructed.

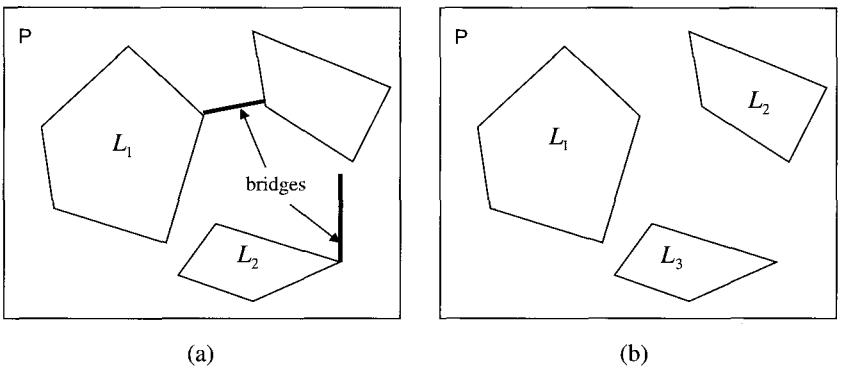


Fig. 30. Bridge problems. (a) An example of bridges; (b) elimination of the bridges.

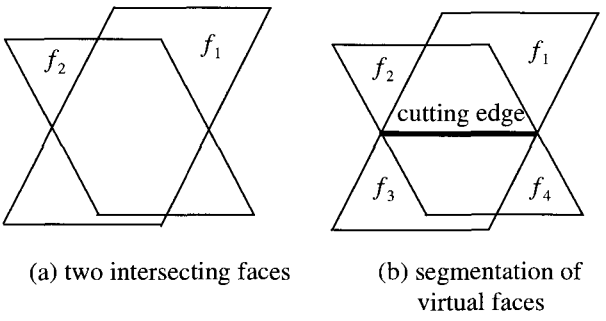


Fig. 31. Inserting a cutting edge and segmentation of faces.

3.2.10. *Virtual block construction*

Because of ghost faces, an object may be decomposed into smaller sub-objects called *virtual blocks* in solid model reconstruction. A virtual block B is defined as non-coplanar faces $f_1, f_2, \ldots, f_n$ and with its interior surrounded by them. If two distinct faces f_i, f_j are adjacent, there must exist an edge that belongs to $\partial f_i \cap \partial f_j$, and an edge is shared by only two virtual faces.

In most cases, a lot of ghost faces may appear because of information loss. As a result, a single solid may be decomposed into several blocks, and some virtual blocks that cannot be a part of the object are generated. As shown in Fig. 32, an object is divided into six blocks and two of them cannot be used to reconstruct the object.

To construct a virtual block, a face loop that comprises the boundary of the block must be generated. The list of virtual faces is used as input of this step. An item for each virtual face f holds a list of edges $e_i \in \partial f$. Edges are stored in the edge loop according to their order, and their direction vectors are aligned in a clockwise direction with respect to the normal of the face that contains them.

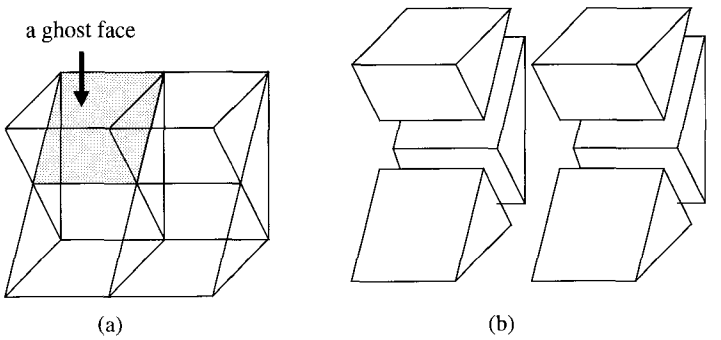


Fig. 32. An example of candidate blocks of an object. (a) A ghost face; (b) reconstructed virtual faces.

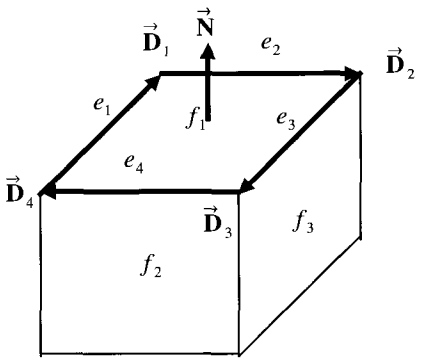


Fig. 33. Order of edges and their direction vectors in a cube.

While a real face belongs to a single block, a ghost face is shared by two virtual blocks as shown in Fig. 31. If we regard the open space surrounding an object as a block, all virtual faces belong to two adjacent virtual blocks. So a virtual face must be referred twice to construct a face loop as in edge loop construction.

In order to make a virtual block, we should construct a loop of faces that surround the block. At first, an arbitrary face f is selected. Then we find faces adjacent to the face f for each edge $e_i \in \partial f$ and insert them into the current face loop. After finding all adjacent faces for all edges e_i, select another face f' in the face loop, and then repeat the same process for each edge $e_i' \in \partial f'$. A virtual face is generated when all faces in the loop are examined.

The most important thing is to find adjacent faces for each edge that comprises the boundary of a face. If there exists a shared edge between two faces, those two faces are regarded as *connected.* If two faces are connected and belong to the same block, they are called *adjacent.* If only one face is connected to a face, that is an adjacent face at the same time. If the number of connected faces is greater than one, the face whose direction vector $\vec{\mathbf{D}}$ makes the smallest counterclockwise angle

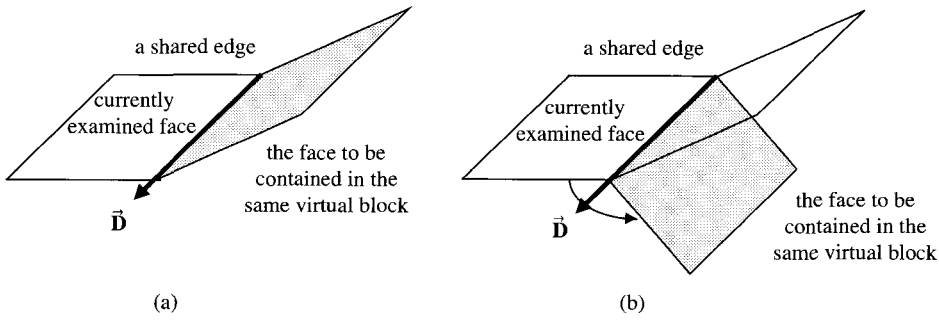


Fig. 34. Selecting an adjacent face for constructing a face loop. (a) Number of adjacent face = 1; (b) number of adjacent faces > 1.

with the currently examined face becomes an adjacent face. Procedures for a virtual block construction is summarized as follows:

(1) Select a face in the virtual face list, and insert a face in the virtual face list into the current face loop L_f.
(2) Select a face that is not referred twice and find adjacent faces for each edge according to the following condition.
(3) (a) If the number of connected faces is only one as shown in Fig. 34(a), store that face into the current face loop since it must be the adjacent one.
(b) As shown in Fig. 34(b), if the number of connected faces is greater than one, the face whose direction vector $\vec{\mathbf{D}}$ makes the smallest counterclockwise angle with the currently examined face becomes an adjacent face.
(4) Repeat step (3) until we examine all edges twice, we can get all the faces adjacent to the face.

Figure 35 shows an example of the construction of c-blocks for an object.

3.2.11. *Virtual block combining and decision making*

Since an object can be represented as the union of virtual blocks, the last step is to construct a B-rep model by assembling valid blocks with a gluing operation. A set of virtual blocks generated in the previous step may contain ghost elements that cannot be a part of the object, we should decide whether a block is a part of an object or not. To decide the state of a specific block B_i, we project $O^* = \cup_{k=1}^{i} B_k$ to three views, which is the union of valid blocks $B_1, \ldots, B_{i-1}$ and B_i to be examined. If $V(O^*|P) \subseteq V(O|P), E(O^*|P) \subseteq E(O|P)$, B_i is a part of the object. If there exists a vertex $p \in V(O^*|P), p \notin V(O|P)$ or an edge $s \in E(O^*|P), s \notin E(O|P)$, B_i cannot be a part of the object. It assigns *accept* state to a valid block and remaining blocks have *reject* state. A solid model can be reconstructed by assembling accepted blocks.

Figure 36 shows an example of determining the state of a virtual block. When B_1 is selected as an initially accepted block, three intermediate solids S_1, S_2, S_3

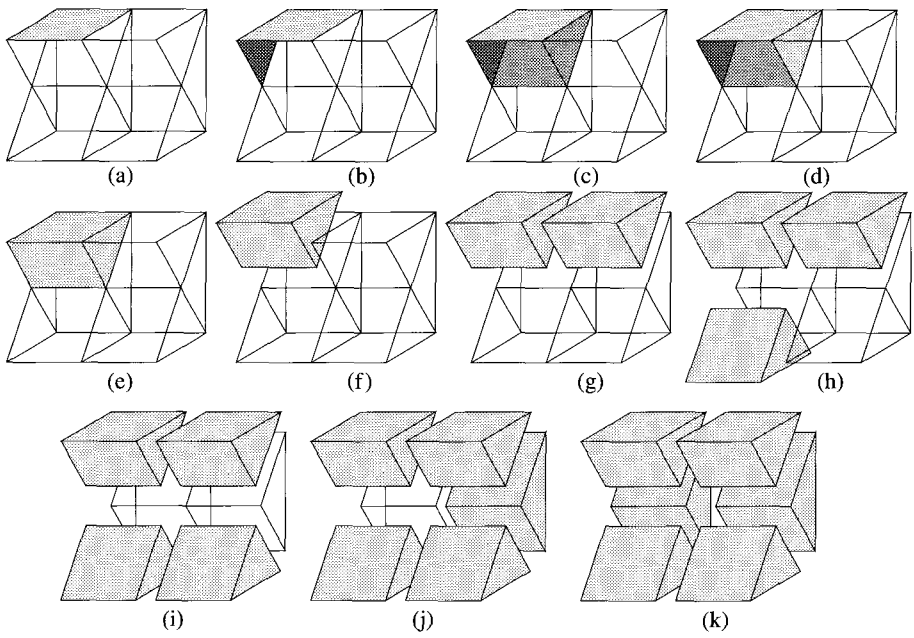


Fig. 35. Construction of virtual blocks. (a)–(f) Steps for constructing a block; (g)–(k) steps for constructing all blocks.

can be generated by gluing neighbors B_2, B_3, B_4. When S_2 and S_3 are reprojected onto three views, $s_2 \in E(S_2|P), s_2 \notin E(O|P)$ and $s_3 \in E(S_3|P), s_3 \notin E(O|P)$ are generated. Therefore B_3 and B_4 cannot be valid blocks and only B_2 is accepted.

When two valid blocks are assembled, a face shared by them is removed, and an edge shared by coplanar faces of two blocks is also eliminated as shown in Fig. 37(b). Since a cutting edge is inserted when two virtual faces that can be faces of an object are segmented, two of four segmented faces must be ghost faces. Therefore, when a block is rejected and a face of the block is segmented by a cutting edge, the blocks that have the faces coplanar to the face are also rejected.

Broken lines in the views are used to get the accurate solution in case that a set of views can be interpreted as several difference objects. Blocks containing edges projected as broken lines are regarded as being located at the back of the blocks that have edges projected as solid lines. In Fig. 38, three orthographic views can be interpreted as two objects. However, a solution in Fig. 38(b) is selected as final solid using broken line information.

When the final model is selected, ghost elements are removed from sets of candidate vertices, candidate edges, and virtual faces. Final solid model for an object is completed by means of the sets of vertices, edges, and faces. Figure 39 shows some examples of three orthographic views and solid models reconstructed from them.

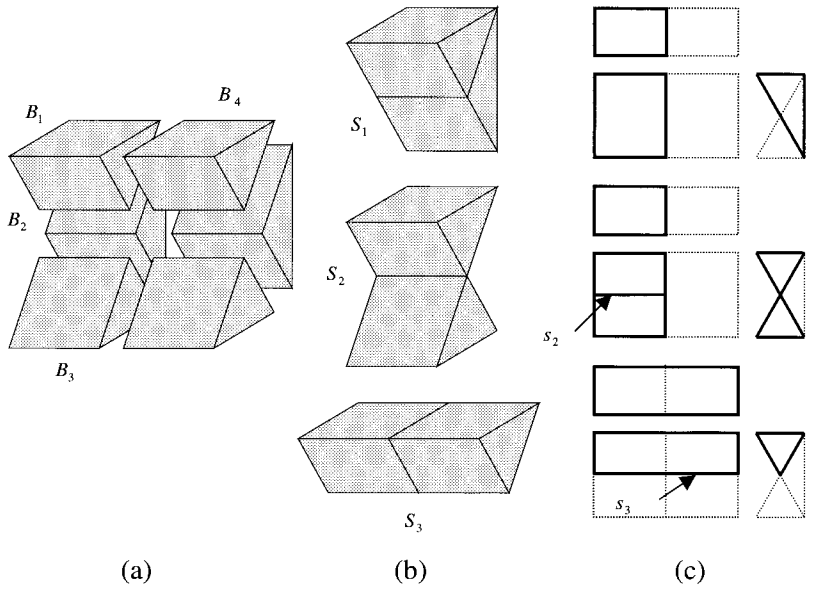


Fig. 36. Determination of state of a c-block. (a) Virtual blocks; (b) intermediate solids; (c) reprojection.

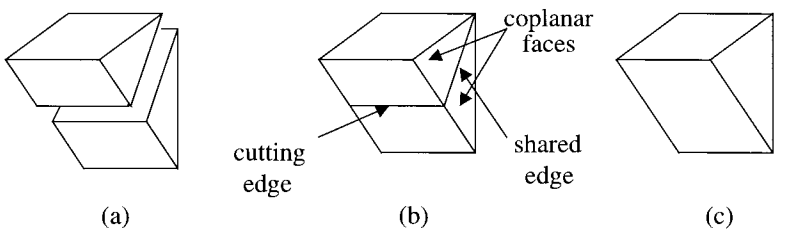


Fig. 37. Assembling two blocks: (a) two valid blocks; (b) eliminate an edge shared by two coplanar faces; (c) a merged block.

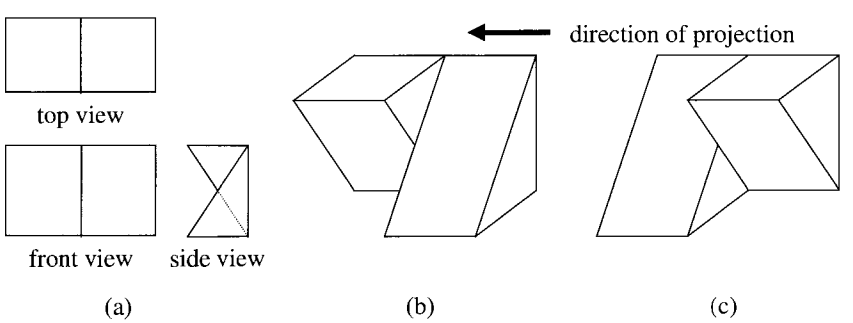
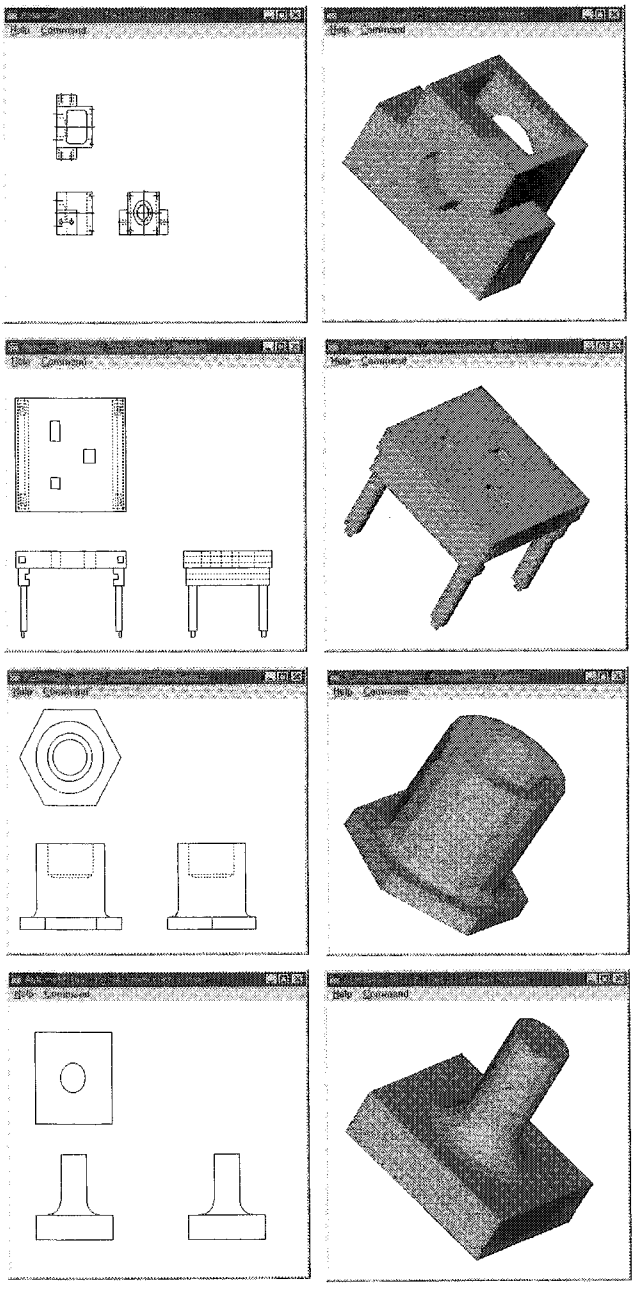
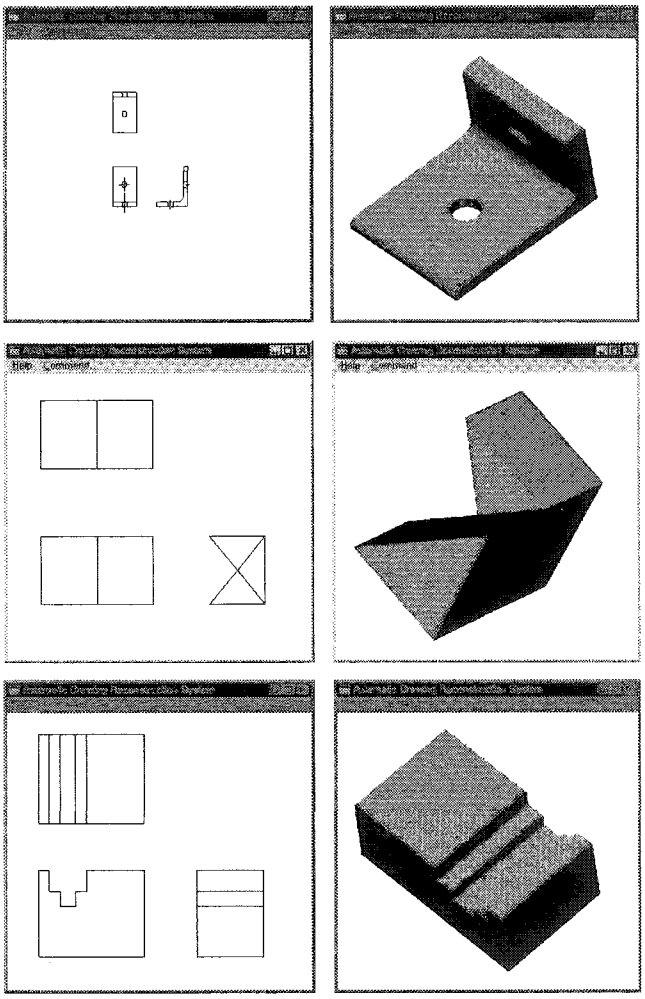


Fig. 38. Determination of final model using broken line information. (a) Three views; (b) solution I; (c) solution II.



(a) (b)

Fig. 39. Examples of three orthographic views and solid models reconstructed from them. (a) Three orthographic views; (b) solid model.



(a) (b)

Fig. 39.

References

1. S. C. Agarwal and W. N. Waggenspak, Decomposition method for extracting face topologies from wireframe models, *Computer-Aided Design* **24**, 3 (1992) 123–140.
2. B. Aldefeld, On automatic recognition of 3D structures from 2D representations, *Computer-Aided Design* **15**, 2 (1983) 59–64.
3. B. Aldefeld and H. Richter, Semiautomatic three-dimensional interpretation of line drawings, *Computers and Graphics* **8**, 4 (1984) 371–380.
4. D. D. Bedworth, M. R. Henderson and P. M. Wolfe, *Computer-Integrated Design and Manufacturing* (McGraw-Hill International Inc., U.S.A., 1991) pp. 232–291.
5. I. Carlbom and J. Paciorek, Planar geometric projections and viewing transformations, *Computing Survey* **10**, 4 (1978) 465–502.

6. T. Chang and R. A. Wysk, *An Introduction to Automated Process Planning Systems* (Prentice-Hall, Englewood Cliff, N.J., 1985).
7. H. Chen and W. Wang, The feudal priority algorithm on hidden-surface removal, *ACM Computer Graphics Proceedings*, 1996, 55–64.
8. D. N. Chorafas and S. J. Legg, *The Engineering Database* (Butterworth, 1988).
9. D. Dori and K. Tombre, From engineering drawings to 3D CAD models, *Computer-Aided Design* **28**, 4 (1995) 243–254, 1995.
10. D. Dutta and Y. L. Srinivas, Reconstruction of curved solid from two polygonal orthographic views, *Computer-Aided Design* **24**, 3 (1992) 149–159.
11. J. Foley, A. van Dam, S. Feiner and J. Hughes, *Computer Graphics Principles and Practice* (Addison-Wesley, U.S.A., 1990) pp. 548–557.
12. J. Foley, A. van Dam, S. Feiner, J. Hughes and R. Phillips, *Introduction To Computer Graphics* (Addison-Wesley, U.S.A., 1993) pp. 377–378.
13. A. L. Poston, C. L. Smith, and T. Au, *Fundamentals of Computer-Integrated Manufacturing* (Prentice-Hall International Inc., U.S.A., 1991) pp. 174–183.
14. H. Fuchs, Z. M. Kedem and B. F. Nayler, On visible surface generation by a priori tree structure, *Proceedings of ACM SIGGRAPH80*, 1980, 124–133.
15. J. Grudin, CSCW: History and focus, *IEEE Computer* **27**, 5 (1994) 19–26.
16. K. Gu, Z. Tang and J. Sun, Reconstruction of 3D solid objects from orthographic projections, *Computer Graphics Forum* **10**, 5 (1986) 317–324.
17. B. Ho, Inputting constructive solid geometry representations directly from 2D orthographic engineering drawings, *Computer Aided Design* **18**, 3 (1986) 147–155.
18. M. Idesawa, A system to generate a solid figure from three views, *Bulletin of JSME* **16** (1973) 216–255.
19. M. Idesawa, T. Soma, E. Goto and S. Shibata, Automatic input of line drawing and generation of solid figure from three-view data, *Proceedings of the International Joint Computer Symposium*, 1975, 304–311.
20. C. Jackins and S. L. Tanimoto, Oct-trees and their use in representing three dimensonal objects, *Computer Graphics and Image Processing* **14**, 3 (1980) 249–270.
21. R. A. Jarvis, A laser time-of-flight range scanner for robotic vision, *IEEE Transactions Pattern Analysis and Machine Intelligence* **5**, 5 (1983) 505–512.
22. M. Karima, K. S. Sadhal and T. O. McNeil, From paper drawings to computer-aided design, *Computer Graphics and Application* **5**, 2 (1985) 27–39.
23. C. Kim, M. Inoue and S. Nishihara, Heuristic understanding of three orthographic views, *Journal of Information Processing* **15**, 4 (1992) 510–518.
24. D. Lamb and A. Bandopadhay, Interpreting a 3D object from a rough 2d line drawing, *Proceedings of Visualization 90*, 1990, 59–66.
25. S. J. Lee, R. M. Haralick and M. C. Zhang, Understanding objects with curved surfaces from a single perspective view of boundary, *Artificial Intelligence* **26** (1985) 145–169.
26. J. Malik, Interpreting line drawing of curved objects, *International Journal of Computer Vision* **1** (1987) 73–103.
27. M. Mantyla, *An Introduction to Solid Modeling* (Computer Science Press. Inc., U.S.A., 1988) pp. 3–11.
28. G. Markowsky and M. A. Wesley, Fleshing out wireframe, *IBM Journal of Research and Development* **24**, 5 (1980) 582–597.
29. J. R. Miller, Architectural issues in solid modelers, *Computer Graphics and Application* **9**, 5 (1989) 72–87.
30. V. Nagasamy and N. A. Langrana, Engineering drawing processing and vectorization system, *Computer Vision, Graphics and Image Processing* **49**, 3 (1990) 379–397.

31. I. V. Nagendra and U. G. Gujar, 3-D objects from 2-D orthographic views — A survey, *Computers and Graphics* **12**, 1 (1988) 111–114.
32. S. Nishihara and K. Ikeda, Interpreting engineering drawings of polyhedons, *Proceedings of 9th International Conference on Pattern Recognition*, 1988, 869–871.
33. M. Noll, A computer technique for displaying N-dimensional hyperobject, *Communications ACM* **10**, 8 (1967) 469–473.
34. J. D. Palmer and N. A. Fields, Computer-supported cooperative work, *IEEE Computer* **27**, 5 (1994) 15–17.
35. A. A. G. Requicha, Representation for rigid solid: Theory, methods, and systems, *Computing Surveys* **12**, 4 (1980) 437–464.
36. A. A. G. Requicha and H. B. Voelcher, Solid modeling: A historical summary and comtemporary assessment, *IEEE Computer Graphics and Applications* **2**, 2 (1982) 9–24.
37. A. A. G. Requicha and H. B. Voelcher, Solid modeling: Current status and research directions, *IEEE Computer Graphics and Applications*, **3** (1983).
38. J. Rossignac and H. Voelcker, Active zones in CSG for accelerating boundary evaluation, redundancy elimination, interference detection, and shading algorithms, *ACM Transactions on Graphics* **8**, 1 (1989) 51–87.
39. H. Sakurai and D. C. Gossard, Solid model input through orthographic views, *Computer Graphics* **17**, 3 (1983) 243–252.
40. B. S. Shin, K. S. Oh, P. M. Ku and Y. G. Shin, An efficient algorithm for three-dimensional model reconstruction, *2nd Asian Conference on Computer Vision*, Singapore **1** (1995) 383–387.
41. B. S. Shin and Y. G. Shin, Fast 3D solid model reconstruction from orthographic views, *Computer-Aided Design* **30**, 1 (1998) 63–76.
42. K. Sugihara, *Machine Interpretation of Line Drawing* (The MIT Press, 1986).
43. W. C. Thibault and B. F. Naylor, Set operation on polyhedra using binary space partitioning tree, *ACM SIGGRAPH 87*, 1987, 153–162.
44. D. Vossler, Sweep-to-CSG conversion using pattern recognition techniques, *Computer Graphics and Application* **5**, 8 (1985) 61–68.
45. W. Wang and G. G. Grinstein, A polyhedral object's CSG-rep reconstruction from a single 2D line drawing, *Proceedings of 1898 SPIE Intelligent Robots and Computer Vision: Algorithms and Techniques*, **1192**, 1989, 230–238.
46. W. Wang and G. G. Grinstein, A survey of 3D solid reconstruction from 2D projections line drawings, *Computer Graphics Forum* **12**, 2 (1993) 137–158.
47. A. Watt, *3D Computer Graphics*, 2nd Edition (Addison-Wesley, U.S.A., 1993) pp. 33–35.
48. K. Weiler, Edge-based data structures for solid modeling in curved-surface environment, *IEEE Computer Graphics and Application* **5**, 1 (1985) 21–40.
49. M. A. Wesley and G. Markowsky, Fleshing out projections, *IBM Journal of Research and Development* **25**, 6 (1981) 934–954.
50. D. J. Wilde, The geometry of spatial visualization: Two problems, *Proceedings of the 8th IFTOM World Congress*, Prague, 1991.
51. T. C. Woo, A combinatorial analysis of boundary data structure schemata, *IEEE Computer Graphics and Applications* **5**, 2 (1985) 18–27.
52. Qing-Wen Yan, C. L. Philip Chen and Z. Tang, Efficient algorithm for the reconstruction of 3D objects from orthographic projections, *Computer-Aided Design* **26**, 9 (1994) 699–717.

CHAPTER 6

COMPUTER TECHNIQUES AND APPLICATIONS FOR REAL-TIME EMBEDDED CONTROL IN MECHATRONIC SYSTEMS

MATJAŽ COLNARIČ
University of Maribor,
Faculty of Electrical Engineering and Computer Science,
SI-2000 Maribor, Slovenia.
E-mail: colnaric@uni-mb.si

WOLFGANG A. HALANG
University of Hagen,
Faculty of Electrical Engineering,
D-58084 Hagen, Germany.
E-mail: wolfgang.halang@fernuni-hagen.de

Computer Aided and Integrated Manufacturing Systems rely heavily on the techniques of Mechatronics System. These systems consist of three main physical components — viz., mechanical, electrical and computer subsystems. The latter are often called embedded computer control systems. By definition they operate in the hard real-time domain, i.e., they must be permanently ready to respond to the requests from the operating environment in pre-determined time frames; hence their other names: responsive or reactive systems. In this chapter, first the basic properties of real-time systems will be examined. Then, an outline of an architecture for a consistent embedded control will be given. Further, some domain explicitly involved in embedded systems will be dealt with like hardware architectures operating system issues, languages, computers and worst case executive time analysis.

Keywords: Mechatronics; real time systems; embedded computer control systems.

1. Introduction

Mechatronic systems are increasingly being used for a large variety of purposes and, accordingly, draw great attention of engineering science. These "systems for perception and action" consist of three main physical components, viz., mechanical, electrical and computer subsystems. The latter are often called embedded computer control systems. By definition they operate in the hard real-time domain, i.e., they must be permanently ready to respond to the requests from the environment in predetermined time frames; hence their other names: responsive or reactive systems.

Some twenty years ago the discipline of real-time systems began to increasingly attract more and more interest. All domains involved received considerable attention from computer scientists all over the world, and from most major fields of computer science and engineering.

Apparently the most characteristic misconception on the domain of hard real-time systems[26] was that real-time computing was often considered as fast computing. It is obvious that computer speed itself cannot guarantee that specified timing requirements will be met. Thus, ultimate guidelines for their design have changed from fast to fast enough: it is of utmost importance that systems meet their pre-set deadlines.

To be able to achieve this, determinism and predictability of the temporal behavior of computing processes is necessary: these properties essentially imply all other requirements.

Unfortunately, although there is no dispute on that among scientists, the state-of-the-art in practical applications is still far away from meeting (or even proving to meet) these requirements: in general, program execution and system response times are usually unpredictable and not consistently analyzed. Hence, flawless performance of control systems cannot be guaranteed.

The reason for this is that common hardware and software as employed in control applications are optimized for average performance, whereas in order to guarantee that deadlines are met, it is necessary to consider the worst-case behavior. In execution time analysis, necessarily there is a certain amount of pessimism involved, considerably diminishing the estimated average performance.

In order to avoid having to use more powerful hardware platforms because of these pessimistic estimations, designers tend to develop and test control applications in the usual ways of non-real-time computing, at risk of not meeting deadlines. The consequences, however, could be severe in certain cases, like massive material losses or even endangerment of human lifes. This situation must be overcome soon. The gap between academic research and practical implementations must be narrowed to provide for much more consistent and safer computer control systems.

Following the ultimate objective of mechatronic design, we carried out a research project systematically addressing all crucial layers of hardware and software components in a holistic manner to provide for practically usable hard real-time control system design techniques. Our objective was not to generate novel approaches or methods, but to select and consistently use concepts from our and other groups' previous research.

In this sense, we used off-the-shelf processors and other hardware components. They were carefully selected according to their temporal determinism. To achieve temporal predictability and to optimize performance, hardware and software were co-designed in the large and in the small scale, so that their features mutually support each other.

In this chapter, first the basic properties of real-time systems will be briefly revisited. Then, an outline of the architecture for a consistent embedded control

system will be given. Furthermore, some domains explicitly involved in embedded systems design will be dealt with, like hardware architectures, operating system issues, languages, compilers, and worst case execution time analysis.

A special section is dedicated to safety-critical systems, bearing in mind that most control systems are safety-critical to a certain extent.

Although interesting and relevant to some more sophisticated computer control systems, higher-level issues such as advanced software engineering topics, real-time databases or artificial intelligence will not be covered here.

2. Basic Properties of Embedded Real-Time Systems

Real-time operation is the operating mode of a computer system in which programs for the processing of data arriving from the outside are permanently ready, so that their results will be available within predetermined periods of time. The arrival times of the data can be randomly distributed, or already determined *a priori* depending on different applications.[5] Although functionally correct, results produced beyond pre-determined time frames are wrong.

For computer control systems, one distinguishes between hard and soft real-time requirements. Whereas in cases of soft real-time, the penalty for missing a deadline increases with time; in hard real-time cases, late results are useless with any consequences that may imply. The general optimization criterion in soft real-time environments is average performance; while for hard real-time applications, it is necessary to consider worst case behavior. Most systems, however, are hybrids: hard and soft real-time tasks usually co-exist in applications, where hard real-time tasks execute with higher priority as a rule, and the soft real-time tasks run within the performance reserve of the hard real-time ones.

To guarantee specified temporal behavior of control systems, predictability of temporal behavior was set as the ultimate objective. Being able to assure that a process will be serviced within a predefined time frame is of utmost importance. In multiprogramming environments, this condition can be expressed as schedulability: the ability to find, *a priori*, a schedule such that each task will meet its deadline.[29]

For schedulability analysis, the execution times of the hard real-time tasks, whose deadlines must be met in order to preserve the integrity of a system, must be known in advance. These, however, can only be determined if the system functions in a deterministic and predictable way in the time domain.

There are two different comprehensions of the predictability of temporal behavior: layer-by-layer (microscopic) and top-layer (macroscopic) predictability.[27]

Layer-by-layer predictability requires each of the layers in a real-time computing system to behave deterministically and predictably, thus providing the necessary basis for the next layer. This approach is suitable for embedded control systems with moderate complexity. It was shown that their temporal behavior can be guaranteed *a priori*.[7]

Top-layer predictability only considers the behavior on the highest (application) layer and provides handlers to deal with missing deadlines. This concept is suitable for sophisticated and complex applications where it is impossible to deal with time in detail. However, in embedded systems, especially those with high integrity requirements for safety-critical applications, it is necessary to strive for microscopic predictability.

Beside timeliness and predictability, an important property of control systems is dependability. The hardware part of these systems has already reached a high degree of dependability; methods and techniques for their verification are well established. Also, some extremely dependable hardware platforms have been developed, e.g., the VIPER-1A[15] processor.

On the other hand, the dependability of software is still lagging far behind. The reason for this is its inherent complexity. Unfortunately, software engineering has not sufficiently progressed in this specific field yet. For the time being, the most important principle to be followed when designing systems with severe dependability requirements is simplicity.

Simple solutions, however, are the most difficult ones; they require high innovation and complete intellectual penetration of issues:

Progress is the road from the primitive via the complicated to the simple.

Easy understandability is the most important precondition to ensure the correctness of a system.

3. Architecture of a Consistent Real-Time Control System

As it is common in engineering, there are always many possible system designs fulfilling a given set of demands—provided the problem is solvable with available technology. To derive an architecture appropriate for real-time control systems, we start off by considering an analogy from another field, where systems coping with real-time conditions have long been developed and used.

The example system comprises a manager and his or her secretary. The duties of the secretary are the reception of mail and telephone calls, the elimination of unimportant chores, and the minimization of interruptions to the manager's work by visitors and callers. Furthermore, the secretary schedules the manager's work by arranging the files in the sequence in which they are to be treated and through the administration of his or her meeting appointments. Thus, the manager's work becomes less hectic—i.e., the work's "real-time conditions" are eased—and more productive, because he or she can perform the tasks with less frequent interruptions in a more sequential and organized manner.

By taking pattern from this and related models, we now define the overall structure of a computer designed to meet the requirements of real-time operation.

The asymmetrical concept is depicted in Fig. 1. The system consists of two dissimilar processors, a task processor and an operating system kernel processor. The task processor is a classical processor as commonly used in process control, which is

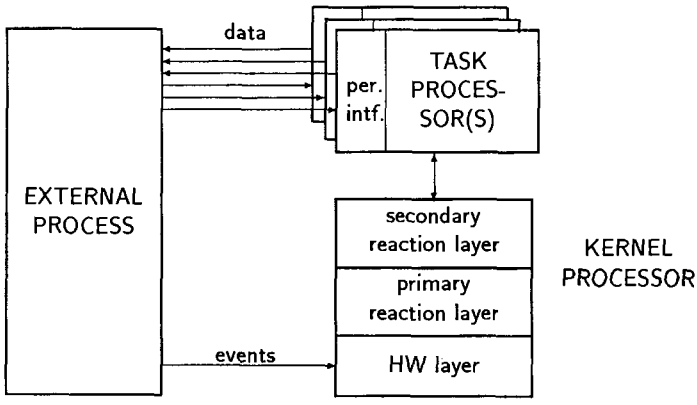


Fig. 1. The basic architectural concept

executing user application tasks. It acquires data from the controlled environment, and controls it by the control output generated. It also executes those operating system tasks that interface the user tasks. Being the creative component, the task processor corresponds to the manager in the above analogy.

The operating system kernel routines run on the second dedicated processor which is clearly and physically separated from the outer layer tasks. This co-processor houses the system functions, event, time and task management, communication and synchronization. Although important and actually controlling the operation of task processor, these functions are routine and would impose unnecessary burden to the latter. Thus, the kernel processor corresponds to the manager's secretary.

This concept was elaborated in detail in Refs. 2 and 7. If necessary, it can easily be extended to multiple task processors, each executing its own task set, being controlled by a single operating system kernel co-processor.

3.1. *Implications of employing earliest deadline first scheduling*

The fundamental requirement expected to be fulfilled by a process control system employed in a hard real-time environment is to carry out all tasks within predefined time frames (assuming this is actually possible). Algorithms generating appropriate schedules for all possible task sets, such that each task would conclude its execution before its given due date (deadline) are referred to as feasible. A number of such algorithms have been identified in the literature; for example, Ref. 7.

On the other hand, neither of the two scheduling policies supported most frequently in commercially available programming languages and real-time operating systems, namely the first-come-first-served and the fixed priority algorithm, is feasible.

It is well established that for scheduling independent tasks on a single processor, the earliest deadline first algorithm is feasible.[10,12,20,25] According to it, among all tasks waiting for execution, the one with the closest deadline will be scheduled.

When applied to multiprocessors, however, the earliest deadline first scheme becomes unfeasible. An extension of the policy that re-establishes feasibility on homogeneous multiprocessors is the throw-forward algorithm.[11,12,14] However, the latter leads to more pre-emptions and is more complex.

Another algorithm which is feasible for both single and multiprocessor systems is the least laxity first algorithm.[10,11,21] Unfortunately, it is only of theoretical interest, as it is pre-emptive and requires processor sharing when several tasks have the same laxity. That, in turn, calls for counter-productive context switching overhead.

In contrast to least laxity first, the earliest deadline first algorithm does not require context switches, unless a new task with an earlier deadline arrives or an executing task terminates. In fact, if the number of pre-emptions enforced by a scheduling procedure is considered as selection criterion, the earliest deadline first algorithm is optimal.[10] Even when tasks arrive dynamically, this policy maintains its properties and then generates optimal pre-emptive schedules.[17]

The above discussion suggests structuring real-time computer systems as single processors. The idea can be extended to distributed systems by structuring them as sets of interconnected uniprocessors, each dedicated to controlling a part of an external environment. The earliest deadline first scheduling algorithm is to be applied, independent of considerations of the overall system load, on each node in a distributed system. The implementation of this scheme is facilitated by the fact that industrial process control systems are already typically designed in the form of co-operating, possibly heterogeneous, single processor systems, even though the processors' operating systems do not schedule by deadlines yet.

This scheduling strategy establishes the direction in which the architecture should be developed. The objective ought to be to maintain, as much as possible, a strictly sequential execution of task sets. The processor(s) need(s) to be relieved of frequent interruptions caused by external and internal events in the sequential program flow. These interruptions are counter-productive in the sense that they seldom result in an immediate (re-)activation of a corresponding task.

To summarize, earliest deadline first task scheduling on a single processor system has the following advantages:

- scheduling on the basis of task deadlines is problem oriented;
- It allows the formulation of tasks and the extension and modification of existing software without the knowledge of the global task system;
- All tasks can be treated by a common scheduling strategy, regardless of whether they are sporadically or periodically activated, or have any precedence relations;
- Feasibility;
- Upon a dynamic arrival of a ready task, the task's response time can be guaranteed (or a future overload can be detected);

- The cost of running the algorithm is almost negligible (its complexity is linear in the number of tasks in the ready queue);
- Ease of implementation;
- The cost of checking for feasible schedulability of a task set is almost negligible (again, linear in the number of tasks in the ready queue), and the check itself is trivial, i.e., the operating system is enabled to supervise the observance of the fundamental timeliness condition;
- Facilitation of early overload detection and handling by dynamic load adaptation, thus allowing system performance to degrade gracefully;
- Achieving the minimum number of task pre-emptions required to execute a feasible schedule;
- Achieving maximum processor utilization while maintaining feasible schedulability of a task set;
- It is essentially non-pre-emptive, i.e., task pre-emptions may only be caused when dormant tasks are activated or suspended ones are resumed;
- The sequence of task executions is determined at the instants of task (re-)activations and remains constant afterwards, i.e., when a new task turns ready, the order among the others remains constant;
- The order of task processing is essentially sequential;
- Resource access conflicts and deadlocks are inherently prevented;
- Unproductive overhead is inherently minimized;
- The priority inversion problem, which received much attention in the literature, does not arise at all; and
- Pre-emptable and (partially) non-pre-emptable tasks can be scheduled in a common way.

3.2. *Architectural concept*

The asymmetric multiprocessor architecture as shown in Fig. 1 can be employed as either a stand-alone device or a node in a distributed system. Each node consists of one co-processor and one or more general task processors. It is assumed that the latter cannot be interchanged, because process control computers are usually connected to a specific part of a technical process (an external environment). Hence, the functions of the general processors are determined by their respective subprocesses and cannot be migrated. Thus, the tasks on each general processor can be scheduled independently of the tasks on other general processors. The data transmission lines to and from all input/output devices in the system are connected to the task (i.e., general) processors, whereas all interrupt lines are wired to the co-processor.

Real-time data processing systems are expected to recognize and react to occurring events as soon as possible, or even, in the ideal case, instantaneously. In conventional hardware, prompt recognition and reaction are accomplished by interrupting the running task, determining the source of the event, and switching to an appropriate interrupt handling task. The running task is thus pre-empted, even though

Table 1. Function assignment in the co-processor.

1. Hardware layer
Accurate real time management based on a high resolution clock.
Exact timing of operations (optional).
Separate programmable interrupt generator for software simulation.
Event representation by storage element, latch for time of occurrence, and counter of over-run arrivals of events.
Synchroniser representation.
Shared variable representation.
2. Primary reaction layer
Recognition of events, i.e., interrupts, signals, time events, status transfers of synchronizers, and value changes of shared variables.
Commencement of secondary reactions.
Recording of events for error tracking.
Management of time schedules and critical instants.
3. Secondary reaction layer
Deadline driven processor scheduling with overload handling.
Task oriented hierarchical storage management.
Execution of (secondary) event reactions, esp. tasking operations.
Synchronizer management.
Shared variable management.
Acceptance of requests.
Initiation of processor activities.

it is likely that the interrupt does not relate to it. Furthermore, the task which is handling the newly arrived interrupt will not necessarily be executed before the pre-empted one, once the interrupt has been identified and acknowledged.

Owing to this inherent independence, the possibility to apply parallel processing—task execution and event recognition and administration—is given here. In order to preserve data integrity in the conventional architecture, tasks may inhibit to be interrupted during the execution of critical regions. Hence, there may be a considerable delay between the occurrence of an event and its recognition, for which no upper bound can be guaranteed. This situation is further impaired when several events occur (almost) simultaneously, thus resulting in both continuous pre-emptions of their corresponding handler tasks, and delay of some of the lower priority reactions.

To address these problems, our co-processor should provide a separate, independently working event recognition mechanism capable of commencing a primary reaction to an event within a predefined, guaranteed and short time frame. In order to provide this capability, the co-processor is structured into three layers, whose functions are compiled in Table 1. In short, the co-processor unit executes the kernel of an operating system developed according to the needs of real-time application software.

4. Hardware Architectures

In contemporary state-of-the-art control computers, conventional microprocessors are used, which are optimized for high throughput in the average case. This,

however, is in contradiction with the hard real-time optimization criterion, viz., worst-case performance.

Processor utilization itself is a thinking category of the 1940's, which became obsolete by the achievements in solid-state technology and, consequently, the high availability and low prices of hardware components. Sub-optimal utilization is a very cheap price to be paid for improvement in simplicity and dependability.

4.1. *Undesirable properties of conventional architectures*

Undesirable properties of conventional architectures are those which make a system's temporal behavior (1) unpredictable or/and (2) difficult to estimate. A common counter-productive characteristic of their design is excessive complexity, which makes their verification difficult or impossible.

The increase in microprocessor performance through the years was mainly influenced by two factors, viz., technological advances and new ideas employed in architectural design. These new ideas are, e.g., the RISC philosophy, parallel processing (pipelining), caching, and others. While RISC ideas are very suitable also for architectures employed in hard real-time systems for the simplicity and verifiability achieved, the other two features (although always present also in RISC processors) present a hazard to determinism and predictability of their temporal behavior.

4.1.1. *Independent parallel operation of internal components*

In order to fully exploit inherent parallelism of operations, the internal units of processors were becoming more and more autonomous resulting in highly asynchronous operation. Due to this development the average performance of processors is increased. However, it is a complex task to analyze corresponding sequences of machine code instructions in order to determine their execution times. For the time being, verification of such complex processors is practically impossible.

The most common way to improve the throughput of a processor by exploiting instruction level parallelism is pipelining. Various techniques are used to cope with the well-known pipeline breaking problem due to the dependence of instruction flow control on the results generated by previous instructions. All of them, however, try to optimize average performance.

To predict execution times, complex machine code analyzers would be necessary, based on detailed information about instruction execution, which is often proprietary and, thus, inaccessible. Owing to its complexity, a processor with pipelined operation is also very difficult to verify.

4.1.2. *Caching*

Fetching data from off-processor sources represents a common bottle-neck, especially in the von Neumann processor concept. To avoid it, cache memories were introduced. Since instruction fetching from memory takes a considerable amount of

execution time, the latter thus depends highly on whether an instruction is found in a cache or not. To predict this, complex analyzers would be necessary, emulating the caching operation. Even the most sophisticated ones would fail in the case of multiprogramming, where caches are filled with new contents on every context switch.

Based on considerations similar to the ones on processor architecture, classical system architectures contain features to improve average performance. Again, these features may lead to undesirable consequences rendering the prediction of process execution times difficult or even impossible. Some of them are listed in the sequel.

4.1.3. *Direct memory access*

Since processors are ineffective in transferring larger blocks of data, direct memory access (DMA) techniques were designed. With respect to the delays caused by DMA transfers, there are two general modes of DMA operation, viz., the cycle stealing and the burst mode. A DMA controller operating in the cycle stealing mode is literally stealing bus cycles from the processor, while in the other mode the processor is stopped until the DMA transfer is completed. Although the processor has no control over its system during such a delay, this mode is more appropriate when predictability is the main issue. If block length and data transfer initiation instant are known at compile time, the delay can be calculated and considered in program execution time estimation. Furthermore, block transfer is also faster, because bus arbitration needs only to be done once.

4.1.4. *Data transfer protocols*

Data transfer via microcomputer busses also deserves some consideration. Synchronous data transfer protocols by definition ensure predictable data transfer times. Asynchronous ones are more flexible, however, their behavior is very difficult to control, especially in shared-bus systems. The best way to guarantee realistic data transfer times seems to be synchronous operation of all potential bus masters.

Similar conclusions hold for local area networks as well. For distributed control systems, appropriate network protocols with deterministic and predictable behavior must be chosen. The widely used CSMA/CD (Ethernet) protocol is inappropriate because of non-deterministic resolution of collisions. A frequent problem is also the complexity of protocols: the standard document DIN 19245 on the field bus Profibus, e.g., has 750 pages.

4.1.5. *Exception and interrupt handling*

The problems discussed in the previous paragraphs could either be prevented, or the measures potentially causing non-deterministic or unpredictable behavior could be renounced. However, in process control computers operating in dynamic mode,

i.e., immediately responding to the events in their environments, some kinds of exceptions and especially interrupts are unavoidable. Since these events occur asynchronously to program executions, the latter are necessarily delayed. This problem cannot be solved in classical single processor architectures.

4.2. *Prototype of an appropriate hardware platform*

Above it was considered as disadvantageous that in the single processor architecture the operating system is running on the same processor(s) as the application software. In response to any event occurring, the context is switched, system services are performed, and scheduling decisions are made. Although it is very likely that the same process will be resumed, temporal determinism of process execution is violated and a large amount of performance is wasted by superfluous overhead.

Following the general architectural outline as given in Sec. 3, employing a second, parallel processor to carry out the operating system services is suggested. Similar concepts were independently considered in a number of research projects and implemented in several successful prototypes (e.g., Refs. 4, 19, 24 and 28). Compared to Ref. 4, our approach is more complex and supports multiprocessing as well, and to a certain extent, distributed processing. However, it is still targeted at embedded applications and is thus, less sophisticated than Ref. 28. The latter supports local area networking and features flow control filters, reflective memories, and a VLSI implemented scheduling controller.

In Fig. 2 our experimental hardware platform is shown, consisting of task processors (TPs) and a kernel processor (KP), which are fully separated from each other. Loose coupling is achieved by serial links and a few additional signals. This promises easy local distribution of process control (task) processors, and migration of processing power close to where it is needed.

The **kernel processor** (KP) is responsible for all operating system services. It maintains the real-time clock, and observes and handles all events related to it, to the external signals and to the accesses to the common variables and synchronizers, each of these conditions invoking assigned tasks into the ready state. It performs earliest deadline first scheduling on them and offers any other necessary system services.

The external process is controlled by tasks running in the **task processors** (TP) without being interrupted by the operating system functions. A running task is only pre-empted if, after re-scheduling caused by newly invoked tasks as a consequence of events, it is absolutely necessary to assign the highest priority to it in order to ensure that all tasks will meet their deadlines.

Although the real-time clock is really necessary only in the kernel processor (and optionally in peripheral controllers to allow for exactly timed and jitter-free I/O operations as described in Ref. 8), it may be available as relative Julian time throughout the system: in any of the interested units it can be implemented in hardware or in software as a counter of standard time unit ticks (*RT clock* signal),

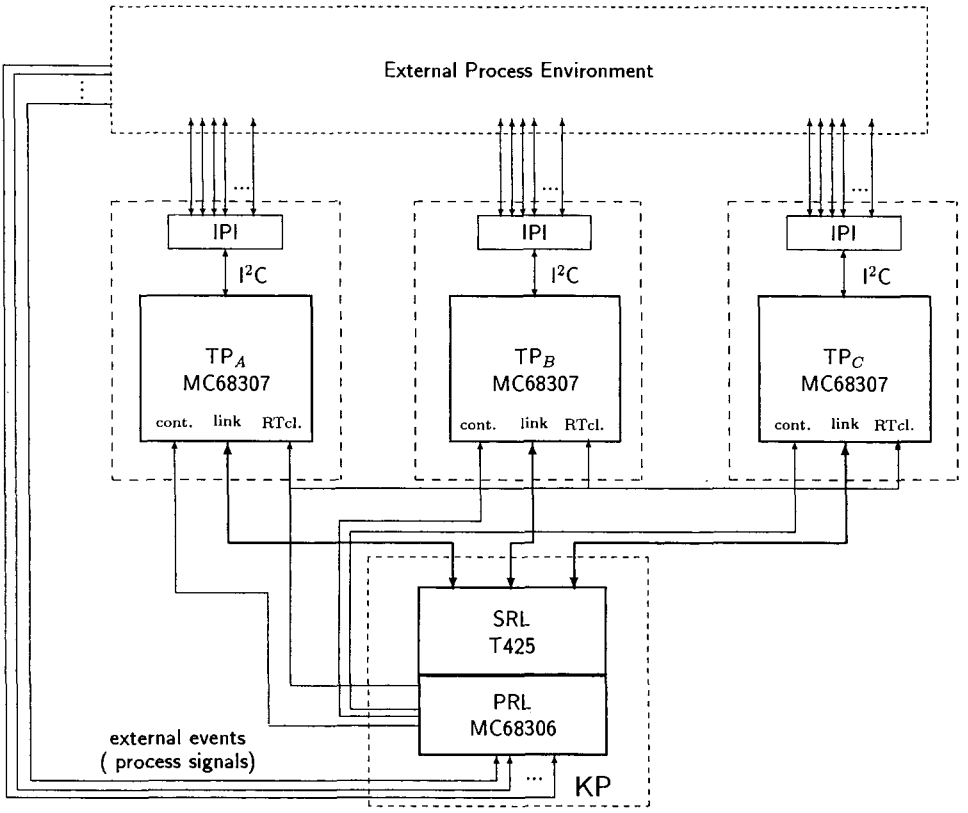


Fig. 2. General model of the hardware architecture.

generated by the kernel processor and beginning at system initialization. Absolute time can easily be calculated from the counter value and the absolute time recorded for the system initialization instant.

In process control it is often necessary to perform certain operations at exactly defined instants. As a rule, this cannot be achieved with high accuracy by means of scheduling tasks to be activated at those times because of operating system overhead and non-deterministic precedence relations holding between the active tasks at those very moments—particularly not in dynamic environments. This option is implemented by halting task processing in a certain state and re-activated by continuation signals generated by the kernel processors.

The presented concept prevents non-deterministic interruptions from the environment to reach the task processors by (1) careful avoidance of sources of unpredictable processor and system behavior; (2) by loose coupling of task processors; and (3) by synchronous operation of the kernel processor. The predictability of temporal system behavior achieved through this concept is expected to be sufficient to provide the necessary basis for the higher levels of system design.

In the following paragraphs the implementation of the hardware units will be detailed.

4.2.1. *Operating system kernel processor*

To support determinism and in order to achieve better performance, the operating system kernel processor is divided into two layers, a higher-level part called *secondary reaction layer (SRL)*, and a lower-level part or *primary reaction layer (PRL)* (see Fig. 3).

Since their functions are tightly coupled, the processors implementing the two layers communicate via a dual-port memory (*DPRAM*), where their interfaces reside in form of operating system data structures and the real-time clock storage element. Conflicts of simultaneous access are resolved by an arbiter built inside the dual-port memory. However, accesses are exclusively performed in carefully defined time slots; this way conflicts are avoided.

Differing slightly from the general model as shown in Fig. 1, the hardware layer is implemented inside the primary reaction layer microcontroller for practical reasons.

In short, the PRL is performing elementary hardware functions like external event recognition, time and time event management etc., while the SRL serves as

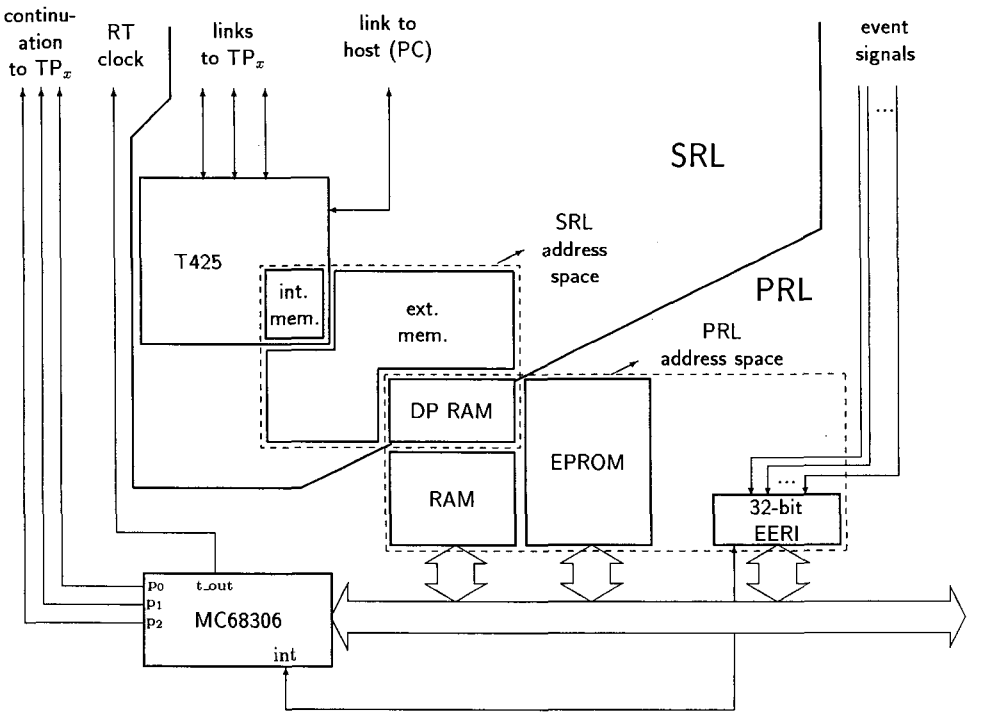


Fig. 3. Operating system kernel processor.

interface to the task processors, manages their operating system service requests and carries out the actual task scheduling on the basis of information on events provided by the PRL.

There are several reasons for structuring the operating system processor into two parts. An important one is the natural parallelism between the lower-level (signal sensing, administration of real-time clock, time schedules etc.) and the higher-level functions (scheduling, operating system routines); exploiting that, the performance of the kernel processor is improved and, hence, the response times are shortened. Further, the primary reaction layer is intended to be implemented in an FPGA in the forthcoming versions of the prototype. Finally, in our particular implementation, the transputer does not provide useful interrupt handling capabilities and its two level scheduling was found inappropriate for this purpose; thus, we decided to add the lower-level processor.

In the sequel, functions and implementation of the PRL and the SRL will be detailed.

Primary Reaction Layer is implemented using a Motorola MC68306 microcontroller. Its basic functions are:

- *Generation of the RT clock signal.* RT ticks are generated by the MC68306 built-in timer/counter. For every tick a preset value is loaded into the counter and decremented with every clock cycle; upon expiration, the *RT clock* tick is generated at the timer's external output and the internal interrupt is signalled.
- *Administration of the real-time clock in form of Julian time.* Time is maintained by software: on every *RT clock* tick the time is incremented. The 32 bits long word allows for relative times of 49 days with a resolution of 1 msec. During power-on, all real-time counters throughout the system are reset and the *RT clock* signal is disabled. In the initialization sequence the kernel processor acquires precise actual absolute time (either from a host computer or via some other means—e.g., the Global Positioning System), adds the necessary (constant and known) overhead until the counters are started, and distributes it among all interested parties. Then, the *RT clock* signal is enabled and all counters begin to count the same ticks. Hence, the relative times will be the same, and the constant offset to the absolute time known all over the system.
- *Administration of time events introduced by time-related schedules (either for tasking operations or for continuation of their execution).* SRL provides a vector of activities which are scheduled to be activated at certain times. PRL administers this vector, selects the closest time event and, upon its occurrence, provides information accordingly so that the associated actions can be taken over by SRL.
- *Reception and handling of signals from the process environment (and monitoring whether overruns occur).* Events from the environment are received by a specially designed external event recognition interface implemented by an FPGA. It provides 32 parallel digital I/O ports; active (negative) transition on each of these ports sets a bit in a register which is periodically polled; any bits set are cleared and

registered in the operating system data structures together with time stamps. If an event appears more than once during the same polling cycle, an overrun bit is set indicating that at least one event was not serviced.

- *Provision for exact timing and/or synchronous operation of task processors.* The already mentioned exact timing feature offers the possibility to schedule *a part of a task* to be executed at an exactly defined instant which, in principle, cannot be accomplished by usual task-level scheduling on a time event. To achieve this, a task running in a task processor can halt its own execution using an operating system call. In the call, a time schedule is associated with this action; when it is fulfilled, the *continuation* signal re-enables the program in the task processor by releasing the wait state. The continuation signals for all task processors are driven by the microcontroller's I/O lines and can be triggered individually or simultaneously by disjunctive combination of the I/O port data register.
- *Providing information on the actual events in the current system cycle to SRL* by putting them into the common data structures, to be available for actual scheduling of the associated tasks.

Secondary reaction layer communicates with the task processors, from which it receives calls for operating system service. These calls may require certain information about the state of the system or the tasks. Tasks may access the synchronizers and the common global variables residing in the SRL; a change in status of these structures triggers an event to which a task activation may be associated. Important types of service calls are tasking requests; a task running in a task processor may request time-related and/or event-related scheduling of different tasking operations. The conditions are passed to the PRL which triggers events upon their fulfillment.

When such events occur, the SRL invokes the tasks assigned to them, performs schedulability analysis, and actually places the ready tasks into the processor queue according to their deadlines.

The SRL is, from the viewpoint of hardware designers, a standard transputer system with an external bus access to the dual-port RAM. It has three bi-directional links to task processors and one to a host for purposes of monitoring and program down-loading. The latter can also be used as a link to a fourth task processor once an application is operational and its programs are placed in ROMs. These serial point-to-point communication paths are, apart from adequate processing power, the main motivation for using transputers.

Although not very practical, the system can also be scaled to slightly larger applications by adding an additional transputer to one link of the secondary reaction layer. Thus, six task processors can be supported, and a further two with each additional transputer. However, this would influence the response times of the system due to the additional load. To this point, a number of three or four task processors supported in the basic configuration was found to be most reasonable with regard to overall system performance.

4.2.2. *Application task processors*

The actual process control programs are executed in task processors. Only a small part of the operating system resides here performing communication with the SRL and context-switching upon the latter's request. In a task processor's memory the program code of each task mapped to it is kept. Also, parts of these tasks' control blocks are residing there, holding the contexts of eventually pre-empted tasks.

Basically, a task processor can be implemented using a wide variety of processing devices, such as microcontrollers, digital signal processors, or automata based on FPGAs or special-purpose VLSI components. Important selection criteria are the envisaged purpose in the application domain, and determinism and predictability of the temporal program execution behavior. To support run-time analysis for such a diversity of processors, we introduce a configurable compiler with a built-in run-time estimator using different methods and techniques, which will briefly be outlined in Sec. 6.3.

In Fig. 4, the task processor architecture is shown as implemented in our prototype with the Motorola MC68307 microcontroller. The temporal execution behavior of the MC68000 microprocessor kernel is not excessively complicated; analyses have shown that it is predictable with ease.

Every message received on the transputer link interrupts task program execution. It can either be a call for pre-emption of the running task, or a response to an operating system service request. In both cases the delay in task execution caused can be considered in the context-switch time, or in the operating system response time, respectively.

The microcontroller's internal timer can be used as a counter of *RT clock* ticks. Although conceptually the task processor does not need a real-time clock, its presence may save service calls to the kernel processor.

Below, some features of the task processor are presented.

• *Intelligent Process Interfaces* (IPIs) serve as interfaces between task processors and controlled environments. They are based on microcontroller or FPGA-supported peripheral devices connected to task processors via I^2C serial buses. Their services are available by calling pre-defined peripheral device drivers and providing parameters and data. IPIs provide higher-level peripheral interface services. By moving intelligence towards peripheral devices, task processors are relieved from that work, peripheral services are much more flexible and fault-tolerant, and system safety is enhanced.

Optionally, in simple implementations, instead of peripheral interfaces on an I^2C bus, peripheral devices may be directly implemented on task processor modules. For this purpose the MC68307 microcontroller's remaining peripheral interfaces may be used. Then, alternative peripheral device drivers must be prepared. On the other hand, in more demanding environments, peripheral interfaces may be linked to application task processors by sensor/actuator buses or, generally, field buses. Under the aspect of real-time capabilities, viz., bounded end-to-end communication

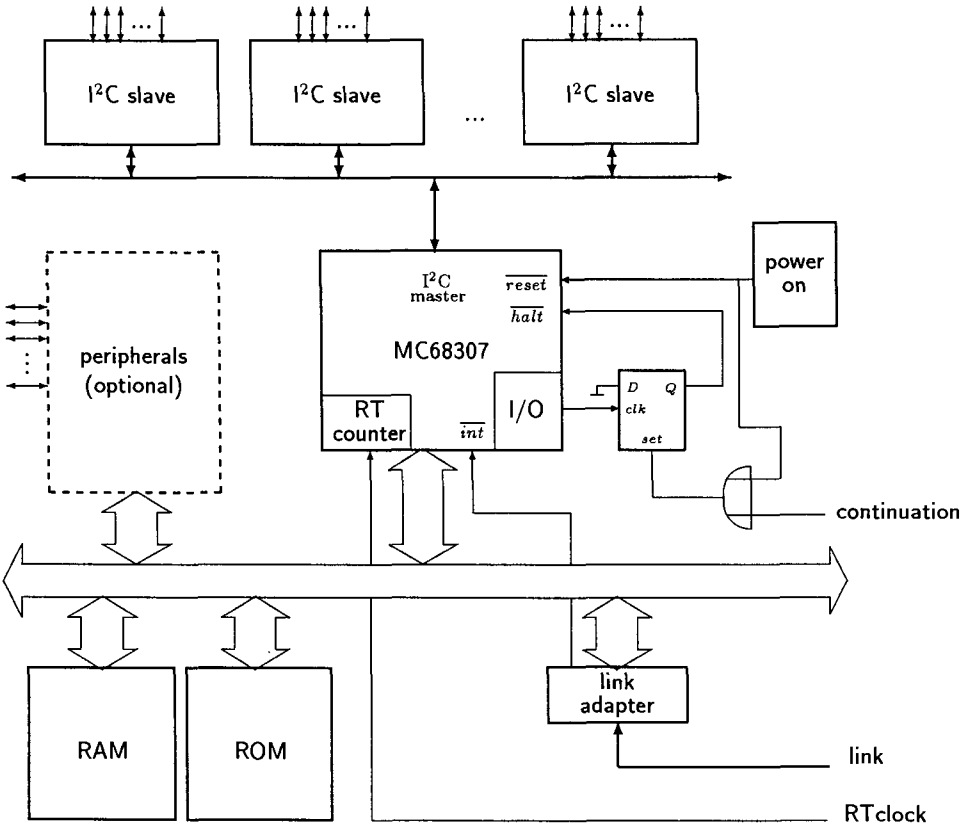


Fig. 4. Application task processor.

delay and deterministically guaranteed access time to the transmission medium, the best performance among field buses is then offered by InterBus-S.[1]

Peripheral devices must be carefully selected and implemented in such a way that their timing behavior is deterministic and can be predicted. For instance, analogue-to-digital conversion can take different time in certain implementations depending on the input level. A consistent implementation of such a peripheral device is to permanently sample and convert the analogue value; upon request the value acquired last is always returned without any delay. Assuming that the sampling interval is sufficiently small, the result is acceptable.

• *Exact timing of operations.* This feature already mentioned in the discussion of the kernel processor requires a counter-part in the task processors. After a task, reaching a point from where continuation at a certain time instant is requested, has issued the corresponding service call to the kernel processor, it enters the wait state sending a *wait* signal to a simple circuit triggered by an I/O line. When the instant for resuming the task processor's operation and the execution of the task is reached

(which is indicated by a time event), the PRL generates a *continuation* signal which resets the *wait* signal to the inactive state thus resuming operation.

While providing the most precise timing of process execution, e.g., for purposes of temporal synchronization, this implementation seems to have the drawback of halting the processor, preventing other ready tasks from executing during the idle time until the moment of continuation. In practice, however, there is no blocking of other tasks, since the waiting times are kept short and are considered in the estimation of the corresponding tasks' run-times. As an alternative to this solution, and to prevent idle waiting, an exact timing feature is also implemented on the level of intelligent peripheral interfaces. Along with an I/O command, the time when the operation is to be performed is transmitted to an IPI, which has access to the real-time clock. The calling task is suspended, waiting for the requested instant when the data are sent out or, respectively, latched into a buffer, ready to be read by the task processor. Which method is utilized depends on whether only peripheral operation or task execution as a whole is requested to be timed precisely.

With exact timing of input/output operations as introduced in[8] application designers can cope with the jitter problem: with a slight remaining jitter only depending on the real-time clock, they can precisely prescribe when certain sensors are read or actuators set in closed regulatory loops.

• *Distributed processing.* Process control and embedded applications are often implemented as distributed control systems in rather natural ways. Our architecture matches this approach on two levels, viz., on the level of task processors and on the level of intelligent peripheral interfaces.

Task processors are connected to the kernel processor via transputer links which are limited in length; however, it is possible to implement them in fibre optics, thus allowing distribution over longer distances. The serial bus connecting peripheral interfaces with task processors works with lower transfer rates and is, thus, less problematic; commercial drivers allow transfer over sufficiently long distances. Since in both communication protocols only master-slave and single-master options are implemented, overhead is extremely low in data transfer.

5. Operating System Support and Tasking

According to their nature, scale, purpose, implementation etc., in embedded real-time control systems different options are realized with respect to system software functions. In some larger scale and complex control applications, complete operating systems can be employed. In smaller applications, real-time executives are sufficient, providing typical basic components of a process control computer operating systems as described in Ref. 30, such as interrupt handling, task management, communication, and synchronization, as well as time administration, input/output routines, and an operator interface. Often, these functions are custom-designed and integrally included into the application program code.

One distinguishes between the nucleus and the shell of a supervisor. Operating system nucleus processes are called to be of the first kind, are activated by interrupts, and handle events which require attention of the operating system. The shell of the supervisor consists of processes of the second kind; these are handled in the same way as user tasks, i.e., under control of the task management.

The co-processor introduced in this chapter is dedicated to the execution of processes of the first kind. Hence, the co-processor is home to event and time management, as well as task management, communication, and synchronization. Since there is no intrinsic difference between user and system processes of the second kind in real-time environments, the functions of the supervisor shell, such as data exchange with peripherals and file management, can be provided in the form of tasks or subroutines running on the general processor(s). Thus, our approach constitutes a physical implementation of the layered model for real-time operating systems, involving a clear, physical separation between operating system nucleus on one side, and operating system shell and application software on the other.

Basic and theoretical information about the operating system and its rationale was presented in Ref. 2. Here, we only mention that in our prototype implementation the earliest deadline first scheduling policy is implemented. To prevent unnecessary context-switches, a simple modification was introduced. Even if a newly arrived task has an earlier deadline than the executing one, the running task is not pre-empted, unless the situation caused by the latest event requires immediate task re-scheduling. Apart from better performance due to considerably less pre-emptions, the problem of resource contention resulting from mutual exclusion of tasks using critical regions is thus minimized.

An application task, which exists inside, and is intended to be executed by, a task processor, is in one of the following states (see Fig. 5):

Dormant: Being in the dormant state means that a task's code exists, and the task control block (TCB) holds its parameters. Actually, the TCB is split into two parts which hold the system information about the tasks and the tasks' current variable values, and reside in the SRL and in the task processor, respectively.

Scheduled on event: A task is assigned to an event; as soon as the latter occurs the task will be activated.

Ready: If a task's condition to be run is fulfilled, it is moved into the *ready* state. Each time a task enters this state, a schedulability analysis is dynamically performed to check whether its deadline is closer than the one of the executing task and, further, whether its laxity allows the execution of the latter to be continued and, thus, a pre-emption avoided without violating any deadlines.

Buffered activation: Only one instance of a task may be active at the same time. If an already active task is to be activated anew due to fulfillment of some schedule, its activation is buffered, i.e., postponed until the already active instance is either completed or terminated.

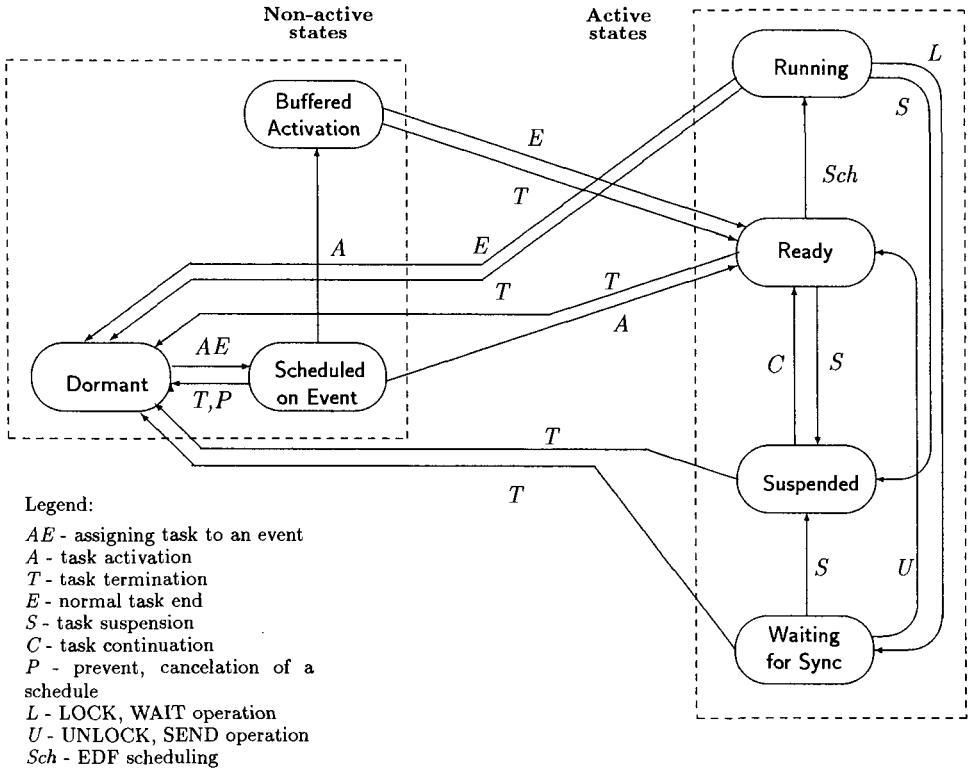


Fig. 5. Task states and their transitions.

Running: A task can only be put from the *ready* into the *running* state by the scheduler, which is part of the operating system being executed by the SRL.

Suspended: A task activation may be suspended. In this state a task retains its internal logical state; however, since suspension is not temporally bounded, timing circumstances become invalid and must be restored upon continuation of task execution by defining a new deadline.

Waiting for synchronization and critical regions: To synchronize with another one, a task may send, or wait for a signal. Also, to protect a part of a program, it may be enclosed into a critical region. If a certain signal is not present on "wait", or a task cannot enter an already occupied critical region, its state changes to *waiting for synchronization.*

Tasks change their states upon operations such as *activate, terminate, end, suspend, continue* (see Table 2) etc., which are included in the syntax of the programming language used to write application software as described in Sec. 6.3. These operations can be scheduled to be executed upon fulfillment of a condition, called event, or a combination of several thereof.

Table 2. Syntax of the programming language features supported by the operating system.

Scheduling support

<non_time_schedule > ::= WHEN <non_time_event> { OR <non_time_event> }
<non_time_event> ::= <interrupt_id> | <shared_variable> [<rel_op> <exp>]
<simple_time_schedule > ::= AT <time_exp> | AFTER <dur_exp>
<periodical_time_schedule>::=[<simple_time_schedule>]EVERY<dur_exp>[DURING<dur_exp>]
<time_schedule> ::= <simple_time_schedule > | <periodical_time_schedule>
<combined_schedule > ::= <non_time_schedule> OR <time_schedule>
<schedule> ::= <non_time_schedule> | <time_schedule> | <combined_schedule>

Task activation

<task_activation> ::= [<schedule>] <task_activation_statement>
<task_activation_statement> ::= ACTIVATE <task_id> DEADLINE IN <dur_exp>

Task termination

<task_termination> ::= [<schedule>] <task_termination_statement>
<task_termination_statement> ::= TERMINATE [<task_id>]

Task prevention

<task_prevention> ::= [<schedule>] <task_prevention_statement>
<task_prevention_statement> ::= PREVENT [<task_id>]

Task suspension

<task_suspension> ::= [<schedule>] <task_suspension_statement>
<task_suspension_statement> ::= SUSPEND [<task_id>]

Task continuation and resumption

<task_continuation> ::= [<schedule>] <task_continuation_statement>
<task_continuation_statement> ::= CONTINUE <task_id> DEADLINE IN <dur_exp>
<task_resume > ::= [EXACTLY] <schedule> <task_resume_statement>
<task_resume_statement> ::= RESUME [<task_id>] [DEADLINE IN <dur_exp>]
TIMEOUT <dur_exp> [ONTIMEOUT <statement>]

Normal task end

<task_end> ::= <task_end_statement>
<task_end_statement> ::= END

Synchronization constructs and critical region concept

<send_statement> ::= SEND <signal_id>
<wait_statement> ::= WAIT <signal_id> TIMEOUT <dur_exp> [ONTIMEOUT <statement>]
ENTER <semaphore_id> [NP] TIMEOUT <dur_exp> <statement>{<statement>}
[ONTIMEOUT <statement>{<statement>}] LEAVE

Explicit assertion of execution time

DURING <dur_exp> [NP] DO <statement>{<statement>}
[ONTIMEOUT <statement>{<statement>}] *FIN*

Status or value acquisition constructs

<task_state_statement> ::= TSTATE <task_id>
<sync_state_statement> ::= SSTATE <sema_id>
<get_RTC_statement> ::= NOW

Interrupt enabling and disabling constructs

<interrupt_enable_statement> ::= ENABLE <interrupt_id>
<interrupt_disable_statement> ::= DISABLE <interrupt_id>

Events can be time-related, and if of sporadic or periodic nature, they may occur at certain instants or after specified time intervals. Non-time-related events are interrupts from the environment and value changes of shared variables. Time-related and non-time-related schedules may be combined. This way, temporally bounded waiting for events from the environment is enabled.

When a schedule is fulfilled, an associated task may be:

activated: the task is put into the ready state and consequently, actually run.
terminated: the task is prematurely terminated and put into the dormant state; also, all schedules related to the task are cancelled.
suspended: the task's activation is suspended, but its internal logical state is preserved, temporal circumstances are disregarded, and thus, invalid.
continued: after a suspension the task can be re-activated; a new deadline must be supplied.
resumed: in general, "resume" is a combination of "suspend" and "continue", i.e., the task is immediately suspended and resumed when the associated schedule is fulfilled. The actual execution of this function is dependent on the option "exactly": if it is not selected, the task changes its states from "running" to "suspended" and again to "ready", as it would if the functions "suspend" and "continue" were called explicitly. However, when "exact" timing is requested, the task does not leave the "running" state. Upon such a function, call execution is halted, and continued immediately after occurrence of the scheduling event, which may be time-related or not, without any delays for scheduling, context switching and similar overhead. Waiting in the suspended state is temporally bounded by a "timeout" clause, which is necessary to retain temporal predictability of task execution. If the timeout expires before the event has occurred, a given "ontimeout" sequence is executed. It is also possible to disregard temporal circumstances and set another deadline, which is reasonable when the "exactly" option is not chosen and the task must be re-scheduled for continuation.

Apart from the above, there are some other related actions, such as:

end: normal task end when execution is completed.
prevent: cancellation of certain schedules.
tstate, sstate, now: monadic operators to acquire task or semaphore states, or time.
enable, disable: external signal masking.

For consistent inter-task synchronization, send and wait functions are provided using signals. Further, mutual exclusion of tasks can be achieved using temporally bounded critical regions; they may be pre-emptable or not as indicated by an "np" option. If any non-pre-emptable critical region exists in a task set, the longest one must be considered in schedulability analyses (for details, refer to Ref. 2). Both features are implemented using semaphores and "lock" and "unlock" functions, which are also shown in the state transition diagram of Fig. 5.

Both waiting for a signal and waiting to enter a critical region are temporally bounded by "timeout" clauses; corresponding "ontimeout" sequences need to be provided. Thus, synchronization delays become predictable and can be considered in task run-time estimation and in schedulability analysis.

Similarly, it is possible to temporally bound any part of a program using a "during" construct. This has two functions: to protect any temporally non-deterministic features (e.g., peripheral operations); and to assert explicit estimation of a certain program part's run-time, whose automatically obtained estimate is expected to be imprecise and pessimistic. Automatic estimation of the run-time is overridden by this assertion.

6. Programming Languages and Worst Case Execution Time Analysis

In process control, two categories of control computers are generally used: generic computer systems and industrial programmable controllers. For programming the former, universal high-level programming languages are used; unfortunately, many applications are still programmed in assembly languages which means much more error-prone programs and less productive application program development. The latter are programmed using a variety of vendor-specific programming means.

6.1. *Industrial controller programming*

Vendors of programmable logic controllers usually provide their own programming tools. Most of their properties have been included in the standard IEC 61131-3[13] with the goal of harmonizing program design. In this section it will be shown that they actually provide a modern, simple, graphical means of programming which can also be employed for programming generic systems.

The programming tools can be grouped into five categories:

- instruction list: a low-level assembly-like language.
- ladder diagram: logic functions expressed in a form of relay-logic.
- function block diagram: standard and commonly used routines that are graphically represented by blocks which are then "wired" together.
- structured text: structured high-level-like language.
- sequential function chart: graphical representation of program structures on the task level, including parallel and alternative execution, etc.

While the first two are merely meant for compatibility with some older systems, the latter three represent powerful tools for application design.

Function blocks are graphical means for representing program units on different levels and structuring them into nested entities. At the outside, they show their input and output variables; the functions implemented by them can be defined with the structured text. With sequential function charts, it is possible to define relations at the task level.

Although it is beyond the scope of this chapter, it is obvious that this means could be used for programming generic control computers as well: standard procedures are programmed, analyzed and verified, and put into ROMs. The application programmer merely needs to "wire" them together into specific applications. This way, the complexity of designs can be greatly reduced, and the verification of programs composed out of standard program blocks becomes rather easy.

6.2. *High-level real-time programming languages*

A high-level programming language to be employed for conventional programming of applications in real-time environments should fulfill the following properties:

- it should support predictability of temporal behavior of programs: most of the commercially available and broadly used languages do not meet this requirement. It is thus left to the programmer to use for sensitive parts of applications only constructs which do not jeopardies predictability, and to analyze, and at run-time supervise, the temporal behavior of program execution.
- it should be safe to allow for reliable programs: it should support strong type checking and make provisions for exception handling.
- support for multiprogramming: it should include task support and good synchronization means.
- it should provide good hardware access: peripheral device drivers are a very common and important part of real-time applications.
- support for programming-in-the-large: it should be modular and allow separate compilation.
- it should be maintainable: real-time software has a long life expectancy; programs should be easy to read, understand, and modify.

High-level programming languages for programming generic computer systems used in control applications can be further divided into two categories: implementation languages and real-time languages.

Implementation languages do not provide much support for real-time programming. Application programmers have to call certain library or operating system kernel routines which deal with the real-time requirements to a certain extent. By far the most common one among implementation languages is C or C++. Its advantages are good availability, connectivity with, and support for program development tools; most programmers are familiar with it; the disadvantages are poor readability and maintainability, and the inability to meet almost all of the above mentioned requirements of real-time programming.

For programming real-time applications, proper real-time programming languages should be used which should fulfill the above enumerated requirements. There are not many such languages commercially available; many of them were either academic prototypes or proprietary languages of certain large companies or institutions.

Two well established and broadly used real-time languages should be mentioned here, Ada and PEARL.

Ada is probably the most widely used language for real-time applications. It has a number of good properties including most of the ones mentioned above. One major disadvantage, however, is that it is very large and complex. It includes almost any feature of any other modern language and is, thus, hard to learn and use, and it is difficult to produce efficient code.

PEARL[6] was designed and standardized by a group of German engineers in the early 1970's, and has been re-designed in the late 1990's (PEARL90). It also fulfills most of the above requirements; it is easily understandable, much simpler, and matches the thinking of the control engineer much more closely than Ada. As an example, consider the following PEARL implementation of an alarm clock: `AT 7:00 EVERY 3SEC DURING 10MIN ACTIVATE RINGING.`

Unfortunately, PEARL is not broadly used outside of Germany and Europe, and does not have much support by other commercial program development tools.

In recent years, object orientation is also being considered in the real-time programming domain. The classical programming languages are being adapted to this paradigm, and new ones are emerging. The most important one to be mentioned in this context is Real-Time Java. Although the common Java is originating from the process control domain, there are still some obstacles to its consistent applicability, the most serious one being the temporal impact of garbage collection.

6.3. *Programming the prototype implementation*

As a part of our project, a new real-time programming language was developed and a compiler with a built-in run-time analyzer designed. Being aware of the unpopularity of producing new languages and writing own compilers, we considered the following arguments for doing so:

- In spite of the growing urge, we found that commercially available and widely used high-level programming languages and compilers still do not meet all the needs of embedded hard real-time application programming. Most languages used for such systems produce codes that lead in one way or another to unpredictable timing behavior.
- Another pro-argument was the co-design of our particular experimental hardware architecture, the corresponding operating system and the application development tool. It is impossible to support the features of the former two with any existing compiler.
- Further, experience with run-time analysis of high-level source code programs showed that access to the compiler-internal data structures is required, which is not possible for commercial compilers.

Owing to these considerations, we decided to design our own programming language "miniPEARL" based on the standardized programming language PEARL

with certain modifications and simplifications.[31,32] MiniPEARL is a strongly typed and structured high-level real-time programming language. Certain constructs were added to standard PEARL, e.g., enhanced tasking operations and real-time specific programming support as described in the previous section.

Strict temporal determinism on the programming language level can only be achieved under the assumption that each statement is executed in a bounded and a priori predictable time period. For this reason, each feature that could take an arbitrarily long time to execute is either renounced in miniPEARL or is guarded with a timeout clause.

To allow for that, it was necessary to introduce certain restrictions to the features of standard PEARL:

- GOTOs are not allowed: Their use can result in unstructured code which is difficult to manage. Instead, EXIT and LOOP statements are introduced. The former is used for preliminary exit from innermost structures, and the latter is used in the REPEAT construct for immediate initiation of the next iteration of a loop. As a consequence, labels, except for procedure and task declarations, are obsolete and therefore renounced.
- Each loop block must be tightly bounded: Lower and upper counts of iterations must be present and defined with compile-time-constant expressions, so that the longest execution time of a loop can be estimated. The "while" or "repeat" conditions should be replaced by explicit IF–EXIT statements within loops.
- Pointers and recursion are not allowed: Their use can result in severe memory-management problems. They can produce temporally non-deterministic actions, and cannot be considered in a priori timing analyses.
- Each synchronization construct must be temporally bounded: Synchronization constructs (for example, critical regions or semaphores) may take arbitrarily long times for their execution. This must be bounded in real-time systems. Each such command must be temporally guarded, and an explicitly defined action must be provided for the case that a time-out occurs.
- Explicitly asserted execution times: In some cases, because of the nature of a program, estimation may yield very pessimistic execution times. To resolve this problem, additional execution information must be given by the programmer. This can be done by adding new constructs (pragmas) into the program code, as proposed in.[22,23] However, this method requires complex analyses, and is not feasible in all situations. To overcome this problem, the explicit execution time of a program segment can be asserted by the system developer, overriding the estimated result with the asserted one. However, to guarantee that the actual execution time will not be longer than the declared one, for safety reasons the segment must be guarded by time-out control, and a time-out action must be present.
- Task scheduler support: The scheduling algorithm in the operating system kernel processor relies on the residual execution time of tasks. This time is computed

```
ADD16   MACRO
        MOVE.W  &1,D0
        ADD.W   &2,D0
        MOVE.W  D0,&1
        TIME 36
        ENDM
```

Fig. 6. An example for a translation macro.

as the maximum execution time of a task minus the accumulated running time. However, the actual execution time is expected to be shorter than the estimated one. To achieve a more reliable and realistic estimation of the residual time, it can be updated explicitly, whenever possible, by asserting into the code information on the estimated residual execution time, which gradually becomes more precise as a task approaches its end.

Since the temporal execution behavior of a program as a whole is tightly correlated to that of all its parts, all program code must be compiled at the same time. Although the organization of programs is modular, modules may not be compiled separately (except for syntax checking).

6.4. *Worst-case execution time estimation*

To estimate the execution times of programs written in the high-level programming language miniPEARL, the following three steps are carried through:

(1) First, the compiler transforms application source code into an intermediate form, in the course of which a modified syntax tree is constructed and symbol tables for identifiers, procedures and tasks are built. To this internal form of a program global code optimization is applied. Constructs like straight line code blocks, alternatives, condition evaluations, loops etc. are located.

(2) In the second step, the duration of each straight line portion of assembly language (or machine) code is calculated using so-called translation macros. These macros are specific to the hardware architecture and processor used. To support different platforms, different sets of macros can be used.

Each element of a syntax tree corresponds to one or more macros. As an example, a macro that corresponds to the addition of two integer variables is shown in Fig. 6.

In this macro, the placeholders `&1` and `&2` are replaced by the first and the second calling parameter, respectively. Output code is generated in the code generation phase either in assembly or in machine language form. In the first case, the generated code must be assembled before it is loaded into the target

system. In the second case, compiling is faster, but the compiler output is not readable and, thus, difficult to verify.

The same macro is also used in the timing analysis phase. The directive TIME states the basic execution time of this macro's code. It is not implemented as an attribute of the macro because, for code optimization, conditional expansion of macros is allowed and execution times may vary with operand types or values. The exact execution time will be established when the macro is called by the compiler and operands are specified. For the example macro, its call could look like:

$$\text{ADDI L(4)}, 3$$

where the notation L(4) represents an absolutely addressed local variable residing in location 4 and the second operand is a constant. Additional time used for effective address calculation and operand fetches and stores is added, yielding the total execution time of the macro.

For further analysis, straight line blocks are replaced by corresponding delays and are not considered any more.

(3) Finally, in the third step, the nested structures in the compiled source code are searched recursively until the innermost one is located. This is then replaced by a delay corresponding to its worst-case execution time: alternatives are reduced to the execution time of the longest path; execution times of loop bodies are multiplied by the maximum number of iterations; subroutine calls are reduced to their durations. Then, the same procedure is performed on the next enclosing nested structure. Finally, recursive application up to the outermost structures yields the execution time of the program.

Since the language is relatively simple, in comparison with others it can be more efficiently compiled into object code, which can be more easily optimized. Moreover, analysis of the code's temporal execution behavior is easier, more accurate and less pessimistic. Through accessing compiler-internal data structures, the syntax tree can be precisely analyzed, and any global or local program code optimizations considered.

6.4.1. *Pessimism in task run-time estimations*

A drawback of many methods to estimate task execution times, and one of the reasons why they are not more widely used, is that they yield relatively pessimistic results. The reasons can be found on all layers involved in programming, from non-deterministic language features via operating system overhead to hardware behavior.

In our approach, the influence of the operating system overhead was minimized by migrating the kernel to a separate processor. Only context switches and non-pre-emptable critical regions have to be considered.

State-of-the-art processors are optimized for high average performance and include features that behave, as a rule, temporally non-deterministically, like caching, internal parallelism, pipelining etc. These features cannot be consistently considered in temporal analysis. Often, worst-case execution times are given, and sometimes even statistically estimated average times, only. The execution times of instructions heavily depend on their sequential order in program code. Animating sequences of instructions often fails, because their exact execution behavior is not disclosed by the processor manufacturers.

In our concept of macros it is possible to see whether a preceding macro does not end with a jump or branch enabling the subsequent one to start earlier with a pre-fetch. A macro can, thus, give the subsequent one a "bonus", which may be subtracted from its execution time.

For even more precise run-time determination, our analyser provides another option—generation of pilot code. The idea behind is to measure execution times of actual code on a target system. Since a target system may not be available at the time of software development, and since it is not always easy to set the worst-case test scenario, the compiler is used to generate pilot object code. This is characterized by always running through the path with the longest execution time (the longest alternative in IF and CASE statements is always selected, the maximum possible number of loop iterations is forced, etc.). Further, input/output commands are replaced by corresponding delays. Thus, only partially built or similar target systems with the same processor are sufficient for the measurements.

Several modern microprocessors already include special testing and debugging support, the Background Debugging Mode (BDM). The new version of our run-time analyser will be able to automatically communicate with the microprocessor, measure the critical parts of programs and, in the subsequent steps, use the most precise results of the program run-time measurements, already taking into account the effects of parallel execution, caching and other effects that are difficult to consider otherwise.

Most problems, however, are introduced by programming itself. Studies (cp., e.g., Refs. 22 and 23) based on supplying additional knowledge about programs to analysers were made to achieve better estimation of run-times. Although yielding very good results which are close to the actual values, these solutions do not appear to be sufficiently practical for implementation yet. If application designers feel that an automatic estimation is excessively pessimistic, they can thoroughly test and measure the critical parts of a given code until they become familiar with the temporal circumstances. In this case they can make use of the possibility to explicitly define execution times for such code segments and, thus, override automatic estimation.

During run time, however, certain controls have to be provided to ensure that the program execution complies with the limitations asserted into the program language constructs.

7. Safety critical applications

It seems to be easier to assure dependable hardware architectures than dependable software. The reason for this is that verification methodologies and techniques for hardware design are well established for already a long time. Hardware can be safety licensed for critical applications by licensing authorities (like the TÜV in Germany) using well established procedures.

Software is much more complex, taking into account the entire life cycle from specification, via design and coding, to compilation into machine code which is the actual final result. There is a number of hazards to its integrity, from faults and errors common in specifications and in all levels of design, to implementation problems like arithmetic precision (e.g., the Ariane 5 disaster), to mention just a few.

Licensing of software-based systems is extremely difficult. Although considerable research activities have been devoted to the area of formal specification and verification of programs, the techniques developed are generally applicable to relatively simple cases only. Also, they are still not formally accepted by the licensing authorities, as they rely in turn on complex computer tools which are inherently unsafe.

The only method officially recognized by the German licensing authorities is diverse back translation.[16] It consists of re-gaining a requirement specification from the software examined by several, independently working licensors or groups. To eliminate possible effects of error-prone compilers, this must be carried out on the machine code read out of the target system. It is obvious that this method is only feasible for very limited applications such as emergency shut-down systems. It is not practically usable for most industrial applications.

For these reasons, in extremely critical applications and systems, like safety back-up systems in nuclear power plants or avionics, where safety is crucial and formal licenses are necessary, in general only hardware-based or trivially simple programmed systems can be licensed for the time being. However, due to economic reasons, it is necessary to provide methods and technologies allowing program-based systems of some complexity to be safety licensed as well. In this sense, alternative control systems have been elaborated,[9] which can be rigorously proven correct with acceptable effort even for systems with realistic complexity.

8. Typical Set-up and Design of Applications

As an example, let us finally consider our embedded computer controller to be built into a classical industrial process consisting of mechanical production machines. Typically, such a process is physically distributed over a production hall. The control system senses system states, characteristic values, and data inputs and, with calculated results, controls the actuators. The times to react upon events from the process are in the order of magnitude of milliseconds and must be guaranteed.

Sensors provide two kinds of data: they acquire values of input variables and notify the control system of events, which influence the further process behavior

(e.g., changes of certain internal states like start or termination of certain activities, operator interventions, alerts, etc.). The former are polled by intelligent peripheral interfaces, and controlled by task processors synchronously with application program processing. The events have the form of asynchronous signals, and are directly fed to the kernel processor's primary reaction layer. Upon occurrence of these signals of programmed time events and changes of synchronizers and common variables, unified system events are reported to the secondary reaction layer, where associated tasks are then scheduled earliest deadline first. If necessary, a context switch is requested in the corresponding task processor to activate a task handling an event. Such tasks may request certain operating system kernel services during their execution.

The intelligent peripheral interfaces are selected to be mounted on the devices from which they acquire input data and/or in which they drive actuators. The associated application tasks are statically mapped onto one to four task processors, depending on the performance required.

The application tasks are programmed in miniPEARL, which represents a simple, deterministic and, to certain extent, safe[3] subset of a standardized high-level programming language, which is convenient for control applications. The particular hardware configuration is described in the system part of the application program. Then, the application or problem part is designed, containing the actual algorithmic implementation. The syntax of miniPEARL already includes all necessary scheduling constructs and other operating system calls; the application programmer does not have to be aware of specific operating system issues.

Computer-aided application engineering is not supported yet. It is one of the directions for further research which will feature aspects like graphical programming, specification animation, automatic program generation, rapid prototyping, and stepwise refinement of functional and temporal issues. An interesting and motivating approach towards the design of hardware configurations and software systems, however based on DSPs, can be found in Ref. 18.

9. Conclusion

In order to assure predictable execution behavior of real-time systems, it is necessary to determine *a priori* bounds for task execution times. In this chapter a consistent, holistic design of an experimental control system for embedded applications operating in the hard real-time domain is described. It is shown how the execution times of tasks running on task processors can be estimated, providing the information necessary for schedulability analysis.

In this project, we did not intend to develop many novel concepts. We merely tried to strictly apply known ones, and to combine different design domains to obtain practically usable experience in designing predictably behaving embedded control systems.

This project is an on-going one: having concluded the hardware design, built the operating system and the compiler with the built-in analyser, we decided to change the hardware implementation radically by replacing the transputer in the implementation of the secondary reaction layer by another microprocessor (Motorola ColdFire). Also, the task processors are being re-designed using the same processors as they feature better possibilities to estimate the temporal behavior and sufficient performance.

The proposed approach involves many components; its price exceeds the price of standard industrial controllers. Although the hardware price is often a significant parameter, it is very low in comparison to the costs of an application environment and of software design. Furthermore, a consistently designed control system can prevent costs of production stand-stills and/or damages, or even endangerment of human health or lives, caused by failures, which can by far exceed the hardware costs.

By designing embedded control systems in this way, the requirements of hard real-time systems, viz., timeliness, simultaneity, predictability, and dependability, are better met than by state-of-the-art technology.

Acknowledgements

The authors wish to thank Domen Verber, Roman Gumzej and Stanislav Moraus, their colleagues from the Real-Time Systems Laboratory of the Faculty of Electrical Engineering and Computer Science in Maribor, Slovenia, for the elaboration and implementation of the laboratory prototype.

Work on this project has been supported by the Ministry of Science and Technology of the Republic of Slovenia through the grant no. J2-9043-0796-97.

References

1. W. Blume and W. Klinker, *The Sensor/Actuator Bus: Theory and Practice of Interbus-S.* Landsberg/Lech: Verlag Moderne Industrie—Die Bibliothek der Technik, Band 78, 1993.
2. M. Colnarič, W. A. Halang and R. M. Tol, Hardware supported hard real-time operating system kernel, *Microprocessors and Microsystems* **18**, 10 (1994) 579–591.
3. M. Colnarič, D. Verber and W. A. Halang, Supporting high integrity and behavioral predictability of hard real-time systems, *Informatica* (Special Issue on Parallel and Distributed Real-Time Systems) **19**, 1 (1995) 59–69.
4. J. Cooling, Task scheduler for hard real-time embedded systems, *Proc. Int'l Workshop on Systems Engineering for Real-Time Applications* (IEE, London, 1993) 196–201.
5. *DIN 44 300 A2: Informationsverarbeitung* (Beuth Verlag, Berlin-Cologne, 1972).
6. *DIN 66 253: Programming Language PEARL, Part 1: Basic PEARL* (Berlin, 1981).
7. W. A. Halang and A. D. Stoyenko, *Constructing Predictable Real Time Systems* (Kluwer Academic Publishers, Boston-Dordrecht-London, 1991).
8. W. A. Halang, Achieving jitter-free and predictable real-time control by accurately timed computer peripherals, *Control Engineering Practice* **1**, 6 (1996) 979–987.
9. W. A. Halang and M. Colnarič, Outsourcing the Development of High Integrity Software, in *Software Quality—the Way to Excellence. Proc. Sixth European Conference*

on Software Quality, ed. W. Wintersteiger (European Organization for Quality and Arbeitsgemeinschaft für Datenverarbeitung, Vienna, 1999) 495–501.
10. R. K. J. Henn, *Deterministische Modelle für die Prozessorzuteilung in einer harten Realzeit-Umgebung*, PhD thesis, Technical University of Munich, 1975.
11. R. K. J. Henn, Zeitgerechte Prozessorzuteilung in einer harten Realzeit-Umgebung, in *Informatik-Fachberichte 5* (Springer-Verlag, Berlin-Heidelberg, 1976) 343–359.
12. R. K. J. Henn, Feasible processor allocation in a hard real-time environment. *Real-Time Systems* **1**, 1 (1989) 77–93.
13. International Electrotechnical Commission: Standard IEC 61131-3 *Programmable Controllers, Part 3: Programming Languages*, Geneva, 1992.
14. H. H. Johnson and A. Maddison, Deadline scheduling for a real-time multiprocessor. *Proc. Eurocomp. Conference* (1974) 139–153.
15. J. Kershaw, The VIPER microprocessor. Technical Report 87014, Royal Signals and Radar Establishment, Malvern, Worcs (Her Majesties' Stationery Office, London, 1987).
16. H. Krebs and U. Haspel, Ein Verfahren zur Software-Verifikation, *Regelungstechnische Praxis rtp* **26** (1984)73–78.
17. J. Labetoulle, Real time scheduling in a multiprocessor environment, Technical report (IRIA Laboria, Rocquencourt, 1976).
18. R. Lauwereins, M. Engels, M. Adé and J. A. Peperstraete, Grape-II: A system-level prototyping environment for DSP applications, *IEEE Computer* **28**, 2 (1995) 35–44.
19. L. Lindh, *Utilization of Hardware Parallelism in Realizing Real-Time Kernels*, PhD thesis (Royal Institute of Technology, Stockholm, 1989).
20. C. L. Liu and J. W. Layland, Scheduling algorithms for multiprogramming in a hard real-time environment, *Journal of the ACM* **20**, 1 (1973) 46–61.
21. A. Mok, *Fundamental Design Problems of Distributed Systems for the Hard-Real-Time Environment*, PhD thesis (Massachusetts Institute of Technology, 1983).
22. C. Y. Park, Predicting program execution times by analyzing static and dynamic program paths, *Real-Time Systems* **5**, 1 (1993) 31–62.
23. P. Puschner and C. Koza, Calculating the maximum execution time of real-time programs, *Real-Time Systems* **1**, 2 (1989) 159–176.
24. J. Roos, The performance of a prototype coprocessor for Ada tasking, *Proc. Conference Tri-Ada* (ACM, 1990) 265–279.
25. P. Sorenson, *A Methodology for Real-Time System Development*, PhD thesis (University of Toronto, 1974).
26. J. A. Stankovic, Misconceptions about real-time computing, *IEEE Computer* **21**, 10 (1988) 10–19.
27. J. A. Stankovic and K. Ramamritham, Editorial: What is predictability for real-time systems, *Real-Time Systems* **2**, 4 (1990) 246–254.
28. J. A. Stankovic, Adjustable flow control filters, reflective memories, and coprocessors as support for distributed real-time systems, *International Journal of Mini and Microcomputers* **17**, 1 (1995).
29. A. D. Stoyenko, *A Real-Time Language With A Schedulability Analyzer*, PhD thesis (University of Toronto, 1987).
30. Guideline VDI/VDE 3554: *Funktionelle Beschreibung von Prozeßrechner-Betriebssystemen.*
31. D. Verber and M. Colnarič, A tool for estimation of real-time process execution times, *Proc. Software Engineering for Real-Time Applications Workshop* (IEE, London, 1993) 166–171.
32. D. Verber and M. Colnarič, Programming and time analysis of hard real-time applications, *Proc. 20th IFAC/IFIP Workshop on Real Time Programming* (1995).

This page is intentionally left blank

INDEX

Computer Aided and Integrated Manufacturing Systems

This is an invaluable five-volume reference on the very broad and highly significant subject of computer aided and integrated manufacturing systems. It is a set of distinctly titled and well-harmonized volumes by leading experts on the international scene.

The techniques and technologies used in computer aided and integrated manufacturing systems have produced, and will no doubt continue to produce, major annual improvements in productivity, which is defined as the goods and services produced from each hour of work. This publication deals particularly with more effective utilization of labor and capital, especially information technology systems. Together the five volumes treat comprehensively the major techniques and technologies that are involved.

World Scientific
www.worldscientific.com
5249 hc

ISBN 981-238-339-5(set)
9 789812 383396

ISBN 981-238-979-2
9 789812 389794