AN ABSTRACT OF THE THESIS OF

Joohee Kim for the degree of Doctor of Philosophy in Industrial Engineering presented on June 27, 1988.

Title: An Expert System for Flexible Manufacturing System Scheduling: Knowledge Acquisition and Development

Abstract approved: Redacted for Privacy
Eugene F. Fichter

Expert systems have been suggested as a solution for difficult problems, including FMS scheduling. As one of the aspects of artificial intelligence (AI), expert systems have achieved considerable success in recent years in medical science, chemistry, and engineering. However, building an expert system is a difficult task, the most crucial problem being that of knowledge acquisition. Obtaining expert knowledge is a difficult and time-consuming process. Moreover, since FMSs represent a relatively new technology, experts capable of FMS planning and scheduling are generally unavailable.

One possible solution for this problem is to train a non-expert operator, allow the operator to practice with a simulated system and accumulate experience, and then build an expert system using the newly acquired

knowledge. To this end, an interactive graphic simulation method for the effective utilization of human pattern-recognition ability is proposed. Once the required knowledge is elicited through an interactive graphic simulation model, an expert system is developed from acquired rules. The method includes an FMS simulation model, a Gantt chart-based schedule, a simulator, an expert system, and a human operator. First, an initial schedule is simulated, utilizing the expert system to determine the loading sequence and a dispatching rule. The schedule is then updated by an expert system and/or human operator with the capability of maximizing schedule objectives, while at the same time saving reasons for changes as new production rules, which are subsequently generalized and added to the expert system knowledge base.

The system is implemented in Smalltalk/V on an IBM PC/AT and the implementation is based upon a detailed sample problem. It was determined that a human operator can obtain near-optimum schedules in short time periods, at the same time gaining valuable experience in use of the scheduling process. Furthermore, it was determined that this model can be a useful training device for inexperienced operators and a time-saving decision-making aid for expert schedulers.

An Expert System for Flexible Manufacturing System Scheduling: Knowledge Acquisition and Development

by

Joohee Kim

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Completed June 27, 1988

Commencement June 1989

APPROVED:

Redacted for Privacy

Professor of Industrial and Manufacturing Engineering

Redacted for Privacy

Head of Department of Industrial and Manufacturing Engineering

Redacted for Privacy

Dean of Graduate School

Date thesis is presented June 27, 1988

Typed by B. McMechan for Joohee Kim

Acknowledgements

I wish to express sincere gratitude to my major professor, Dr. Eugene F. Fichter, for his continuous guidance, advice and most friendly assistance during the preparation of this thesis.

I also wish to express gratitude to Dr. Kenneth H. Funk of the Department of Industrial and Manufacturing Engineering for his valuable suggestions and comments.

Special thanks go to the faculty members in the Department of Industrial and Manufacturing Engineering: Dr. West, Dr. McDowell, Dr. Safford, and Dr. Randhawa; and to my committee members for their helpful advice: Dr. David S. Birkes of the Department of Statistics, Dr. William S. Bregar of the Department of Computer Science, and Dr. James C. Rawers of the Department of Mechanical Engineering. In particular, Dr. West provided me with financial support and valuable experience.

My thanks are extended to friends and colleagues in the Department of Industrial and Manufacturing Engineering for the many discussions which helped me to formulate some of the ideas presented: Tony Chou, Bauji Zhou, Joongnam Kim and Sunuk Kim.

Finally, I would like to thank the members of the Corvallis Korean Church for the love and faith that I will live with.

"For God, who commanded the light to shine out of darkness, hath shined in our hearts, to give the light of the knowledge of the glory of God in the face of Jesus Christ. But we have this treasure in earthen vessels, that the excellency of the power may be of God, and not of us."

2 Corinthians 4:6-7

To my parents, brother and sisters, and my wife, Sooyoun Kim, for their love, encouragement, and sacrifice, without which I would not have been able to finish my graduate work.

Table of Contents

Table of Contents (continued)

List of Figures

List of Tables

List of Tables (continued)

AN EXPERT SYSTEM FOR FLEXIBLE MANUFACTURING SYSTEM SCHEDULING: KNOWLEDGE ACQUISITION AND DEVELOPMENT

1.0 INTRODUCTION

1.1 FMS Background

In recent years a great deal of research has been undertaken concerning the need for improved productivity in the mid-volume manufacturing area. For example, about 75 percent of all metalwork manufacture is involved with the batch production of a variety of products in mid-volume quantities (Cook, 1975) and a number of studies have suggested that the ideal production system for this environment is a flexible, computer-integrated manufacturing system. This type of system is often referred to as a flexible manufacturing system, or FMS.

An FMS is an integrated system of machine modules and material handling equipment placed under computer control for the automatic processing and manufacturing of parts (Chen & Talavage, 1982). The major components of an FMS include:

1) Work stations. Machine tools that perform required manufacturing operations and which are typically numerically controlled machines, e.g., CNC mills and lathes.
2) Material handling systems which transport workpieces from one work station to the next, including conveyors, stacker cranes, tow-carts, and automated guided vehicles.
3) Auxiliary equipment, including such facilities as inspection stations, in-process storage areas, loading/unloading stations, or washing and heat treatment stations.
4) Control systems responsible for automatic monitoring, i.e., controlling and scheduling the operations of other components of the system.

Examples of existing FMSs have been analyzed and described by Groover (1984), Ranky (1983), Stecke (1983), and Stecke and Solberg (1981). In 1982, Dupont-Gatelmand surveyed the different types of FMSs used for machining and assembly operations, focusing upon three categories of systems: (1) flexible modules and units, (2) flexible transfer lines, and (3) unaligned flexible systems. Within each of these categories there are several sub-groups, classified in accordance with the type of operating mode and material handling system in use. Whatever the specific purposes

of flexible manufacturing systems, they share one common purpose: To produce a wide variety of parts, using the same set of facilities. An FMS can be programmed to suit a variety of production requirements, thereby fully utilizing resources for processing of a broad range of products. Some of the benefits realized through the application of an FMS include:

1) Reduced direct labor costs,
2) Improved machine utilization,
3) Reduced work-in-process inventories,
4) Consistency of product quality, and
5) Product and process flexibility, allowing rapid response to changes in demand.

Flexibility is the most important aspect of an FMS in terms of productivity. Increased manufacturing flexibility can offer greater efficiency and increase levels of productivity. However, at the same time the use of an FMS does make the process of production control more complex and difficult since an FMS consists of many interconnected hardware and software components, as well as other manufacturing resources, including tools, pallets, and fixtures that may be in short supply. Therefore, any decision to allocate resources to the production of one workpiece may affect the quantities of resources available to the production of others (Suri & Whitney, 1984). For this reason, FMS production control requires software which is capable

of assisting the FMS operator in planning, scheduling, and monitoring the system.

The literature of operations research has in recent years focused considerable attention upon FMS production scheduling, which in other terms is a specialized instance of job shop scheduling (Blackstone et al., 1982; Buzacott, 1982; French, 1982). It is generally accepted in the field of operations research that the problem of computerized job shop scheduling is at best difficult, meaning that to date no one has been able to achieve a polynomially-bound software solution (Lenstra & Rinnooykan, 1978). In other words, no practical method exists which guarantees generation of an optimum schedule. Common operations research approaches have included:

1) Analytical methods, using mathematical programming, such as linear programming, mixed integer programming, and queueing theory.
2) Heuristic rules, such as the "first-come-first-served" rule, the "shortest processing job first" rule, or the "earliest due date" rule.
3) Simulation studies to test and evaluate system performance.

The problem is that most of these techniques have encompassed a level of computer technology which is too time-consuming for a decision-making environment, or

the systems have been limited in their ability to capture critical system detail. For example, in queueing theory the interarrival time and service time should be exponentially distributed and the processing order should be first-come-first-served, which is usually not realistic. Many similar heuristic rules have been developed, but they are static rules and are not generally suitable for dynamically changing situations.

Knowledge-based expert systems (KBES), one aspect of artificial intelligence (AI), have been suggested as approaches to the solution of the difficult problem of FMS scheduling. Several researchers have undertaken the study of the representation of scheduling in AI. Bullers et al. (1980) demonstrated how predicate logic and theorem proving techniques, using resolution techniques, could be used in a manufacturing environment. However, predicate logic has been widely criticized for lacking the expressiveness necessary for the representation of complex knowledge (Fikes & Kehler, 1985). Fox (1983, 1984) used frames to construct a complex job shop modeling system, ISIS. ISIS is a constraint-directed reasoning system for scheduling a job shop, adopting a heuristic approach to schedule generation. In this sense, knowledge consists mainly of the awareness of constraints and the expertise of the scheduler since expert knowledge can be used to relax constraints

and to determine the proper bounds of solutions. Subsequently, Bruno et al. (1986) used production rules for knowledge representation and developed a production scheduling expert system using OPS5.

Most of the papers reviewed have dealt with knowledge representation methods and/or efficient search techniques to obtain useful solutions for the use of the least possible production times, but they have failed to confront the problem of elicitation of expert knowledge. Hart (1985) analyzed the difficulties of interviewing human experts for the transfer of knowledge to computer programs. The principal problem is not only to find expert knowledge, but that the process is so time-consuming. Furthermore, expert knowledge is often unavailable since in many cases experts simply do not exist. One possible alternative to the solution of this problem might be to train a non-expert operator, allowing the operator to practice with a simulated system, accumulate experience, and then build an expert system using this newly acquired expertise.

The objective of this study is to demonstrate how to develop an expert system for scheduling an FMS, based upon a feasible means of knowledge acquisition. This project is organized as follows. The structure of an FMS model and FMS production control problems are illustrated in Chapter 2. In Chapter 3, characteristics of scheduling problems and some of the existing

approaches are discussed. Chapter 4 encompasses a description of the benefits, as well as the problems, of building an expert system for FMS scheduling, in the process demonstrating a general framework for the development of an expert system with knowledge acquisition. Chapter 5 presents an implementation of the developed system, using an example problem to examine the system in detail and illustrate the rules that have been collected through experimentation. Conclusions and the scope of future research needs are provided in Chapter 6.

1.2 Contribution of This Research

Software and hardware advances in computer technology provide areas for investigation which may lead to further improvements in FMS performance. Even though a number of investigations have been completed in this area during the past few years, an operational software system for the control of an FMS, which is (1) intelligent, (2) experience-based, and (3) self-improving (Blessing & Watford, 1987), has not been developed to date.

The current study presents an original way of building an intelligent FMS scheduling system and implements a prototype model. The system is intelligent

in the sense that it is capable of accessing and utilizing several system knowledge bases. It is also experience-based because the system uses past experience from previous simulation experiments to form rules for better system performance in the future. Research is ongoing and the system is undergoing improvement as additional experiments are performed. Although the prototype does not have self-improvement capability, it is possible to improve and expand its knowledge bases with human operator assistance, as the operator accumulates experience through use of the simulated system.

In general sense, this investigation demonstrates the synergistic application of knowledge related to:

1) the FMS and its control problems,
2) the use of simulation models,
3) the use of management science and operations research techniques,
4) the application of AI techniques,
5) human factors for the man-machine interface of using computer graphics, and
6) the use of interactive problem-solving methods.

The main research contribution of the current study is development of an integrated FMS system, including the application of expert systems, simulation modeling, and computer graphics for FMS scheduling. Figure 1.1 illustrates an integrative view of the FMS scheduling system with the other necessary components.

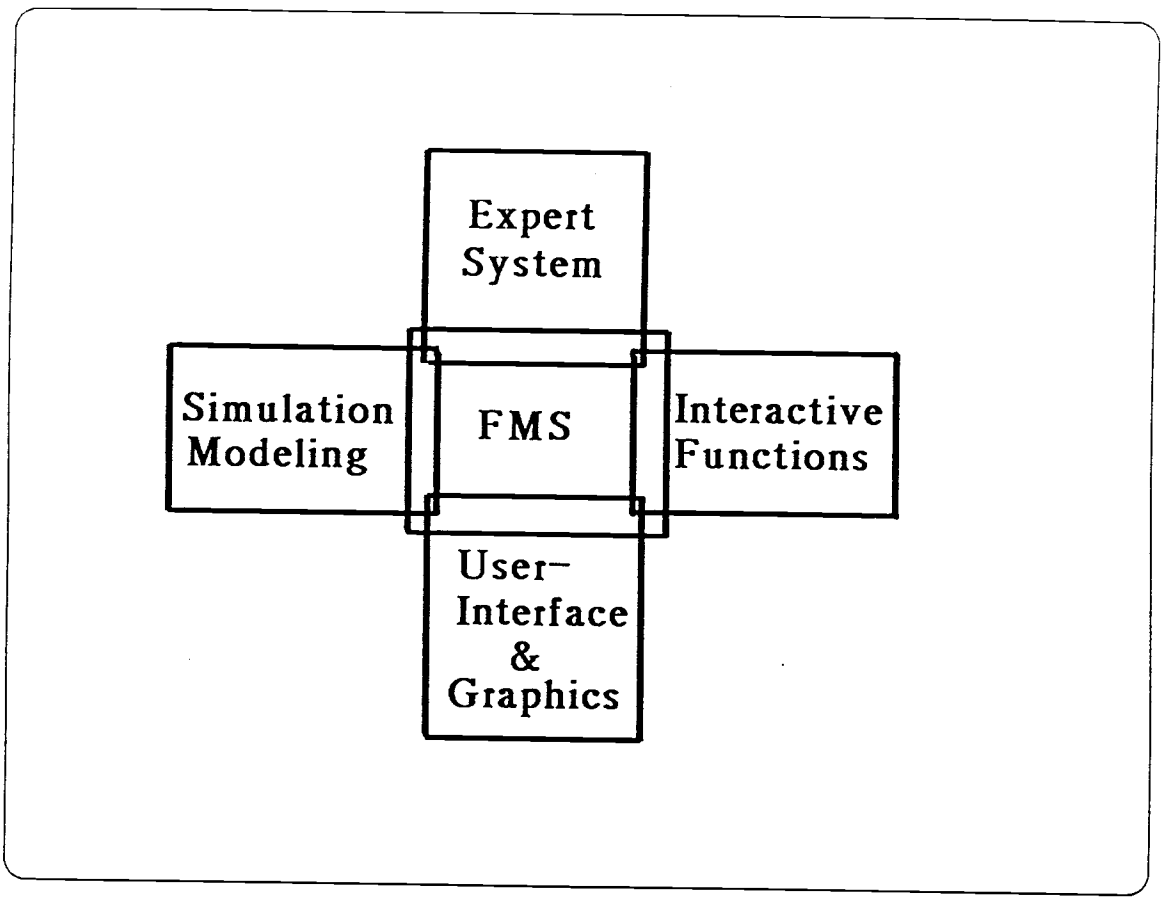


Figure 1.1 Integrated view of FMS scheduling system.

Simulation modeling is one of several requirements for an effective application, along with the development of effective tools for a graphic representation user interface in which the interface is tailored to particular scheduling applications addressing the needs of human schedulers. A more productive view of the application of expert systems is to utilize them as a component of the total scheduling environment for the support of the decision-making process of generating a schedule. The interactive functions of the system allow the operator to control the entire system, providing all of the advantages of an integrated approach without excessive computational requirements. Schedules may be updated frequently and unexpected events, such as machine failures or similar occurrences, may be dealt with immediately.

The principal differences of the prototype system from other intelligent scheduling systems are that it may be used to generate feasible schedules much more rapidly than other systems and that it may be used as a knowledge acquisition method. Most other intelligent systems use some type of search method to generate a feasible schedules and their computation time tends to increase exponentially with the size of the knowledge base. However, the proposed system uses a simulation method for schedule generation, requiring only linearly increased computation time as the size of the problem

increases. With respect to use of the system as a knowledge acquisition method, other intelligent expert systems require an intensive human interviewing process to elicit expertise and build the system knowledge base. The proper prototype system combination of computer graphic simulation modeling and interactive functions allows the operator to test many different situations while at the same time accumulating experience. This process eliminates the time-consuming human interview process, thereby facilitating the knowledge acquisition process.

2.0 FMS MODEL AND PRODUCTION CONTROL PROBLEMS

2.1 General System Design and Specifications

Guidelines for system design and specifications are described for the purpose of simulation model development. To make the simulation model as realistic as possible, these guidelines are based on published reports on related issues (Shanker & Tzen, 1985).

2.1.1 Size of FMS

In the publication, "Collection of European and American FMS," the Japanese Production Technology Investigation Society (1981) collected data on 79 existing FMS in the United States and Europe. Table 2.1 indicates how many of these FMS employ a given number of machines. Table 2.2, based upon a recent survey by Darrow (1986), indicates the worldwide distribution of FMS technology by region.

With these figures in view, four machines have been included in the simulation model implemented for this study. This number of machines should provide for sufficient interaction and combinational complexity for experimental purposes, while keeping the program simple enough to be practical.

Table 2.1 Number of FMS and Corresponding Number of Machines in Operation.

No. of Machines	No. of FMS	No. of Machines	No. of FMS
1	4	10	8
2	6	11	1
3	4	12	4
4	7	13	2
5	10	15	1
6	11	16	1
7	8	28	1
8	4	29	1
9	5	80	1

Table 2.2 Distribution of FMS Technology (based on number of systems and number of machine tools employed).

Region	Systems	Machine Tools
Eastern Europe	23	192
Western Europe	107	485
Japan	59	462
United States	64	330
Totals:	253	1,469

2.1.2 Job Arrival

There does not seem to have been any specific study undertaken to report on the distribution of interarrival times of jobs for FMS. From the study on job shops, exponential or Erlang distributions are mostly used as the candidate distribution system, partly due to its ease of application in mathematical models based upon queueing or network theory. However, in

real FMS situations, jobs are usually available at the beginning and are loaded into the system one by one, each with its own setup time.

2.1.3 Size of Batch

One of the major economic forces which has lead to the rapid implementation of small batch manufacturing FMS is the well-recognized need to reduce work-in-process inventories, while at the same time increasing machine utilization. In particular, this need has been pointed out in the area of the metalwork manufacture of lots of less than 50 pieces (Cook, 1975).

While reviewing several new FMS instituted in Japan in the early 1980s, Ito (1981) mentioned batch sizes ranging from 6 to 30. However, the batch size chosen for the model used in this study is assumed to be based on unit processing. Since setup time is significant in the FMS and the jobs in one batch are similar in nature and processed continuously, we can consider each batch as one job.

2.1.4 Number of Part Types

Jaikumar (1984) conducted a survey of 28 FMS and observed that 25 of them manufactured between 4 and 22 part types, while the remaining 3 systems each produced more than 100 different parts. Smith et al. (1986) surveyed 22 FMS operations in the U.S. and investigated several aspects of FMS characteristics, including the

Table 2.3 Number of Part Types Processed.

Part Types	Percent of Total Response
1-10	22
11-20	14
21-30	7
31-50	14
51-100	7
above 100	36

number of part types processed, batching of part types, system costs and scheduling criteria. Table 2.3 shows the results of the survey question for the number of part types processed. For the purposes of this study, 10 part types are considered insofar as it is believed that this number provides sufficient complexity for experimental purposes.

2.1.5 Number of Operations and Processing Time

There have been no specific reports of practical instances regarding the number of operations and processing times for FMS jobs. However, since very little direct labor time is required for the set-up and/or running of individual FMS machine tools, processing times are highly predictable and nearly deterministic. The exception is downtime, which occurs randomly. In the model developed for this study, the number of operations for each job is set up and is distributed from two to four. Processing time for each operation is

assumed to be constant for planning and scheduling purposes.

2.1.6 Transportation/Material Handling Systems

It has been demonstrated empirically that parts in conventional manufacturing systems spend about 95 percent of their time in a waiting state between operations (Carter, 1971). Parts-waiting time in a job shop is high because of the large queues and larger batches. For an FMS, transportation time is important and is partially dependent upon the utilization and speed of the material handling system. There can be capacity restrictions on the material handling system, i.e., a finite number of AGVs or tow-line carts. Conveyors may be less restrictive with respect to capacity, but are at the same time less flexible in terms of how they may be used.

Estimating the expected value of travel time is also an important factor. In the model used for this study, an average travel time of 10 minutes between two work stations was used since the focus of concern was with system planning and scheduling.

2.1.7 Buffer Limitations/Storage Capacity

FMS material handling facilities are automated, which often places limits on the amount of work-in-process of finished parts that can be held in the system. In-process inventory can be held either in

centralized storage, in buffers provided at the individual machines, or in an automated storage and retrieval system. Conventional job shop scheduling overlooks this aspect, assuming that all necessary storage is available. However, the issue of storage is often a critical constraint in FMS scheduling. Finite buffers or storage capacities can lead to blocking and system starvation, which must be considered in real-time FMS parts scheduling.

2.1.8 Scheduling Period

One manner of dealing with the scale and complexity of the scheduling problem is to decompose it into interacting levels or problem hierarchies, according to the scheduling period (Ammons, 1985; Suri & Whitney, 1984):

1) High or strategic level: production planning;
2) Intermediate or tactical level: release scheduling; and
3) Low or operational level: item movement.

High level production planning problems, including the tasks of part-mix changes, system modifications, and expansion, must be solved at regular intervals of relatively long duration, usually on the order of weeks. Intermediate release scheduling problems, including the division of production into batches, balancing the workload, and responding to changes in

production plans and material availability, must be solved daily or hourly, dependent upon the nature of upper level production planning decisions. The lowest level in the hierarchy makes moment-to-moment routing decisions for items within the production systems and is concerned with detailed decision-making requirements for real-time FMS operations, including the material handling system. The time horizon is typically a few minutes or hours, and the decisions involved are (1) part movement and material handling system, (2) tool management, (3) system monitoring and diagnostics, and (4) reacting to disruptions, including machine failures. In this study, a scheduling problem for a one-day period is considered since it is important to use short-term scheduling periods in an FMS with a dynamic environment in which frequent changes are possible.

2.1.9 Measure of System Performance

Job shop research uses various criteria to measure the performance of scheduling algorithms. For the performance measurements of an FMS, since most are capital intensive systems, throughput time, system output, meeting due dates, total processing time, or machine utilization are used. Of these, total processing time is taken as the criterion of system performance for the model.

2.1.10 Other Factors

There are also limited numbers of pallets and fixtures of different types in an FMS. These resources can affect both planning and scheduling problems and must also be considered.

2.2 FMS Production Control Problems

The aim of an FMS is to achieve efficient and automated high-volume mass production while retaining the flexibility of a manual job shop to simultaneously machine several part types. Because the concept and technology of automated manufacturing are still in their infancy, problems have been encountered in planning and controlling the systems. Managing production for an FMS is more difficult than production line or job shop management because:

1) Each machine is versatile and capable of performing many different operations;
2) The system can machine several part types simultaneously; and
3) Each part may have alternative routes through the system.

These additional capabilities and planning options increase both the number of decision variables and constraints associated with setting up an FMS. To best

utilize the capabilities of an FMS, careful set-up is required prior to production since set-up decisions must incorporate options which reflect new or altered production requirements. This contrasts with a mass production system where set-up is part of the design process and few changes occur following implementation.

Stecke (1983) identified five inter-related production control problems which must be solved prior to system operation:

1) Part type selection problem: From a set of part types that have production requirements, determine a subset for immediate and simultaneous processing.
2) Machine grouping problem: Partition the machines into groups in a way that each machine in a particular group is able to perform the same set of operations.
3) Production ratios problem: Determine the relative ratios at which the part types selected in problem (1) will be produced.
4) Resource allocation problem: Allocate the limited number of pallets and fixtures for each type among the selected part types.
5) Loading/scheduling problem: Allocate the operations and required tools for each selected part type among the machine groups, subject to

technological and capacity constraints of the FMS.

Assuming the solution of part type selection, machine grouping, production ratios, and resource allocation problems, loading and scheduling problems are considered, including selection of a subset of jobs from the job pool and, in the ensuing planning period, assigning these subsets to appropriate machines to achieve production objectives while meeting system constraints. Given the capital intensive nature of FMSs, a sound return on investment can be achieved only when productivity is maximized and idle time is minimized. A more detailed treatment of this problem is included in Chapter 3.

2.3 FMS Model Example

An FMS simulation model was developed to provide the means to study control and scheduling problems. The model consists of four machining centers (Figure 2.1), each with an input buffer for parts setup and queueing. Use of an automatic transportation system has been assumed in the model so that parts can be moved freely from one machining center to another.

The model includes 10 part types for processing; each part is designated by a part identification

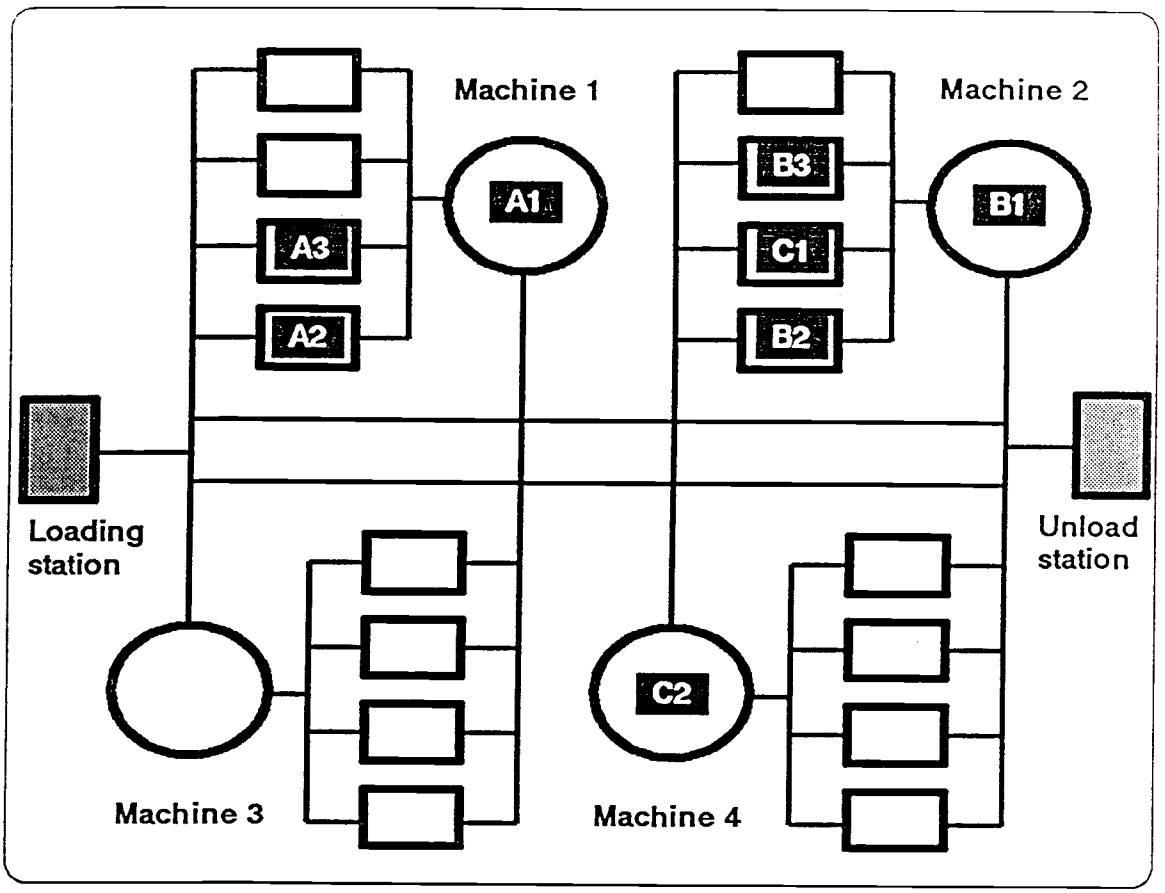


Figure 2.1. FMS model.

symbol, e.g., A1, A2,...J1, J2, J3, and is categorized according to part type, in which parts in each category are assumed to be identical. The letter in the identification symbol indicates the part type, while the number following the letter is used to identify the specific part within the type category. Note that the system's status at the point in time represented by Figure 2.1 indicates that parts A1, B1, and C2 are being processed at machining centers #1, #2, and #4, respectively, and parts A2 and A3 are waiting in the input buffer at machining center #1, while parts B3, C1, and C2 are waiting in the input buffer at machining center #2.

The FMS model also includes loading and unloading stations, each with a holding capacity of one part. It is assumed that it takes 10 minutes to load, set up, and transfer each part from load station to machining center or from one machining center to another. Table 2.4 shows the job types and their processing time information. For example, the sequence for part A is machining center #1 → #2 → #3 → #4, the operations for which take, respectively, 90, 60, 60, and 120 minutes.

2.4 Assumptions

The scheduling problem to be addressed in this study may be defined as follows:

Table 2.4 Job Type and Processing Time Information.

Part Type	Operation Sequence (machining center no.)	Processing Time (minutes)
A	1, 2, 3, 4	90, 60, 60, 120
B	2, 4	90, 60
C	4, 2, 3	30, 60, 90
D	3, 4, 1, 2	120, 90, 60, 60
E	2, 1	60, 90
F	3, 1, 4	120, 90, 60
G	4, 3, 2, 1	120, 90, 60, 90
H	1, 4, 2	60, 60, 60
I	2, 4, 1	60, 90, 120
J	1, 3, 2, 4	60, 90, 120, 150

1) The scheduling objectives are to process all jobs and to minimize the total processing time.
2) Although the job is composed of distinct operations, no two operations for the same job may be processed simultaneously.
3) Each operation, once started, must be completed before another operation may be started on that machine.
4) Each job must be processed to completion.
5) There is no randomness. In particular,
   a) the number of jobs is known and fixed, and
   b) the processing times are known and fixed.
6) A job represents the processing of one part type and a job order represents a set of jobs.

7) A loading sequence represents the system parts loading priorities in the job order.

8) The scheduling of transportation systems is not considered since they are always available.

3.0 EXISTING APPROACHES TO FMS SCHEDULING

In this chapter existing approaches to FMS scheduling are described and their advantages and disadvantages are discussed. More detailed discussion of various aspects of FMS, as well as scheduling procedures, appears in Rachamadugu and Stecke (1986) and Buzacott and Yao (1986). The purpose of this section is two-fold: first, to review the work on FMS modeling, with particular focus on scheduling problems, and, second, to examine some of the implications of an FMS, including the new and challenging problems the FMS poses to production planning and control. These are problems which must be solved in order to encourage the future adoption and effective use of the FMS.

3.1 Characteristics of FMS Scheduling

FMS operational control is directed at the effective scheduling of parts. It is exercised at the intermediate level of detailed FMS decision-making and is part of day-to-day system operation. To a greater degree than a conventional job shop manufacturing system,

an FMS must have short-term scheduling and planning facilities because of the variability and dynamic changes which take place within the system.

Although there are a significant differences between an FMS and a conventional job shop, FMS control is conceptually similar to job shop manufacturing and may be regarded as a special case of standard job shop scheduling. In essence, an FMS is a job shop system with an automatic material handling system.

There are literally hundreds of published studies concerned with scheduling and in addition thousands of individual firms have generated their own scheduling techniques for which there is no published knowledge. Baker (1974) and Conway, Maxwell, and Miller (1967) have provided excellent surveys of the entire scheduling topic, while Day and Hottenstein (1970) and Graves (1979) have provided surveys of the available scheduling techniques.

A job shop scheduling problem (see Chapter 1) is an NP-complete, hard problem, meaning that no polynomially-bound solutions exist (RinnooyKan, 1976). In many job shop manufacturing systems, including the FMS, the preferred sequence of operations on one job may be a function of the sequence of operations chosen for one or more other jobs. In this case, in order to determine the preferred job sequence, it is generally necessary to determine the preferred sequence for all jobs

simultaneously. As a result, the sequencing problem can become one of considerable size and complexity. For example, in a hypothetical case of N jobs and M machines, the number of possible solutions is $(N!)^M$. If $N = M = 5$, then $(5!)^5 = 24,883,200,000$. Even with all of the processing power available today, it is not practical to search exhaustively through a solution set of this size. The problem of discovering preferred sequences in the larger and more complex setting of the typical factory is clearly even more difficult.

Since an FMS is an extremely complex manufacturing system, interconnecting many components, it is unlikely that one single procedure could be developed that would encompass most of the possible situations. Therefore, several approaches are analyzed in the following sections of this chapter.

3.2 Analytical Methods

Various analytical techniques have been used to model the job shop, surveys of which may be found in Conway et al. (1967) and in Graves (1979). Detailed algorithms are provided in Johnson and Montgomery (1974). The job shop has been viewed as sets of jobs and machines, e.g., 2-jobs/5-machines. However, the analytical results have been restricted to simple problems with not more than 10 tasks and 10 machines. One

of the earliest and most popular approaches to the scheduling problem was developed by Henry L. Gantt. This graphic approach, the Gantt chart, is a bar chart which displays job operations over time for each machine. The main drawback of the chart is its dependence on the scheduler's intuition in the improvement of the scheduling process.

In 1956, Jackson extended Johnson's (1954) algorithm with minimum makespan as the criterion, solving the n-jobs/2-machines flow shop scheduling problem. Later, operations research techniques, including integer linear programming, dynamic programming, and branch-and-bound technique, were used to obtain optimal schedules. Jackson's (1957) Decomposition Principle can be regarded as one of the most significant analytical studies in the area of scheduling. Modeling the shop as a network of queues, this principle is used to establish sufficient conditions for decomposition of the network into a set of independent queues. However, as Conway et al. (1967) have mentioned, this technique was not useful in the solution of large scale problems.

3.2.1 Queueing Theory

Since a job shop can be viewed as a network of queues, queueing theory techniques have been widely used in studies of job shops. Unfortunately, since jobs pass through the network in a large variety of

patterns, the use of this method for scheduling is extremely complex for even small job shops. The basic assumptions of the model are as follows:

1) The interarrival time and processing time are exponentially distributed and parts are processed with first-come-first-served queue discipline;
2) The jobs are routed to a machine by a fixed probability transition matrix; and
3) The capacity of the in-process storage space is implicitly bound by specifying a finite number of parts loaded into the system and is assumed to be sufficient to accommodate all jobs in the system in order to avoid blocking.

Based on these assumption, a closed queueing network model, CAN-Q was developed by Solberg (1977). It is a closed queueing network in the sense that the number of parts circulating in a system is at all times constant. This model has been widely used for the preliminary design of FMSs and for the study of some production planning issues in which exact details are not available and in which tolerances for error are not critical. For example, the effects of routing policies on the throughput and in-process inventory of an FMS has been examined in a number of investigations (Solberg 1981; Buzacott & Shanthikumar, 1980). As a design tool, the queueing model is suitable for a "first-pass"

when details about the system are either unavailable or lack clarity. However, when a system description is available, more precise methods of analysis, including computer simulation, are preferred.

3.2.2 Mathematical Programming

The job shop scheduling problem has been formulated as a mathematical program for over 20 years and many investigators have applied integer programming techniques to the problem. Stecke and Solberg (1981) and Stecke (1983) have modeled scheduling problems for a real FMS as an integer program. Several alternative objective functions have been proposed since no single set of objective functions can be applied universally to different production situations. For example, Kusiak (1985) formulated an algorithm for FMS loading problems, and Shanker and Tzen (1985) proposed an integer programming formulation of the operation allocation problem. In the following section, a mixed integer program formulation is presented, along with example problem results.

3.2.3 Mixed Integer Program Formulation

Several attempts were made to solve the job shop scheduling problem with the objective function of minimizing total processing time, mostly through use of an integer program using 0 to 1 variables as the main variables. In these attempts, efficiency differed

according to the method used to express the constraint that each machine is limited to processing one job at a time. Historically, three formulations for the use of the cutting plane method (Bowman, 1959; Manne, 1960; Wager, 1959) have been developed and, in particular, Manne's formulation has a comparatively small number of variables and constraints. Integer programming formulation based upon Manne's method uses the following notation:

T_{ik} = Variable indicating the time at which task i is to begin at machine k.

A_{ik} = Processing time for job i at machine, k.

B = Arbitrary big number for penalty.

Y_{ijk} = Integer variable, either 0 or 1.

D_i = Due time for job i.

The constraint equations and objective function are as follows.

3.2.3.1 Noninterference Restrictions

Given that jobs i and j require, respectively, A_{ik} and A_{jk} processing time units at machine k, if they are to be prevented from occupying the same machine simultaneously, one of the two must precede the other by a time period sufficient to allow completion of one job before the second can begin, i.e., either

$$T_{ik} - T_{jk} \geq A_{jk} \text{ , or}$$

$$T_{jk} - T_{ik} \geq A_{ik} \qquad (1)$$

In order to convert this condition into a linear inequality, it is convenient to define a new integer-valued variable, Y_{ijk}, with the following written restrictions:

$$(B + A_{jk})Y_{ijk} + (T_{ik} - T_{jk}) \geq A_{jk} , \quad (2)$$

$$(B + A_{ik})(1 - Y_{ijk}) + (T_{jk} - T_{ik}) \geq A_{ik} , \quad (3)$$

where B is a large number, i.e., $B >> |Tik - Tjk|$, and

$Y_{ijk} = 0$, if job j precedes job i at machine k, or

$Y_{ijk} = 1$, if job i precedes job j at machine k.

Equations (2) and (3) ensure that the first job will be initiated in sufficient time to be completed before the beginning of the second job.

3.2.3.2 Sequencing Restrictions

Once the noninterference stipulations have been written, the remainder of the formulation automatically follows. If the operation at machine k for job i is to precede the operation at machine l, this means that the operation at machine l will be performed at least A_{ik} time units later than the operation starting time at machine k. This condition becomes:

$$T_{ik} + A_{ik} \leq T_{il} \quad (4)$$

3.2.3.3 Specific Delivery Requirement

If operations at machine p for job i is the last operation which the shop is to perform on the item, and the specified due time of the item is D_i, then this requirement may be written

$$T_{ip} + A_{ip} \leq D_i \quad (5)$$

3.2.3.4 Overall Delivery Requirement

Minimizing the total processing time is employed as the performance measurement. If the total processing time is denoted by F_{max}, the problem consists of the minimization of F_{max} with respect to the nonnegative integers, T_{ik} and Y_{ijk}, subject to constraint equations (2) through (5) and

$$T_{ik} + A_{ik} \leq F_{max} \qquad (i = 1, 2, \ldots n \text{ and } k = 1, 2, \ldots m) . \qquad (6)$$

3.2.4 Computation Results

The mixed integer program, based on the example job order (A 1 B 1 C 1 D 1) for the FMS model presented in Chapter 2, ws formulated and solved using a branch and bound algorithm in a multi-purpose optimization system (MPOS) on the CYBER 720 mainframe computer. The MPOS program code is shown in Appendix A and the program output is shown in Appendix B. A Gantt chart of the summary schedule is shown in Figure 3.1.

However, even for small-sized problems the number of constraints and/or variables becomes very large in any integer linear program formulation and this method is not very effective. Mathematical programming fails to provide real-time control for even small-sized problems, due to excessive computational requirements. In some instances, mathematical programming has been

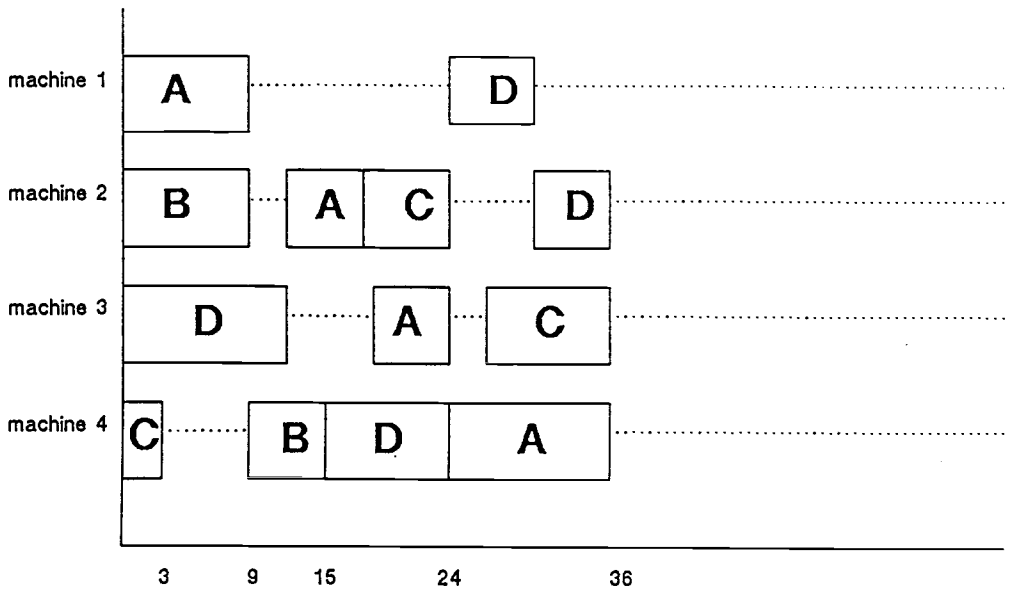


Figure 4. Gantt chart schedule for (A 1 B 1 C 1 D 1)

Figure 3.1 Optimum Gantt chart schedule for the example.

applied with some success to aggregate, long-term planning problems.

Moreover, optimization procedures, including integer programming, have not been used successfully with problems of practical size due to their NP completeness and formulation difficulty in the problem. Davis (1973, p. 306) has noted that,

> even if more efficient integer linear program algorithms were available, the application of the approach might not be feasible because the task of writing the objective function and constraint equations is in itself a formidable one.

This simple example illustrates the complexity of the mathematical programming approach. A comparison was made with the results of the proposed expert system approach in Appendix C. It is normally insufficient to draw any conclusion based on one single comparison, but it is possible to offer a number of observations drawn from the comparison. Since an 8-job example was one of the trials used for the model, the same example job was tried for formulation and solution of the problem using a mixed integer program. The CYBER 720 mainframe computer could not handle the problem due to core memory limitations, calculated as follows for a mixed integer program:

$$IntVar * ((M + 2) * (N + 1) + M + 1)$$

where IntVar is the number of integer variables, M is the number of constraints, and N is the number of total variables, including integer variables. In the case of

the 8-job example, the number of constraints is 260 and the number of variables and integer variables are, respectively, 257 and 224. The total core memory requirement, therefore, is 16,319,968 bytes, a requirement which exceedsn the CYBER 720 core memory of 640 Kbytes. Furthermore, even if the core memory were available, excessive amounts of computation time would be required to solve the problem. Consequently, a simpler 4-job example was solved and compared with the results of the proposed expert system approach.

This complexity has motivated a number of investigations based upon polynomial time and space methods and sub-optimal solutions have been obtained. These heuristics are a more practical approach to real-size resource scheduling problems.

3.3 Heuristic Methods

A number of studies for developing proper heuristic rules have been completed for the control and operation of FMS and the job-shop manufacturing environment. As discussed in the previous section, FMS scheduling problems can be formulated analytically as 0-1 integer programming problems, but the size of the resulting problem is too complex to solve day-to-day scheduling problems and uncertainties in production, such as machine breakdown, preclude the use of these

methods. Therefore, most FMS control systems use heuristic rules to schedule production.

Heuristic methods can involve either priority rules or heuristics, or a combination of the two. Priority rules based upon selective criterion are used to assign numerical attributes to each waiting job and to determine the processing order at a machine, e.g., the first-come-first serve rule, the shortest processing time rule, the earliest due-date rule, or the most-work-remaining rule. "Heuristics" in this sense means "rule of thumb," a meaning which has evolved as heuristic programming has been applied to management problems. Heuristics refer to methods justified for reasons of their internal logic and because they have been empirically tested and found to perform reasonably well. Some examples of heuristics programming include (1) alternate operations, (2) look ahead, (3) insert operations, and (4) exchange operations (Gere, 1966).

In this section, the work of Stecke and Solberg (1981) and Denzler and Boe (1987) on real-time systems is reviewed. A more detailed survey of heuristic rules may be found in Blackstone et al. (1982). Panwalker and Iskander (1977) have comprehensively surveyed scheduling rules, including a summary of 113 dispatching rules derived from a cross-index of 36 technical publications in the field. Performance criteria and the rules developed for each of the referenced

investigations, including treatments of their heuristic dispatching rules and the manufacturing environments for which they have been tested, are included.

In exploring alternative FMS loading and control strategies, Stecke and Solberg (1981) examined the FMS at Caterpillar Tractor Company, which consists of four 5-axis machining centers, three 4-axis machining centers, two vertical turret lathes and an inspection machine. Each machine has a limited capacity tool magazine and two straight-track transporters, which carry parts from one center to another. The 16-station load/unload area also provides a queueing area for in-process inventory (see Figure 3.2). The system was designed to machine four prismatic parts for automatic transmission housings. In scheduling parts, each operation was assigned to an individual machine tooled to perform that task, thereby creating fixed routing for each part. In their design tests of loading and control decision rules, alternate routings were created by assuming increased tool flexibility.

Using a simulation model, 16 loading and control decision rules were tested under 5 different possible sets of manufacturing conditions. Results indicated that a parts loading rule based upon first loading the part with the shortest processing time/total processing time ratio, produced results superior to those of other decision rules tested under all 5 conditions. These

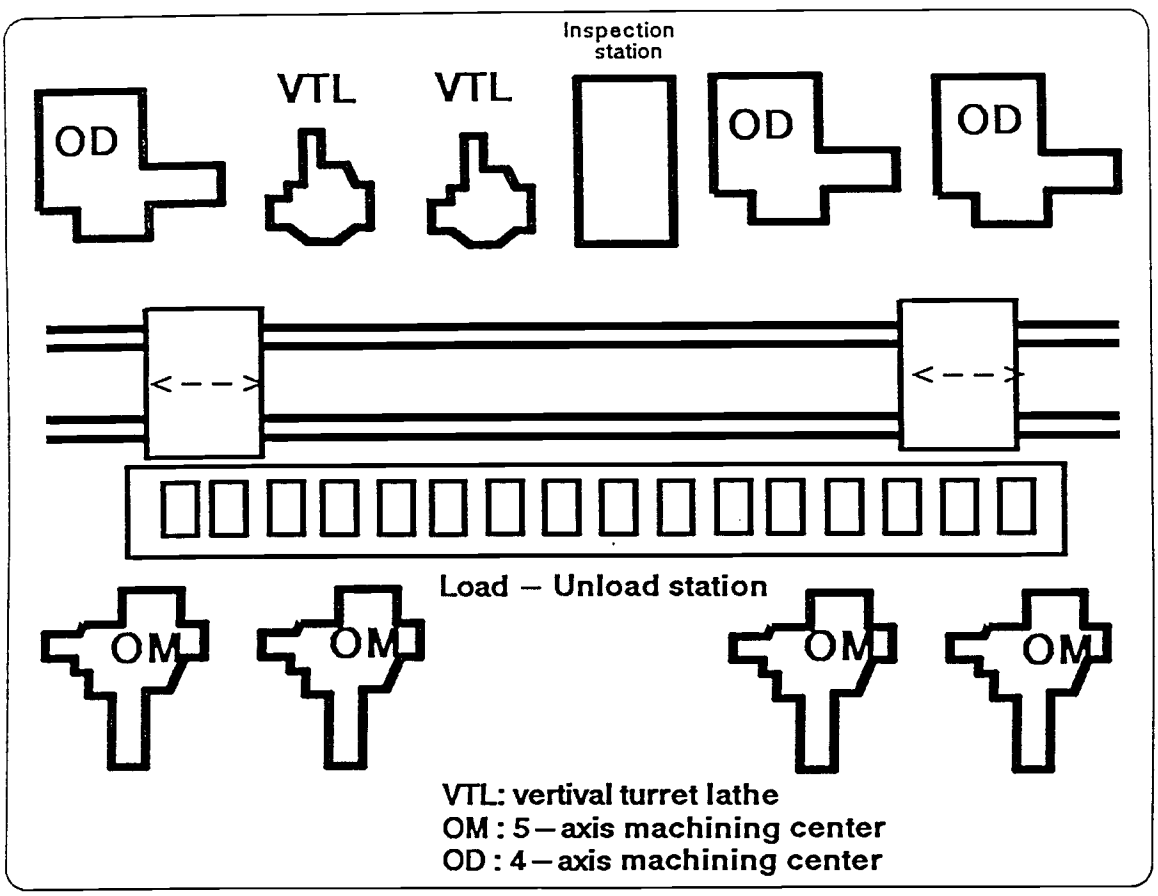


Figure 3.2 Layout of Caterpillar Tractor FMS (from Stecke & Solberg, 1981).

results were from 8 to 24 percent better than those obtained when using the plant's existing scheduling rule. Although these results appear to indicate a significant production improvement, what is not clear is the extent to which further improvement would have been possible. The study did not report on attempts to develop analytic solutions using integer programming, and the applicability of these findings to FMS based upon different design characteristics is unknown.

Denzler and Boe (1987) investigated the effectiveness of heuristic parts scheduling rules used to operate another real-time FMS, a system comprised of 5 indexers, 11 moduline machining stations, 2 loading stations, and 2 unloading stations and directed at the manufacture of 8 prismatic automatic transmission housings. A schematic illustration of the system is shown in Figure 3.3.

The system has 37 carts and 51 pallet fixtures and each of its machines has the capacity to store two parts prior to and two parts following processing on a machine shuttle system. Once placed on a machining center, the parts are processed on a first-come-first-serve basis. There is no common work-in-process inventory storage capacity within the FMS, other than that obtained by cycling a part around the system on a cart.

Load Station			Unload Station		Unload Station			Load Station	
D	C	Machine 1	B	A	A	B	Machine 9	C	D
D	C	Machine 2	B	A	A	B	Machine 10	C	D
D	C	Machine 3	B	A	D	C	Machine 11	B	A
A	B	Machine 4	C	D	D	C	Machine 12	B	A
A	B	Machine 5	C	D	D	C	Machine 13	B	A
A	B	Machine 6	C	D	D	C	Machine 14	B	A
A	B	Machine 7	C	D	D	C	Machine 15	B	A
A	B	Machine 8	C	D	D	C	Machine 16	B	A

Note: Machines 1,2,3,9 and 10 are head indexes. All other machines are moduline machining centers. A and B are upstream inventory holding positions. C and D are downstream inventory holding positions

Figure 3.3 Schematic of a dual loop FMS (from Denzler & Boe, 1987).

The scheduling of all parts, carts, and machines is controlled by a software package based upon three decision rules: (1) A parts load rule directing the loading of the part onto a pallet; (2) A parts launch rule directing the loaded pallet to the first available machine; and (3) A parts routing rule which determines which available machine can perform the next operation. Six loading rules were tested through simulation experiments, comparing the performance of alternative scheduling heuristic rules under conditions of high and low flexibility. Results indicated that the "smallest proportion of job launched" rule, which loads the part with the smallest proportion of its batch among parts in the system, and the "reload with alike product" rule performed slightly better than others when a sufficient number of empty pallets were available. The effectiveness of the parts loading rules were also analyzed, comparing their performance with the upper bound machine utilization rate achieved by the analytical model. The rules developed as outlined above were able to schedule the FMS model at a machine utilization rate within 20 percent of the theoretical upper bound.

These results clearly indicate that the use of heuristic rules can result in schedules within 20 percent of the upper bound machine rate of utilization. However, it should be noted that the parts loading rules Stecke and Solberg (1981) reported as the best

performers did not perform as well as the rules examined by Denzler and Boe (1987). Since the two systems are configured in significantly different ways and the performance of heuristic rules is highly system dependent, the differing results indicate that each system should be individually studied by means of simulation for various loading and real-time control strategies in order to determine which system offers the best results. Even though heuristic methods are generally found to be useful and relatively economical in their utilization of computational resources, compared to the use of optimization techniques, care must be taken because rules which appear to bear positive results can at the same time lead to unexpected performance deterioration.

3.4 Computer-Aided Methods

Recent research concerning the impact of computers upon managerial decision-making have focused attention on computer-based information systems, developing significant insights that have gained public attention. One of the areas of inquiry, based upon software developed for use on personal computers, has been directed at computer-aided scheduling methods as follows:

1) Simulation studies (El Maraghy, 1982; Steudel, 1986);

2) Decision support systems for manufacturing control (Chen & Talavage, 1982; Suri & Whitney, 1984); and

3) Development of interactive scheduling methods (Godin, 1978; Hurrion, 1978, 1980; Hodgson & McDonald, 1981; Jones & Maxwell, 1985).

The principal contributions of these approaches is that they allow the operator to quickly and easily review and revise a number of schedules. As a result, scheduling control of dynamic production systems can become more responsive and effective.

3.4.1 Simulation Studies

To accomplish the complex quantitative analysis and the interaction of production and production control procedures, including their interaction, a high level of knowledge of the system's dynamic behavior is required. This can be obtained either through trial and error implementation or by simulation. It is generally accepted that the control of any process depends upon the availability of a model of the process that permits projection of the results of a contemplated action before it is taken. For purposes of FMS control and scheduling, the characteristics and operation of individual components making up complex and dynamic manufacturing systems are relatively well known, but there is little data available on the integrated system

operation of these components. Thus, simulation would appear to be a logical option under all but the most unusual circumstances.

Simulation models have already played an important role in the development of integrated manufacturing systems and in the organizational and technological design of FMS. FMS simulation has been used mostly to verify design concepts, evaluate alternative configurations, assist in selecting machinery and material handling hardware, and to test system control strategies. The quick feedback provided by the simulation of a given design enables the operator to measure the merits and shortcomings of a system, make the necessary revisions, and arrive at efficient decisions. What is most clear is that the role of simulation in designing and evaluating future factories will increase in importance as more companies become involved in implementing advanced manufacturing technologies (Ballakur & Steudel, 1984; El Maraghy, 1982; Steudel, 1986). At present, over 100 simulation languages have been developed and used for modeling manufacturing systems, including TESS/SLAM-II, CINEMA/SIMAN, GPSS, GASP-IV, SIMSCRIPT II.5, PCModel, SEE WHY, and XCELL. For a comparative discussion of simulation languages, see Shannon and Phillips (1983)

In the simulation of manufacturing systems, it is usually necessary to account for every individual

entity in the system and arrange for its status changes. The result is known as a discrete event simulation, a simulation in which system status changes only at specified times, remaining at the new values for subsequent time units. The length of the time unit may be as brief as necessary in order to capture additional details of system operation, but this should be done only at the expense of longer computer run times.

Scheduling problems can be solved all at once, as done with the use of analytical methods, or they can be solved sequentially, over time. In the latter case, logical rules ("dispatching rules") are used to select the next job for processing when a machine becomes available. Most studies based upon the use of dispatching rules have employed simulation, rather than analytical techniques.

Simulation studies provide greater insight into manufacturing systems and offer the flexibility necessary to allow the evaluation of different decision-making alternatives. But this is accomplished at the expense of additional modeling complexity since manufacturing simulations usually require tailoring to the specific system and, for full scale simulation models, they are expensive and time consuming to build and run. Therefore, a simulation should be performed only when the solution of a problem is more difficult to obtain by use of a conventional analytical method. Simulation

is not an exact method and its utility is apparent only when all of the prerequisites, assumptions, constraints, and possible sources of errors are taken into consideration. The most important advantage of simulation lies in the almost unlimited variations of real processes which can be investigated when process behaviors must be known. This can be done independently of the process and without disturbing it and in most cases, other alternatives may be investigated with little additional effort.

However, the operator should bear in mind that simulations are statistical in nature and results should be treated with due care since they are subject to misinterpretation. Since it is possible to output the value of a parameter with a high degree of accuracy, the inexperienced user may obtain a false sense of accuracy when interpreting the results. Problems may also arise in cases where models are graphically displayed. For example, a user may view the travel of parts on an output screen, observing queue information, the processing of parts at machine stations, and the interaction of system components. However, assumptions concerning travel time speed and processing time may falsify the entire simulation run.

3.4.2 Decision Support Systems (DSS)

While several advanced FMSs have been installed around the world, the capacity of these systems has been underutilized. This has occurred because the efficient operation of an FMS can be very difficult for even the most experienced shop floor supervisor. Due to the complexity of FMS structures, the task of FMS production control cannot rely singularly upon human efforts. In view of the capital intensive nature of the FMS, it is imperative that they be designed to operate as efficiently as possible. Thus, for further applications of an FMS, the concept of computer-based decision support systems (DSS) has been suggested to provide production managers with a basic planning and operational framework (Chen & Talavage, 1982).

The DSS is a tool which helps the FMS operator evaluate the consequences of various alternative actions, primarily at the level of day-to-day operations. However, the role of the DSS can be more broadly conceived, applying it to all organizational levels that interact with the FMS and to the investigation of the utility of the FMS for other operations in the organization. The DSS can assist with both short-term and long-term decisions, offering not only an evaluative function but also a generative function. Given its capacity as a powerful computational system, the DSS can

propose alternative actions, selecting the best option for creation of a recommended course of action. In summary, the DSS provides control capability for structured problems which are anticipated and occur repeatedly, as well as decision-making support processes for unstructured problems which cannot be anticipated and which arise only occasionally.

Suri and Whitney (1984) have analyzed a simplified DSS, illustrating its basic DSS framework and showing how a DSS can assist the decision-maker in controlling production processes in an FMS machine shop. The concepts and issues underlying the DDS were based on experience gained during the installation of FLEXPLAN, a DSS installed with an FMS at Hughes Aircraft Company. Chen and Talavage (1982) have presented the development of the basic DSS logic structure required for an FMS facility. Figure 3.4 shows the interactions between the operator and the DSS. The entire process may invoke the following steps:

1) The operator identifies the problem after receiving alert information;
2) The operator consults the relevant data on system status information and establishes feasible alternatives to solve the problem;
3) Through access to the models available in a DSS, and his own experience, the operator evaluates the alternatives; and

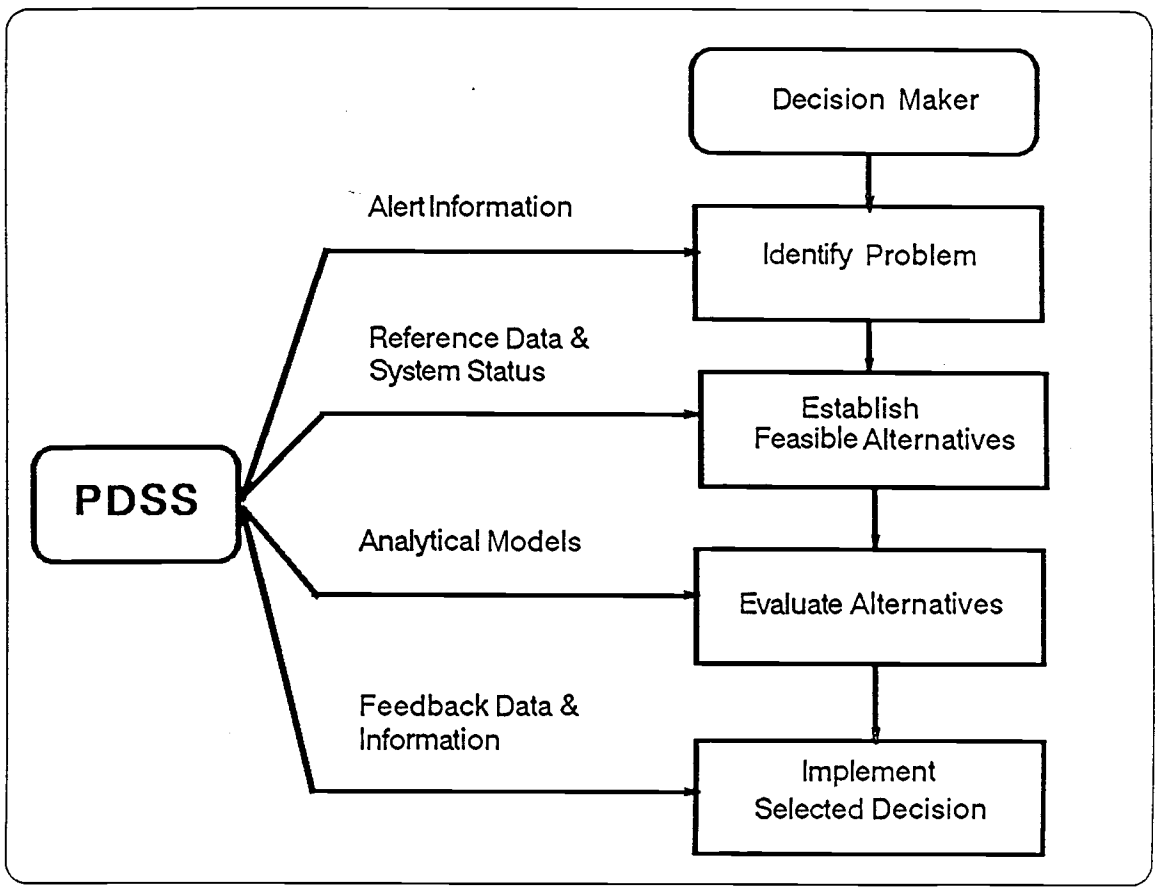


Figure 3.4 Interaction process between an operator and DSS.

4) Through the DSS control, the selected alternative is implemented on the system and feedback information and data are generated and reported.

Many of the ideas and algorithms included in a DSS are not new and the question arises of why they have not lead to a greater number of practical implementations. The answer would seem to be that in the past data bases were not developed to the extent that they could easily exercise control of manufacturing data. Additionally, computational power was not available in the manufacturing environment where it was needed. At present, both of these restrictions have been mitigated: manufacturing data is being captured and organized and computational power, nearly unlimited in extent, is available many forms, including those adaptable to the shop floor. This is a significant change of approach from earlier years. Since understanding human decision processes has proven to be much more difficult than originally believed, the goal is no longer to replace human thought processes, but rather to supplement them. With this constraint in view, it is feasible to give practical consideration to computerized solutions for large-scale manufacturing processes, based upon the pace of the current development of hardware and software technology.

Although the subject of scheduling is not the major focus of a DSS, the more widespread application of DSS concepts will ultimately be of benefit in the scheduling area. Since many scheduling problems are the type of complex, unstructured decision-making situations which can be significantly aided by a DSS, it is reasonable to hypothesize that improved DSS design and increased DSS utilization in the future will certainly include more scheduling applications.

3.4.3 Interactive Methods

The age of interactive man-computer problem solving systems commenced in 1960 with Licklider's, "Man-Computer Symbiosis." It was hypothesized that a symbiotic relationship between humans and computers would lead to greater problem solving capabilities than could either humans or machines left to rely solely upon their own resources.

Problem solving based upon an interactive system has been widely used in several design areas, such as electronic circuits and chemical processes. In the area of scheduling, however, relatively little work has been done. The progress that has taken place is outlined below.

Ferguson and Jones (1969) developed an interactive scheduling system for a job shop of six machines. System interaction took place on a teletype or other

similar type of hard-copy terminal. By selecting the appropriate values of a number of operating parameters, the operator had the option of arriving at decisions on his own or of having the computer provide an option.

An interactive job shop model developed by Holloway and Nelson (1973) also used a hard-copy terminal to permit the scheduler to monitor the progress of the computer as it attempted to set schedules based upon heuristic rules. If the computer had trouble finding a useful solution, the scheduler was allowed to intercede, pointing it in a new direction by altering the system state and then return control to the scheduling heuristics.

Godin and Jones (1969) developed an operational interactive scheduling system for Western Electric in North Andover, Massachusetts. A wire-winding shop supervisor was provided with access to a current and complete shop data base, as well as deterministic simulation capabilities with which to explore scheduling alternatives. A dedicated IBM 360/50 computer was utilized since there were no time-sharing facilities at that time and for this reason the computer was not always available for use. Furthermore, utilizing the system console as the interactive device proved awkward and after running for less than a year, the system was discontinued.

These systems are noteworthy only because of the symbiotic nature of the man-machine relationship. The machine raced through the large numbers of computations implicit in the embedded scheduling heuristics, but when it failed to find useful solutions, a human operator was at hand to provide assistance. However, these systems were not successful in real-time implementations. System hardware was not that advanced and the costs associated with a graphic and interactive system were at the time prohibitive.

Hurrion (1978) investigated visually interactive simulation models for job shop scheduling problems and carried out experimentation on four different sizes of problems. An example of his graphic display is shown in Figure 3.5, followed by a summary of the results for the four different problem sizes in Table 3.1. The results show that the visually interactive approach constituted an improvement to the problems of job shop scheduling, but as may seen in Figure 3.5, the graphic capabilities of the system were still poor. Moreover, the use of a keyboard for interactive functions was far from an ideal solution to the problem.

Finally, in 1985 Jones and Maxwell developed an interactive factory scheduling system using computer graphics. The system, written in FORTRAN 77, was implemented on a VAX 11-780 computer; graphical output was provided by a color display; and a digitizing

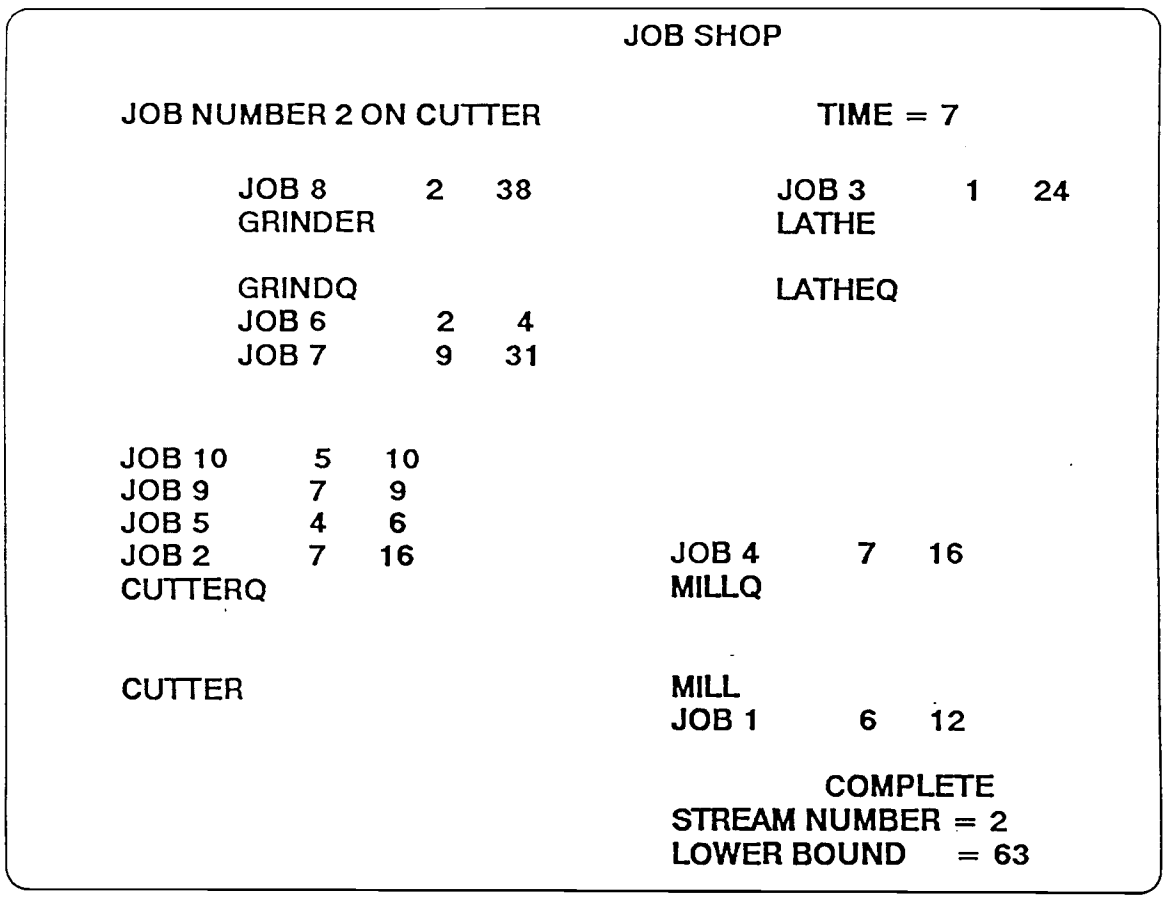


Figure 3.5 Interactive simulation model (from Hurrion, 1978).

Table 3.1 Results of Hurrion's Model.

Problem Jobs/Mach.	No. of Problem	% above L.B. Batch	% above L.B. Interact.	Improved %
10/4	10	18	3	15
20/4	10	3	0	3
20/6	10	14	4	10
25/10	10	9	4	5

tablet was used for input. Figure 3.6 shows the graphic representation of the system and interactive menu selection items. In this instance, the computer graphics involved not only pictorial output, but a more practical and highly interactive means of input. In particular, the user interaction for the implementation is provided through use of a digitizing tablet and on-screen menus. The right hand side of the display in Figure 3.6 displays menus for the model and current schedule. The system was tested by over 200 students at Cornell University, indicating that it could be used to create reasonably complex models (e.g., 25 processes, 25 inventories, and 10 machines) and, within an hour, an initial feasible schedule. However, this system does not use optimization algorithms to generate schedules and for implementation the operator must input schedules by hand.

The use of man-computer interactive methods have been principally employed in complex and large design problems for engineering applications, where the financial costs may be more easily justified. However, the continuing reduction in the cost and size of computers and computer display terminals makes this type of approach feasible for the management of manufacturing job shop scheduling. Using a visual display unit or a computer graphics terminal enables the status of a model to be displayed in the most convenient form for the

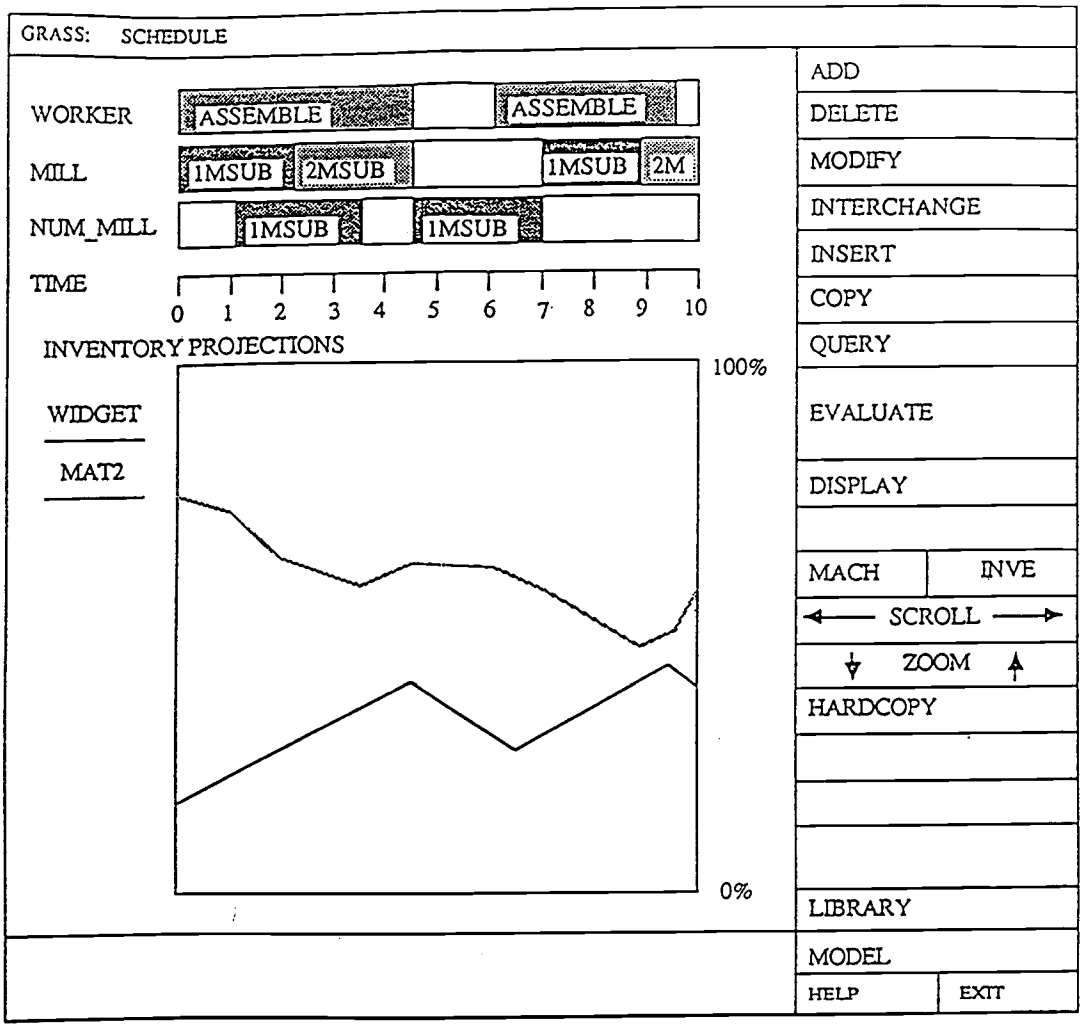


Figure 3.6 Interactive factory scheduling system (from Jones & Maxwell, 1985).

operator. At present, however, the value of graphics representation for the components of an interactive scheduling system still remain to be empirically tested.

4.0 EXPERT SYSTEM APPROACH

Management scientists and operations researchers are faced with the challenge of developing and implementing new methodologies to deal with the rapidly emerging technology of automated manufacturing systems. Even though considerable effort has been expended during the past several decades to develop scheduling and sequencing algorithms which will aid in process flow through the manufacturing facility, available scheduling methodologies are not adequate for integrated computer-aided manufacturing systems. Goldhar and Jelinek (1985) have pointed out that most of the traditional management science and operations research techniques are irrelevant for the factory of the future. Mathematical solutions tend to become intractable, and the use of heuristics is fraught with problems. For example, the selection of the appropriate heuristic is not only a difficult decision, but it is also system dependent.

The expert system has emerged as a new tool for the design and operation of manufacturing systems. As one of the aspects of artificial intelligence (AI), the

expert system has achieved considerable success in recent years. Though most of the systems initially developed belonged to the medical diagnostic realm, e.g., MYCIN (Shortliffe, 1976), PUFF (Kunz, 1978), and INTERNIST (Pople, 1977), interest has in recent years expanded into other areas, including mineral explorations (PROSPECTOR, Duda et al. 1978), computer configuration (R1 or XCON, McDermott, 1980), and management decision-making, planning and control (ISIS, Fox, 1983).

In the following sections, an overview of expert systems and the process of building them is provided, followed by an analysis of the benefits and problems of applying an expert system to FMS scheduling and its development procedure.

4.1 Overview of Expert Systems

An expert system is a computer program which contains the expertise required to solve a specific, domain-related problem. The expertise to solve a problem depends on the available knowledge. Hayes-Roth et al. (1983) define knowledge as "public" and "private," which indicate, respectively, published literature or individual expertise. The ability of an individual expert lies in the capacity to analyze problems both in the exact and inexact realms. Expert systems try to capture the essence of the expertise, both public and

private, related to a problem domain and use it for problem solving and explanation.

4.1.1 Components of Expert System

The general architecture of an expert system, as shown in Figure 4.1, typically includes four main components:

1) The knowledge base: This is an information data base which contains declarations of facts and specifications and problem-solving logic, and which may also contain various if-then rules, heuristics, and decision guidelines. The knowledge base is limited to a specific, usually narrow, domain. There are different techniques for knowledge representation, but most expert systems apply programs that are pattern invoked, e.g., patterns of known facts which lead to solutions.
2) The inference engine specifies how the knowledge must be applied in order to derive solutions to given problems. Control strategies in the inference engine vary from state-space search techniques to problem reduction schemes.
3) An interface through which users can add to or modify the system knowledge, specifying problem details in order to obtain advice or information.

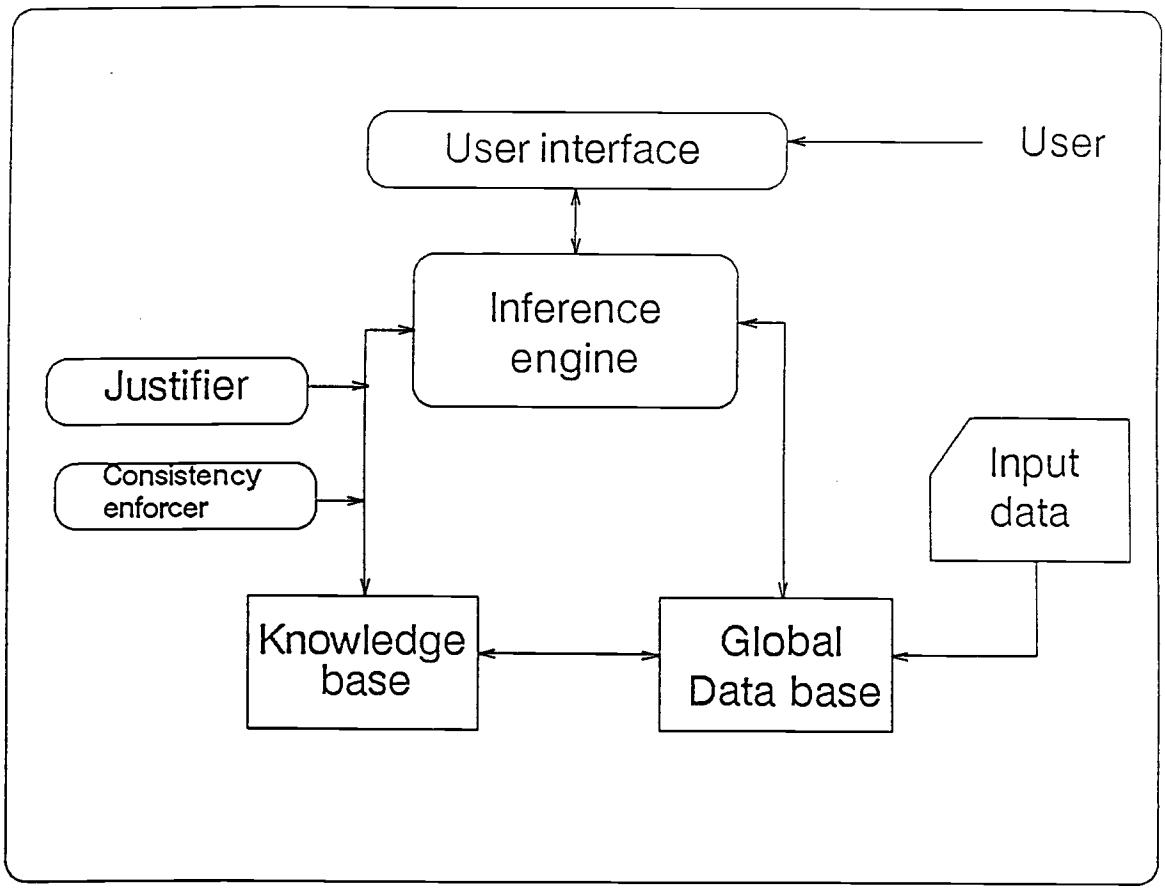


Figure 4.1 General architecture of an expert system.

4) Peripheral systems, such as (a) a consistency enforcer, which adjusts previous conclusions when new data or knowledge alters their basis of support and maintains a consistent representation of the emerging solution, and (b) a justifier, which rationalizes and explains the system's behavior.

A crucial issue is that these components are very difficult to program with conventional programming languages. The reason is that for the problems of interest to this study, the exact procedural solution steps may not be known. For instance, different sets of logic rules could be applied in different instances. As a result, unlike conventional computer programs, the solution logic must be separate from the solution procedure.

4.1.2 Constructing an Expert System

The expert system building phase is time-consuming and not a trivial task. The construction of an expert system takes place in the following five phases:

4.1.2.1 Problem Definition

The problem definition phase involves understanding the problem, identifying problem characteristics, outlining the objectives of the problem solving processes, and clearly defining the methodologies required to solve the problem. During this phase, the knowledge

engineer and the domain expert proceed toward identification of the problem under consideration, involving an informal exchange of views on various aspects of the problem, its definition, characteristics, key elements, and sub-problems. The objective is to characterize the problem and its supporting knowledge structure so that the development of the knowledge base may proceed.

4.1.2.2 Knowledge Acquisition

The prerequisite for an intelligent system is relevant knowledge. To build the dictionary of knowledge, the analyst constructing the expert system must acquire knowledge and incorporate it into the system. Knowledge about the problem domain is acquired either from the study of published literature or from human experts in the domain. The expertise to be elucidated is a collection of specialized facts, procedures, and judgemental rules about the narrow domain area, rather than general knowledge about the domain or common sense knowledge about the world. This knowledge extraction process is termed "knowledge acquisition." The transfer and transformation of knowledge required to represent expertise for a program may be automated or partially automated in some special cases. For the most part a second person, called a knowledge engineer, is required to communicate with the expert and the system.

4.1.2.3 Knowledge Representation

The acquired knowledge must be represented in a computer implementational form. There is a standard set of knowledge representation techniques, any of which can be used alone or in conjunction with others to build expert systems. Each technique provides the program with certain benefits, such as improving efficiency, making it more easily understood and easier to modify. The most widely used techniques in current expert systems are logic, semantic nets, production systems, and frames.

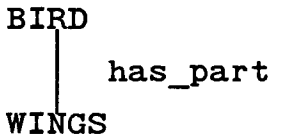
First order predicate calculus is used as a logical representation scheme. The description of the real world is given in term of logic clauses. For example, the fact that all birds have wings can be represented as

$$\forall x \; Bird(x) \rightarrow HasWings(x) \; ,$$

which reads: for any object x in the world, if x is a bird, then x has wings. Logic representations are useful in providing formal proof procedures (Nilsson, 1971), information retrieval (Reiter, 1978), and semantic constraint checking (Nicholas & Yazdanian, 1978). They offer clarity, are well-defined and are easily understood. Each fact needs to be represented only once, regardless of repeated usage. However, logic schemes offer difficulty in procedural representation and if

the number of facts becomes too large, there is a combinatorial explosive situation in the possibilities of which rules to apply to which facts at each step of proof.

Semantic nets were developed by Quillian (1968) as an explicit psychological model for human associative memory. Semantic nets describe the world in terms of nodes, representing objects, concepts, and events, with links between nodes representing their interrelationships. Consider, for example, the simple net:



where BIRD and WINGS are nodes representing objects and has_part is the name of the link specifying their relationship. The statement, "all birds have wings" is among the many possible interpretations of this net.

According to a semantic net representation, knowledge is a collection of objects and associations represented as a labeled graph. Semantic nets are easily understood, but they are difficult to implement. Since there are no conventions about their meaning, the implementation of net structures depends solely on the program that manipulates them. Therefore, inferences drawn by manipulation of the net are not assuredly valid, in the sense that they are assured to be valid in a logic-based representation scheme.

Production systems, developed by Newell and Simon (1972) for their models of human cognition, are a modular knowledge representation that has become increasingly popular in large AI programs. The basic concept underlying these systems is that the database consists of rules, called productions, in the form of condition-action pairs: "IF the condition occurs THEN do this action." The utility of the formalism comes from the fact that the conditions in which each rule is applicable are made explicit and the interaction between rules is minimized. Production systems have been found useful as mechanisms for controlling the interaction between statements of declarative and procedural knowledge. Because they facilitate human understanding and the modification of systems with large amounts of knowledge, production systems have been used in several large expert system applications, such as DENDRAL, MYCIN, and PROSPECTOR.

The most recently developed AI knowledge representation scheme is the frame. Since Minsky (1975) originally proposed its use, the notion of the frame has played a key role in knowledge representation research. Basically, a frame is a data structure that includes declarative and procedural information in predefined internal relationships. Several kinds of information, such as how to use the frame, what can be expected to happen next, and what to do if these expectations are

not confirmed, are attached to each frame as slots and fillers. Consequently, the frame system is the most difficult to implement because it contains a greater degree of structural complexity than do other forms of representation. It is necessary to determine what kinds of frames the system needs, what kinds of slots each frame will require, and how all the frames fit together in a hierarchical structure. An excellent summary of this technique can be found in the *Handbook of Artificial Intelligence* (Barr, 1981).

4.1.2.4 Development of an Inference Engine

The main purpose of developing an expert system is to generate alternate paths that lead to an inference. In order to accomplish this task, the procedures built into the system should act upon the knowledge base in an efficient manner and derive conclusions. This process, generation of alternate paths via a reasoning mechanism through the knowledge base to derive conclusions, is termed "inference."

One of the most variable characteristics of expert systems is the way that they search for solutions. The choice of search method is affected by many different domain characteristics, including the size of the solution space, errors in the data, and the availability of abstractions. However, this method must be decided upon when the system is designed. Inference is at the heart of a reasoning system and failure to organize it

properly can result in problem-solvers that are hopelessly inefficient, naive, or unreliable. As a consequence, search and/or inference mechanisms are among the most critically studied topics in AI and expert systems.

4.1.2.5 Implementation and Evaluation

The first step in implementation is to determine which programming language can be used. Most expert systems are based on *LISP* (Winston and Horn, 1984); however, *PROLOG* (Clocksin and Mellish, 1981) is gaining in popularity. *LISP* offers a high degree of flexibility for writing rules and the expert system builder can specify his own framework of rules. *PROLOG* is based on Horn clauses of logic and rules need to be written explicitly in formal logic. *PROLOG* has a built-in backward chaining mechanism, which makes it more convenient. However, caution needs to be exercised while using the backtracking mechanism in *PROLOG*. Furthermore, neither *PROLOG* nor *LISP* may be convenient for handling mathematical computations.

Several expert system building tools are currently available, including *EMYCIN* (van Meile et al., 1981), *KAS* (Duda et al., 1979), *EXPERT* (Wiess & Kulikowski, 1979) and *OPS-5* (Forgy, 1981).

One important consideration in the implementation phase is that of user interface. Since the expert system is expected to help the novice user, it should be

user-friendly. In addition, it should be conversational in its approach and so far as possible, visual graphics should be used and explanations should be concise, clear and follow the reasoning chain.

Finally, evaluation involves testing the performance and utility of the prototype program and revising it as necessary. The domain expert typically evaluates the prototype and helps the knowledge engineer make revisions. As soon as the prototype runs accurately for a few examples, it should be tested on a variety of problems to evaluate its performance and utility. This evaluation may uncover missing concepts and relations, knowledge represented at the wrong level of detail, or unwieldy control mechanisms. These problems may force the developers to recycle through the various developmental phases, reformulating concepts, refining the inference rules, and revising the control flow.

The expert system must be refined and tested before it can be released for field testing. However, when it is tested by the user on real problems, new complications will arise which may take some time to correct. Users in the field demand more than just high-quality performance; they want a system to be fast, reliable, easy to use, easy to understand, and very forgiving when they make mistakes. Thus, the expert system needs extensive testing before it will be ready for commercial use. More detailed explanations

of building an expert system may be found in Buchanan and Shortliffe (1985) and Hayes-Roth et al. (1983).

4.1.3. Types of Expert Systems

Stefik et al. (1982) pointed out some domain characteristics that affect expert system design: large search spaces, a need for tentative reasoning, time variations, and noisy data. These four characteristics were elaborated in 11 case studies, along with additional guidelines for expert system construction. This information constitutes helpful assistance for expert system designers in the clarification of domain characteristics and the development of conceptual system designs. Ten generic tasks which can be handled by expert systems are summarized in Table 4.1.

Moreover, a number of studies describing expert system applications in the manufacturing environment can be found in Bernold (1985), Falster and Mazumder (1984), Faught (1986), and Lu and Komanduri (1986). A thorough survey of expert systems is provided by Nau (1983).

4.2 Benefits and Problems in Building an Expert System for FMS Scheduling

With the increasing popularity of AI and expert systems, there have been many attempts to build expert

Table 4.1 Generic Tasks of Expert Systems (from Hayes-Roth 1983)

Task	Problem Addressed
Interpretation	Analysis of data to determine their meaning
Diagnosis	Fault-finding in a system based on data interpretation
Monitoring	Continuous interpretation of signal data in order to trigger an alarm when action is necessary
Prediction	Forecast of future events based on past and present data
Planning	Creation of programs that can be executed to attain given goals
Design	Determination of specifications to create objects that satisfy given requirements
Debugging	Prescribing remedies for malfunctions
Repair	Executing a plan to administer a prescribed remedy
Instruction	Diagnosing, debugging, and repairing student behavior
Control	Interpreting, predicting, repairing, and monitoring system behaviors

systems without considering whether an expert system really helps the problem-solving process. In many cases it may be possible to solve the problem by direct analytical means and in such cases an expert system may not be appropriate. An expert system is ideally suited for situations (1) when problems in the domain cannot

be well-defined analytically; (2) when problems can be formulated analytically, but the number of alternate solutions is large, as in the case of combinatorial explosive problems; or (3) the domain knowledge is vast and relevant knowledge needs to be used selectively. If a problem fits into any of these categories, it will be worthwhile to construct an expert system (Kumara, 1986).

The scheduling problem for manufacturing systems is one of these problems. The problem of scheduling is a combinatorial problem, belonging to the class of NP complete, hard problems. In this study the reason for using an expert system for FMS scheduling has been to provide a computerized scheduler that has the same expert knowledge of qualitative measures that a human scheduler possesses, thus reducing the number of available alternatives and simplifying selection of the best schedule.

The potential advantages of using an expert system for scheduling include:

1) the ability to produce feasible schedules rapidly;
2) the ability to quickly accommodate changes within a dynamic environment;
3) the ability to be intelligible to the operator of the system;

4) the ability to be consistent in decision-making to exercise control of an FMS; and

5) the ability to store expertise and use it as a training tool.

Although the expert systems approach to FMS scheduling appears promising, building an expert system for manufacturing system scheduling is a time-consuming and tedious process. Among the five phases of constructing an expert system described in section 4.1.2, knowledge acquisition is the major bottleneck (Hart, 1985; Hayes-Roth et al., 1983) because of its difficult and time-consuming nature. Hayes-Roth et al. described four different methods of knowledge acquisition (see Figure 4.2):

1) interviews with human experts;

2) an intelligent editing program;

3) an induction program; and

4) a text understanding program.

In the first method, a knowledge engineer interviews the expert and extracts the appropriate expertise. In the second method, the knowledge engineer's role is replaced by an intelligent editing program, which has dialogue capabilities and knowledge of the structure of the knowledge base. In this method, the expert converses directly with the editing program in order to structure his knowledge. In the third method, the role of the expert is replaced by a computer pro

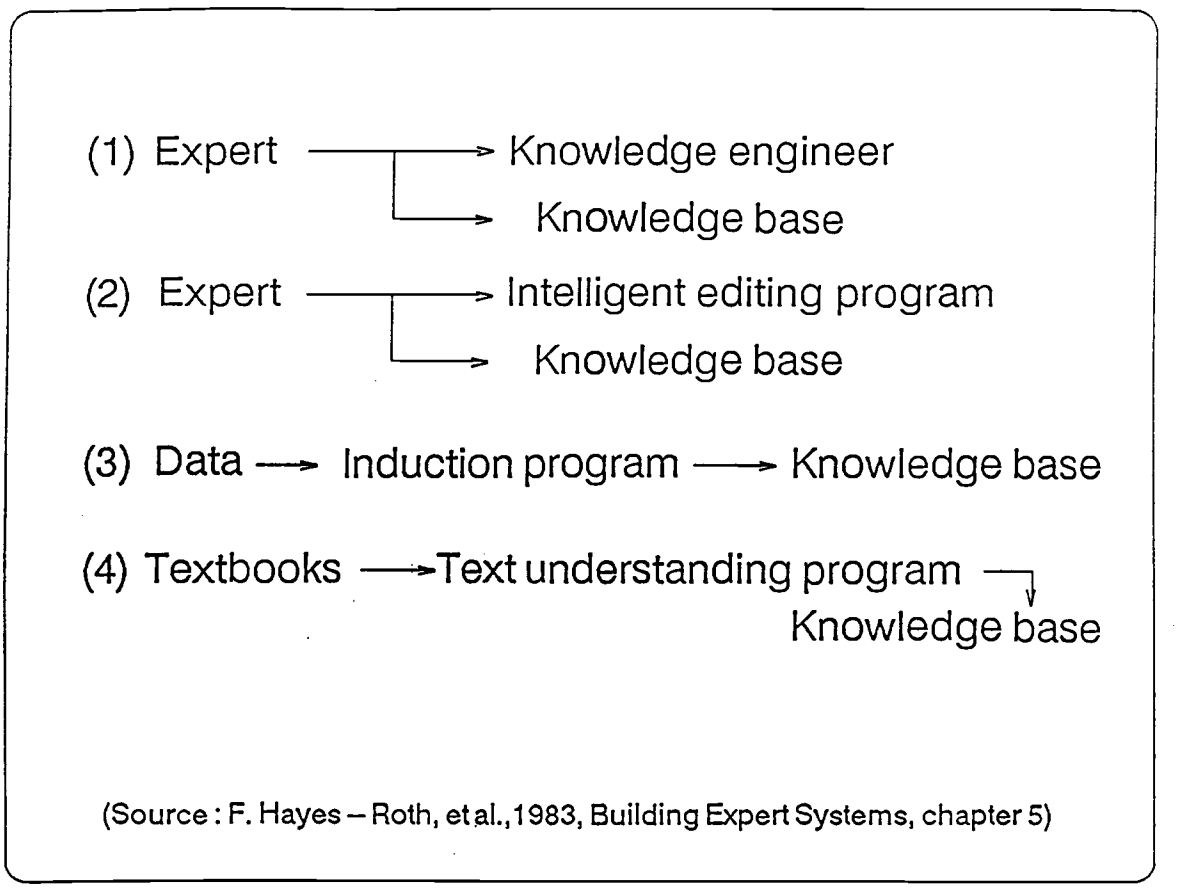


Figure 4.2 Knowledge acquisition methods.

program capable of generalizing specific cases in order to draw conclusions. Finally, a textbook understanding program which has the capability of reading and understanding text could in the future be used as a method of knowledge extraction.

Most of the current expert systems have been developed by interviewing human experts. However, experience has shown that it is often difficult to obtain an expert's true opinion and that when the problem is new, acceptable expert knowledge does not exist. Bell (1985) has explained that an expert systems approach can fail because proper knowledge cannot be acquired for one or more of the following reasons:

1) An expert is not available, though expert knowledge does exist;
2) The expert is unable to communicate his ideas;
3) The expert is unwilling to communicate his ideas; or
4) There is no expert.

In building an expert system for the control of an FMS, the problem of acquiring expert knowledge was encountered in the preparation of this study. Since the FMS reflects relatively new technology, and since most systems are custom designed for specific operations, more often than not an expert human scheduler is unavailable. Furthermore, the complexity of the FMS limits the effectiveness of human decision-making, i.e.,

human operators may often and unknowingly repeat mistakes. The complex and dynamic nature of the FMS environment has made it difficult to develop human expertise, which may prevent the FMS from achieving optimal or near optimal performance on a consistent basis. Therefore, no human expertise is available on how to control an FMS and there is no one to consult and glean rules from in the conventional sense of building an expert system.

Thus, other methods should be designed to obtain the knowledge required for an FMS expert system. To this end a detailed account of a knowledge acquisition method fundamental to the development of an FMS scheduling expert system is given in Kim et al. (1988a). It is generally accepted that humans have good pattern-recognition and inductive-logic capabilities, usually superior to those possessed by a machine. Therefore, a model which can effectively utilize the human's unique information processing abilities can be used for knowledge acquisition. For this purpose, an interactive graphic simulation model has been developed. It is graphic-based because humans perform better in pattern-recognition when graphic information is used in place of text information. Moreover, the graphics environment may be interactively matched to the human's information processing speed. If the computer processing is faster than human information

processing, then a human operator would obtain little information from the computer model. Thus, the human operator should be given control of the system. In the following section, this method of building an expert system is described in greater detail.

4.3 Development Procedure for an FMS Scheduling Expert System

While human operators are unable to handle the information load required for FMS scheduling problems, computer systems offer unique capabilities for confronting this challenge. The problem of remembering previous circumstances upon which further decisions must be based is an ideal application for knowledge based expert systems. The challenge is to develop a computer-based FMS control system, responsible both for controlling the flow of jobs through the FMS and for formulating decisions based on the results of simulation experiments.

A new method for developing an expert system for FMS scheduling is described in this section (Kim et al. 1988b). There are five components in the method:

1) A graphic simulation model of an FMS;
2) A Gantt chart-based schedule;
3) Simulator;
4) An expert system; and

5) An operator.

The method requires considerable interaction with a human operator; a graphic representation, illustrating the interaction process between the system and the operator, is shown in Figure 4.3.

The entire procedure is as follows: Upon arrival of a new job order or a rescheduling request from the FMS system, the simulator generates an initial schedule which considers the operator's objective and the current system status. As the initial schedule is generated, expert systems are consulted to determine the loading sequence for the new job order, to select a dispatching rule which will be used in the simulation, and to edit the initial schedule. The process of generating the schedule is shown in Figure 4.4. The process involves the following steps:

1) The expert system/human operator determines a loading sequence for the job order;
2) The expert system/human operator selects a dispatching rule;
3) The simulator runs a simulation of the model;
4) The simulator generates an initial schedule in the form of a Gantt chart;
5) The expert system/human operator edits the Gantt chart according to scheduling objectives; and

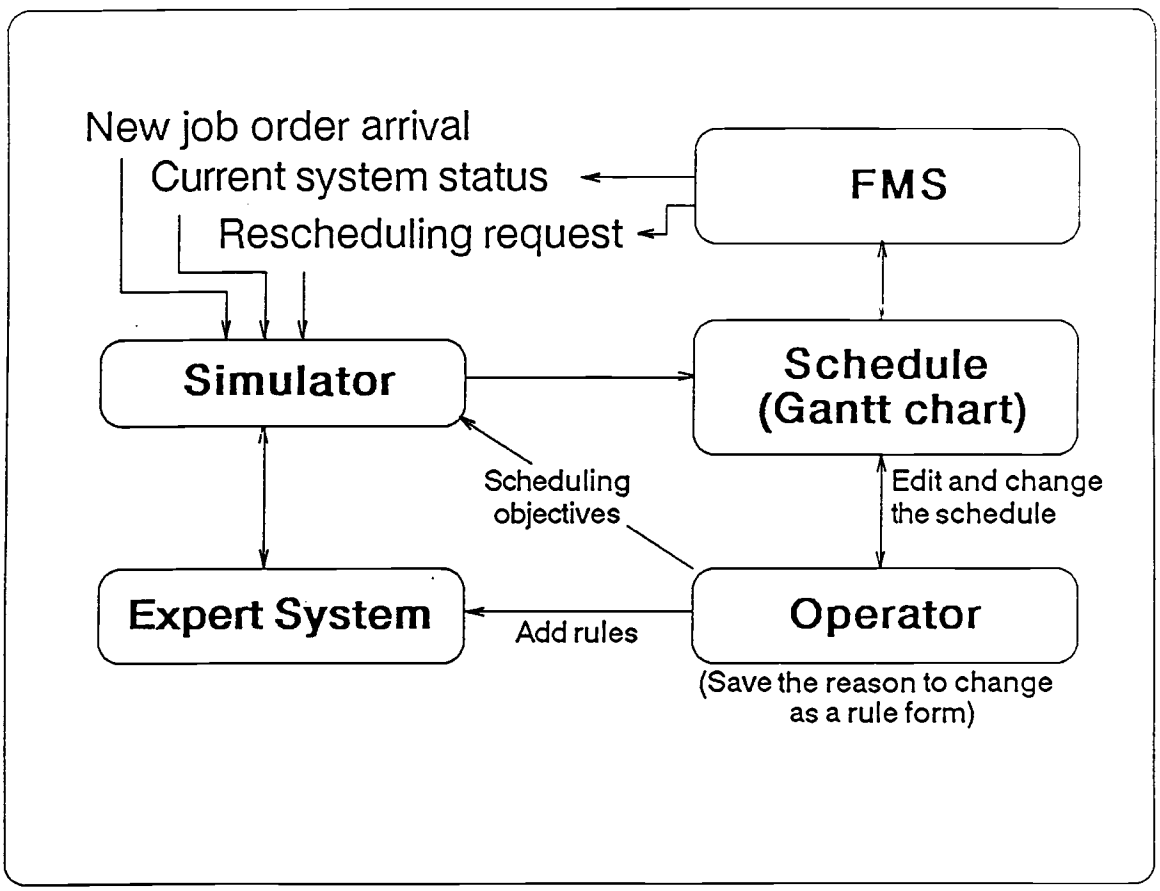


Figure 4.3 Expert system development procedure.

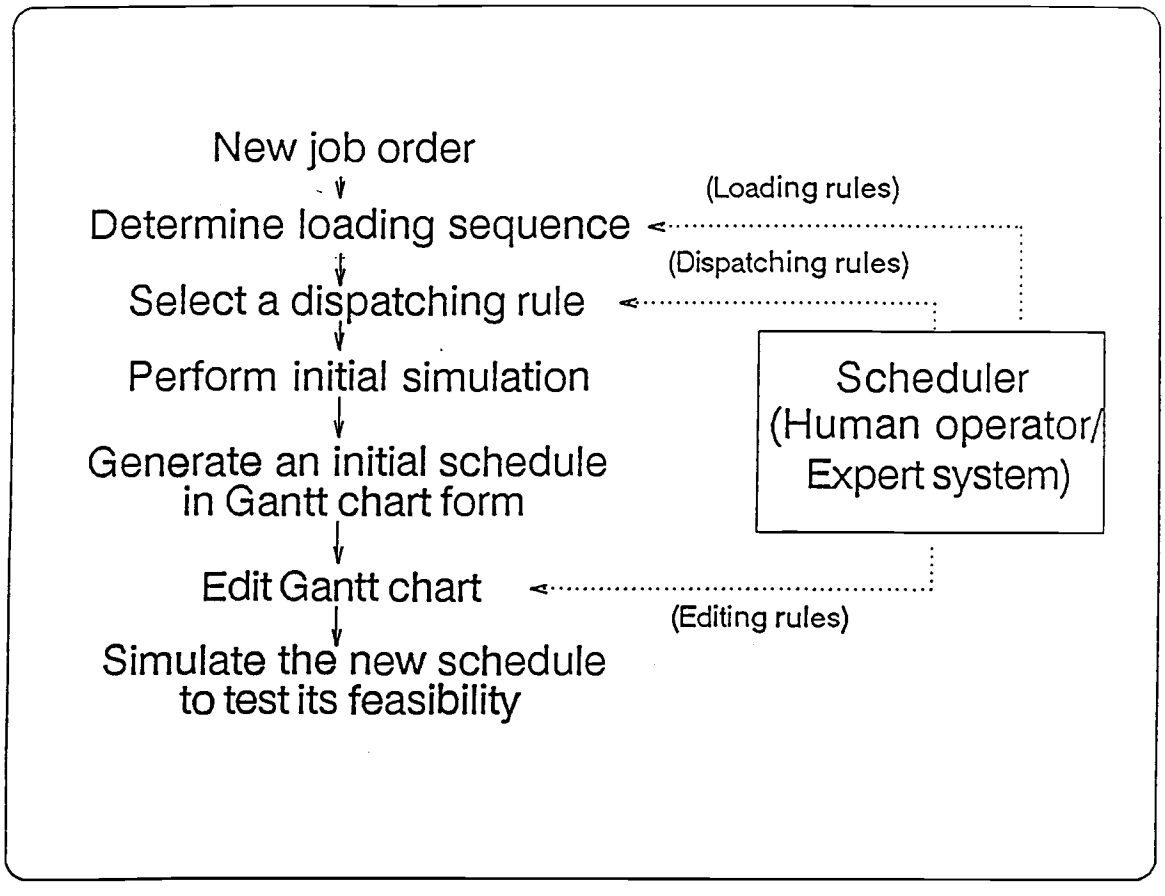


Figure 4.4 Knowledge acquisition procedure in scheduling an FMS.

6) The human operator runs the simulation according to the new schedule to test its feasibility.

The first step is the development of a good initial loading plan, or determining the order in which jobs are introduced into the system, and a dispatching rule, deciding which job to start next at each machine. From the loading plan, an initial schedule is derived through simulation. The development of a good loading plan and determination of a good dispatching rule are important because the value of the final schedule is dependent on their formulation in an initial schedule. Then, the initial schedule is inspected by the operator, who is able to change the schedule to maximize its objectives and to save reasons for the change as a new production rule, i.e., in the form of an "IF condition THEN action." The collected rules are subsequently generalized and added to the knowledge base of the expert system. The process is repeated until the operator no longer needs to change the schedule. In Figure 4.4, it may be seen that three different kinds of knowledge are supplied to obtain a final schedule: (1) knowledge for determining a loading sequence; (2) knowledge for selecting a dispatching rule; and (3) knowledge for changing a preliminary schedule.

The key element of this method is use of a Gantt chart, which can be easily manipulated, for scheduling.

By selecting a black of operations in the Gantt chart with a mouse pointing device, the operator can easily change the schedule and analyze the effects of the change. This interactive facility, based upon graphical representation of the simulation, allows the operator to view many aspects of the system. It not only reflects the state of the system, but helps the human operator determine the causes of problems.

This system is also used to acquire the rules to build the expert system. A human operator controls the model until enough rules are collected to form the knowledge base of the expert system. Whenever the operator selects the next job in determining the loading sequence, the system asks the operator the reason for his/her selection. This reason, with the current state of determination of a loading sequence, is recorded in a file. These records are inspected after simulation, analyzed by the operator and added to the knowledge base of the loading expert system. The editing rules are collected similarly, i.e., the operator can record the change in a file after editing and testing the schedule. These facilities are always available so that new rules can be developed as the need arises.

Separate decision aids are supplied for each type of knowledge acquisition. A job matrix window is used to help the scheduler determine the loading sequence. Several dispatching rules are supplied for selection

and an editable Gantt chart is produced for schedule arrangement. In the next chapter, a detailed description of the model is given, accompanied by some example results.

5.0 IMPLEMENTATION AND OPERATING PROCEDURE

5.1 Implementation

The system was implemented in *Smalltalk/V* on an IBM PC/AT. *Smalltalk/V* (1986) is a dialect of the Smalltalk language, an object-oriented language developed at Xerox (Goldberg and Robson, 1983). Over the past few years, the Smalltalk programming language has attracted considerable interest in the AI programming community since it has not only the flexibility of a general-purpose programming language, but the power of a simulation language for modeling physical objects and their behaviors.

In fact, Smalltalk is much more than a programming language. It is an entire environment, including editor, compiler, debugger, and many other tools to aid in software development. Smalltalk's major strengths include: an excellent user interface, a powerful programming development environment, graphics capabilities, and a single, powerful conceptual metaphor--the sending of messages to objects. The metaphor is powerful because it is both simple and general. In addition, Smalltalk uses a clean embodiment of abstraction in its object concept, assisting programmers in writing

safer and more robust software which requires less debugging. Smalltalk also has the powerful and useful characteristic of using the principle of inheritance, which allows many types of objects to share the same methods and variables. Smalltalk's only serious weakness is its demand for a fast processor and a great deal of memory. However, due to the rapid development of computer technology over the past five years, an efficient implementation of Smalltalk or another object-oriented programming language on a microcomputer is now available. For these reasons, Smalltalk has been successfully used for simulation, expert systems, and integrated programming environments.

In the following two sections, the fundamentals of the Smalltalk system, with descriptions of the functions of objects and classes, and the system operating procedure, accompanied by an example problem, are introduced. A detailed description of the system and its operating procedure is given in a *Schedule Generator Manual* (Kim, 1988c) and a demonstration videotape is available (Kim, 1988d).

5.2 Smalltalk and Protocol Description

In this section, the fundamental concepts of the Smalltalk system, with a description of the functions of objects and classes, is introduced.

5.2.1. Introduction to the Smalltalk System

In Smalltalk, system components are represented by objects which interact through message transmissions. The messages, in turn, cause the execution of the methods prescribed within each class. These objects and messages are used to implement the entire programming environment, i.e., programming in Smalltalk consists of creating objects and classes, and specifying message exchanges sequences among the objects. Table 5.1 provides definitions for the basic Smalltalk terms.

Table 5.1 Definition of basic Smalltalk terms.

Object	Everything in Smalltalk is represented as an object, each consisting of an amount of dedicated memory and a set of operations.
Message	A request for an object to carry out one of its operations. The message does not specify how the function will be performed and the receiver of the message determines how to accomplish the requested function. A set of message represents interactions between system components.
Method	Describes how an object performs one of its operations as requested by a specific type of message.
Class	Describes the implementation of a set of objects, all of which represent the same kind of system component. It describes the form of the dedicated memory for each instance, providing instructions on how to carry out operations.
Instance	Object which is a member of a class.

The most important Smalltalk program design consideration is the determination of the kinds of objects to describe and message names which provide a useful

interaction vocabulary among these objects. The appropriate choice of objects is consequently dependent upon the chosen purpose of each object and the granularity of the information to be manipulated. However, there is no definitive manner of choosing objects and, as with any design process, this is an acquired skill. Different choices provide different bases for extending the use of an application's objects for other purposes.

5.2.2 System Protocol Description

Golberg and Robson (1983) present three ways of describing the implementation of object and class functions, including:

1) A "protocol description" lists the messages in the instances' message interface. Each message is accompanied by a comment describing the operations an instance will perform when the specific message is received.
2) An "implementation description" shows how the functionality described in the protocol description is implemented, giving the form of each instances' private memory and the set of methods that describe how each instance performs its operations.
3) A system browser can be used to present classes in a interactive view. The browser is a part

of the programming interface used in a running Smalltalk system.

In this section, the protocol description format is used to show the functions of objects and classes. The implementation description can be accessed by a system browser in the "Schedule Generator" (Kim, 1988c).

The system includes a set of classes for each system component. Figure 5.1 presents a diagram of the classes used without the Smalltalk system classes. Lines are drawn around groups of related classes; groups are labeled to indicate the sections in which the protocol description of each class can be found.

The Smalltalk system was also developed using the same sequence. First, a simulation model was developed through simulation of classes, including "Simulation Object" and "Simulation". Then, graphical representations of the simulation model (class "Layout" and "LoadingScreen") and a Gantt chart (class "GanttChart") were added to the simulation model, while at the same time the overall control class ("FMSExecutive") was developed. These four groups consisted of an interactive graphic simulation system used as a knowledge acquisition tool. Finally, expert system classes were added to the system.

5.2.2.1 Simulation Classes.

The objects used in computer simulations operate more or less independently of one another. Therefore,

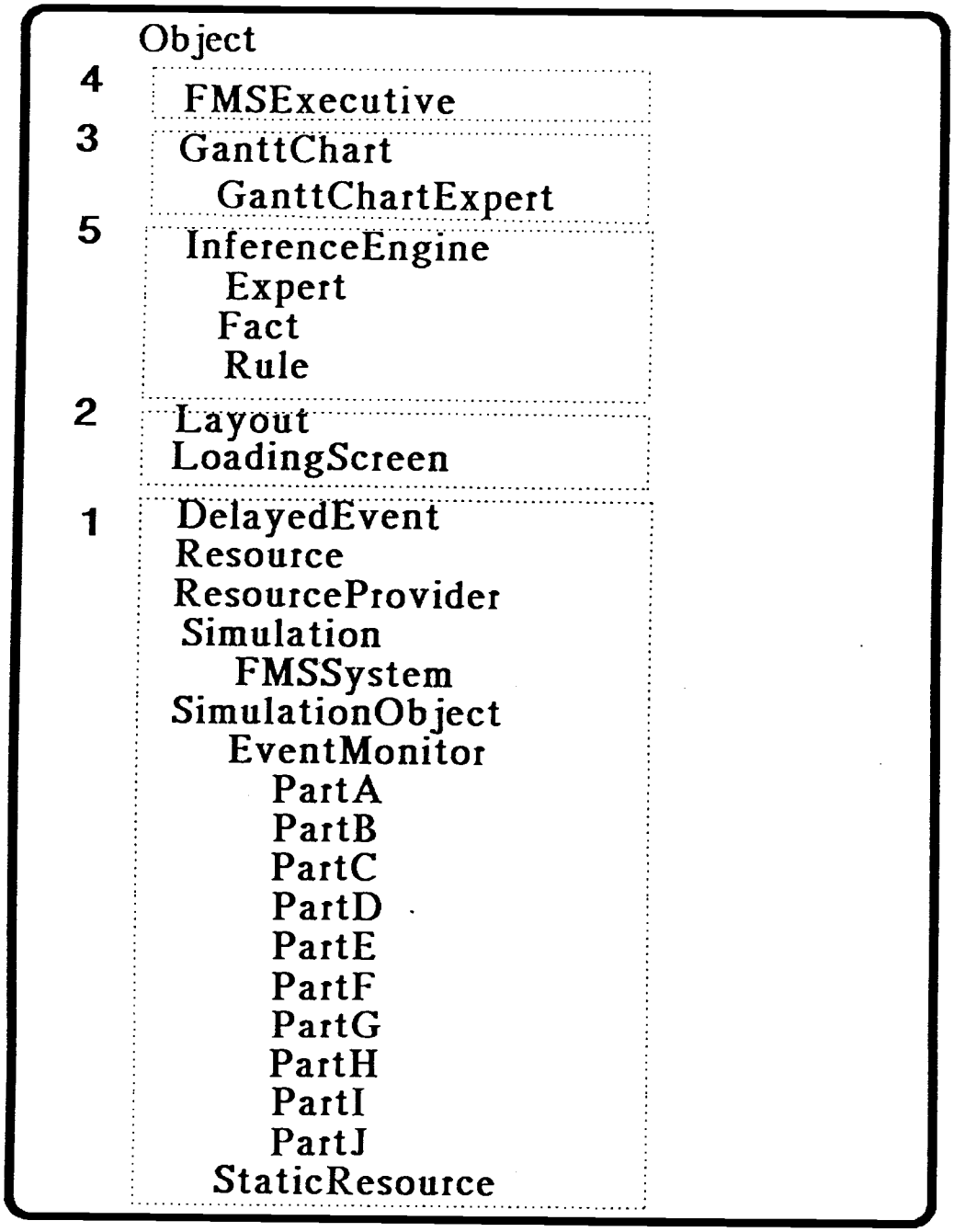


Figure 5.1 Schedule generator system classes.

it is necessary to consider the problem of coordinating the activities of the various simulated objects, which are typically coordinated through the mechanism of message passing. This section describes the classes that provide the basic protocol for "SimulationObject" and "Simulation", which were initially implemented by Goldberg and Robson (1983) and modified for the "Schedule Generator" (Kim, 1988c).

The class "SimulationObject" describes a general type of object that might appear in a simulation, i.e., one with a set of tasks, such as entering, processing, waiting at queue, or leaving the system. The class message provides a framework for carrying out the task. An instance of the class "Simulation" maintains the simulated clock and the queue of events. The major messages that simulation objects can use to describe their tasks are provided in Table 5.2.

The purpose of the class "Simulation" is to manage the topology of simulation objects and to arrange scheduling to occur in accordance with simulated time. Instances of the class "Simulation" maintain a reference to a "SimulationObject" collection for the current simulated time and to a queue of events waiting to be invoked. "Simulation" advances time by checking the queue to determine when the next event is scheduled to take place and by obtaining the instance variable for

Table 5.2 "SimulationObject" instance protocol.

initialize	Initialize instance variables, if any.
startUp	Inform the simulation that the receiver is entering, then initiate the receiver's tasks.
tasks	Define the sequence of activities that the receiver must carry out.
finishUp	The receiver's tasks are completed. Inform the simulation.
holdFor: aTimeDelay	Delay carrying out the receiver's next task until "aTimeDelay" length of simulated time has passed.
acquire: amount ofResource: resourceName	Ask the simulation to provide a resource referred to by the string "resourceName". If one exists, ask it to give the receiver "amount" of resource.
release: aStaticResource	The receiver has been using the resource referred to by the argument, "aStaticResource". It is no longer needed and can be recycled.
inquireFor: amount ofResource: resourceName	Answer whether or not the simulation has at least a quantity, "amount", of resource referred to by "resourceName".
resourceAvailable: resourceName	Answer whether or not the simulation has a resource referred to by "resourceName".
acquireResource: resourceName	Ask the simulation to provide a resource simulation object that is referred to by the "resourceName".
resource: anEvent	The receiver has been servicing the resource referred to by the argument, "anEvent". The service is completed and the simulation object can continue its tasks.
stopSimulation	Tell the simulation in which the receiver is running to stop. All scheduled events are removed and nothing more can occur in the simulation.

the time associated with that event. The major messages in the "Simulation" class are described in Table 5.3.

Table 5.3 "Simulation" instance protocol.

initialize	Initialize the receiver's instance variables.
defineArrivalSchedule	Schedule simulation objects to enter the simulation at specified time intervals, based on typical probability distribution functions.
defineResources	Specify the resources that are initially entered into the simulation.
schedule: actionBlock at: aTimeInteger	Schedule the sequence of actions in "actionBlock" to occur at a particular simulated time, "aTimeInteger".
scheduleArrivalOf: aSimulationObjectClass accordingTo: aProbabilityDistribution startingAt: aTimeInteger	Schedule simulation objects that are instances of "aSimulationObjectClass" to enter the simulation at specified time intervals, based on the probability, distribution, "aProbabilityDistribution". The first such instance should be scheduled to enter at time, "aTimeInteger".
includesResourceFor: resourceName	Answer if the receiver has a resource, "resourceName".
time	Answer the receiver's current time.
startUp	Specify the initial simulation objects and the arrival schedule of new objects.
proceed	This is a single event execution. The first event in the queue, if any, is removed, time is updated to the time of the event, and the event is initiated.
finishUp	Release references to any remaining simulation objects.
enter: anObject	The argument, "anObject", informs the receiver that it is entering.
exit: anObject	The argument, "anObject", informs the receiver that it is exiting.

In implementing the scheduling mechanism, there are several problems that must be solved in the design of class "Simulation", i.e., how is an event stored that must be delayed, how can the operator be certain that all processes initiated at a particular time are completed prior to updating the clock, and how can the request to repeat a sequence of actions for simulation objects by schedule be implemented?

To solve these problems, the class simulation maintains a queue of all scheduled events and the elements of each event are sorted with respect to the simulated time in which they must be invoked. Each event in the queue is handled by the class "DelayedEvent" and processed by the classes for a Smalltalk multiple independent processor.

The framework presented above for specification of event-driven simulations does not include methods for gathering simulation statistics while running. This is accomplished by creating a subclass of "SimulationObject", the "EventMonitor". The class "Resource" and its subclass "ResourceProvide" manage and maintain the required simulation resources. When simulation objects require resources, the requests are queued and ordered with respect to priority. Each time a resource request is made, "Resource" and "ResourceProvide" check to see if one or more of the pending requests can be satisfied.

5.2.2.2 Graphic Layout Classes

There are two classes for representation of the graphical interface. The class "Layout" displays the FMS model layout in the main simulation window and the class "LoadingScreen" displays the job-information matrix window used in determining an initial loading sequence. The methods used in these classes are shown, respectively, in Tables 5.4 and 5.5.

5.2.2.3 Gantt Chart Class

The Gantt chart is also a graphical representation in the main simulation window. The instances of this class display not only an object part as it runs through the simulation, but also manipulate the resulting Gantt chart schedule generated by the simulation. Table 5.6 shows the protocol description of the principal methods used in this class.

5.2.2.4 Overall System Control Class

The class "FMSExecutive" controls and directs all system processes and operations, starting the simulation, opening windows, specifying the menus to be used, and performing the administrative functions necessary to interface corresponding objects and classes. The methods used in the class are listed in Table 5.7.

Table 5.4 "Layout" instance protocol.

initialize	Initialize the instance variable and display the physical layout in the main simulation window.
display: aPoint	Display the receiver part type on the layout.
move: aPoint	Determine the next location of the receiver part type and move to that location.
erasePart	Answers the erase part form.
part	Answer the part form.

Table 5.5 "LoadingScreen" instance protocol.

initialize	Initialize the instance variable and display the job information in matrix form.
setGrid: aMachine with: aSequence	Find the location for "aMachine" with the corresponding sequence, "aSequence", of the operator of the receiver part type.
display: aTime sequence: aSequence machine: aMachine	Display the updated processing time information, "aTime", at the corresponding location, "aMachine", and sequence, "aSequence", as each part is selected in the initial loading sequence.
displayDelete: aTime sequence: aSequence machine: aMachine	Delete the last selected job information in the loading screen window.
deleteJob	Delete the selected job in the loading sequence.
selectA • • • SelectJ	Select the corresponding part type for the job. Order and add the corresponding part type to the initial loading sequence.

Table 5.6 "GanttChart" instance protocol.

initialize	Initialize the instance variables and draw the basic frame of the Gantt chart.
display: aMachine at: aTime with: aTimeDelay for: aPartID	Display an operational block of part, "aPartID", in the location, "aMachine", at time, "aTime", with the processing time length set in "aTimeDelay".
move: aPoint	Select the operational block to be moved by checking the current location of the mouse pointing device. Erase the selected block and relocate it in the Gantt chart.
redisplay: aBox withSeq: i	Following editing, recalculates the correct location of the operational block and replaces it at the operator's command to redraw the Gantt chart.
automatic	Perform an automatic simulation according to the updated Gantt chart schedule to test the feasibility of the new schedule.
manual	Perform a manual simulation according to the updated Gantt chart schedule.
precedenceCheck	Checks the precedence constraint of the part. Reports an error if a part violates its precedence constraint.

Table 5.7 "FMSExecutive" instance protocol.

initialize	Initialize the instance variables and the job information dictionary.
assignSelectedRule: aDispatchingRule	Assign the selected dispatching rule.
cancelSelectedDispatchingRule	Cancel the selected dispatching rule.
consultDispatchingExpert	Call the dispatching expert system.
consultLoadingExpert	Call the loading expert system.
ganttChartMenu	Answer the Gantt chart menu for system control.
jobSelectionMenu	Answer the job selection menu in the job matrix information window for system control.
layoutMenu	Answer the system layout for system control.
ruleMenu	Answer the rule listing menu to select a dispatching rule.
newLoadingSequence	Determine a new loading sequence.
openWindow	Open the main simulation window for the FMS simulation.
openJobMatrixWindow	Open the job information matrix window for determining an initial loading sequence.
quit	Stop the simulation.
runTheSimulation	Start the simulation.
start	Start the program by drawing the layout and the Gantt chart in the main simulation window.

5.2.2.5 Expert System Classes

The expert system application is implemented by the classes "Expert", "Fact", and "Rule". For organizational reasons, all three are subclasses of a new class, "InferenceEngine". The methods used in these classes are presented, respectively, in Tables 5.8,

5.9, 5.10, and 5.11. A brief description of these classes is as follows.

Each instance of the class "Expert" is self-contained with respect to the rules, which when combined with the facts, represent the knowledge base of the expert system. All instances of the class "Expert" share one dictionary of "Facts". The result of this arrangement is that different expert systems can exchange information through the same fact space.

A "Fact" is something known to be true by the expert system, either as a given fact or as an inferred fact, consisting of a pattern and a verbal description of the pattern. Each instance of the class "Expert" contains a collection of rules for its knowledge base, each of which is represented as an object of the class "Rule". When a "Rule" is executed, its conditional part is evaluated as a block of Smalltalk code normally determined from the "Facts" dictionary. If the evaluation answer is true, the action part of the rule is evaluated as another block of code.

Additional methods are described in the class "InferenceEngine", which was initially an abstract class without a method. Subsequently, methods may be added to this class to update system parameter values which are changed as new jobs are selected in an initial loading sequence.

Table 5.8 "Expert" instance protocol.

initialize	Initialize rules as an empty OrderedCollection.
addDispatchingRule	Build rules known to the dispatching expert system.
addLoadingRule	Build rules known to the loading expert system.
selectDispatchingRule	Check the knowledge base of the expert system and select the proper dispatching rule for the arriving job order.
selectNextJob	Considering the state of system parameters, select the next loading job from the jobs waiting in the job order.
explainDispatching	Explain the reason for selecting the dispatching rule.
explainLoading	Explain the reason for selecting the next job in the loading sequence.

Table 5.9 "Fact" instance protocol.

initialize	It is an one-time initialization for the "Fact" class.
is: aPattern	Answer true if "aPattern" matches an existing fact; otherwise answer false.
add: anArray description: aString rationale: factCollection	Set the content of the receiver fact to "anArray", the description to "aString", and the rationale to "factCollection". Then enter the receiver fact into the "Facts" dictionary, answering false if it is already in "Facts", otherwise answering true.

Table 5.10 "Rule" instance protocol.

action	Evaluate the action part of the receiver rule and add the result as a fact.
fire	Answer true if the condition part of the rule is true and the action part is valid; otherwise answer false.
number: ruleNumber condition: condBlock action: actBlock description: aString	Answer a new rule with its contents initialized. The "ruleNumber" helps identify the rule. When "condBlock" is executed and the answer is true, "actBlock" will be fired to create a new fac, which is described in "aString".

Table 5.11 "InferenceEngine" instance protocol.

initialize	Initialize a loading sequence and all the parameters in the system.
readInputData	Read the new arriving job order and calculate the values of the system parameters, including total work load, the number of jobs in the job order, and the number of part types.
selectAllRemainingJobs	Select all the jobs left in the job order.
selectJobByLastOperation	Check the system parameters of the job number finished at each machine, selecting the next job according to its last operation.
selectJobForIdleMachine	Check the system parameters of current machine status, selecting the next job which can be started at an idle machine.
selectJobWithTotalProcessingTime	Select the next job with longest processing time.
updateFacts: aCurrentJob	Update the system parameters after the job, "aCurrentJob", is selected.

5.3 Operating Procedure

The operating procedure for the system is described in the following example problem. Figure 5.2 shows the initial display. The window consists of three panes: the upper pane is the Gantt chart schedule pane used to display an initial schedule and allow the operator to change the schedule; the lower left pane is the system layout pane, used to display system status and part locations during simulation; and the lower right pane is the event-information pane, used to display simulation events and other information about the system. The Gantt chart schedule pane and the system layout pane are graphic panes and the event-information pane is a text pane.

When the operator moves the cursor to the system layout pane and clicks the mouse button, the system layout menu is popped-up. The menu includes three commands, "quit," "start," and "run Simulation," as shown in Figure 5.2. The "quit" command stops the system and returns to the Smalltalk environment. The "start" command initializes the system and asks the operator for a new job order. The "run Simulation" command initiates the simulation, after the operator determines the loading sequence and the dispatching rule.

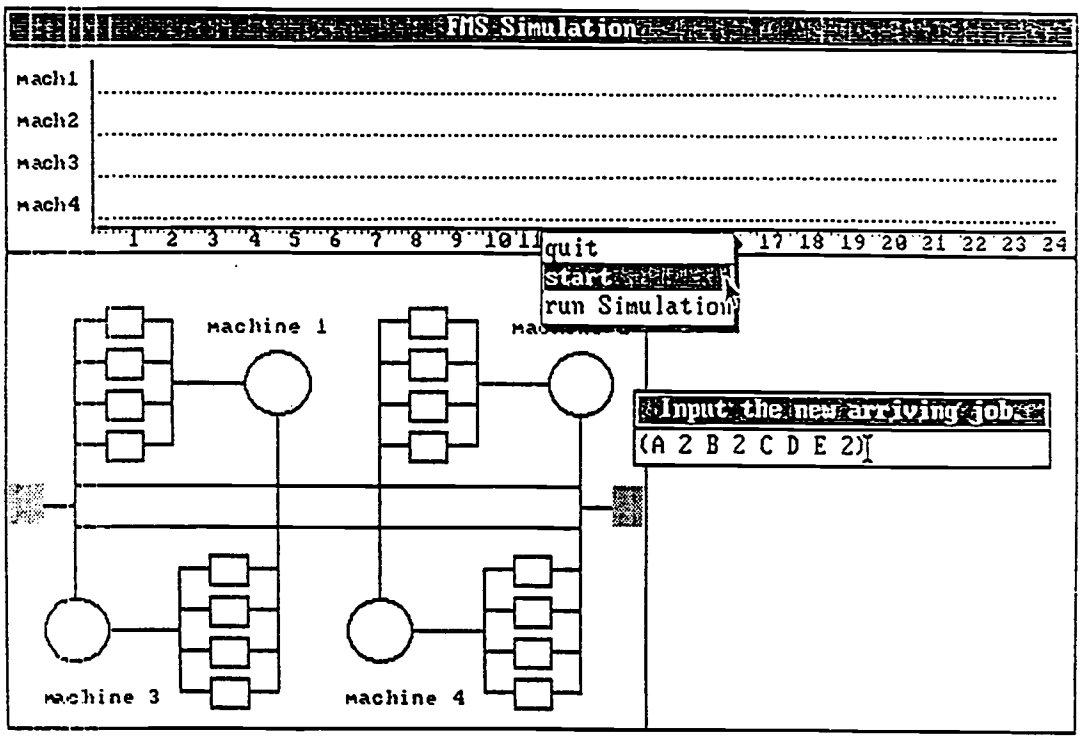


Figure 5.2 Initial stage of FMS simulation model

When the operator selects the "start" command from the menu, a prompt window asking for a new arriving job order is popped-up (Figure 5.2). The operator inputs the new job order, in the example, (A 2 B 2 C D E 2), meaning that the newly arriving job order contains two of part A, two of part B, and one of part C and D, and two of part E.

The job-matrix information window then appears on the screen (Figure 5.3), also consisting of three panes. The upper pane displays job processing information, the lower left pane displays the newly arriving job order, and the lower right pane displays the selected jobs in sequence. This window provides a working ground for determining the loading sequence of the new job order for the scheduler. As shown in Figure 5.3, job information is displayed in a matrix form. For example, the operating sequence of part J is machining center #1 → #3 → #2 → #4 and the processing time is 60, 90, 120, and 150 minutes, respectively.

The operator can select the next job by either analyzing the main matrix, which shows the total workload at each machining center, and comparing the job processing information among the waiting jobs, or by consulting the loading expert system. If the operator selects the next job manually, the system asks the reason why the job is selected. The reason, with current system information, is stored in a file for later

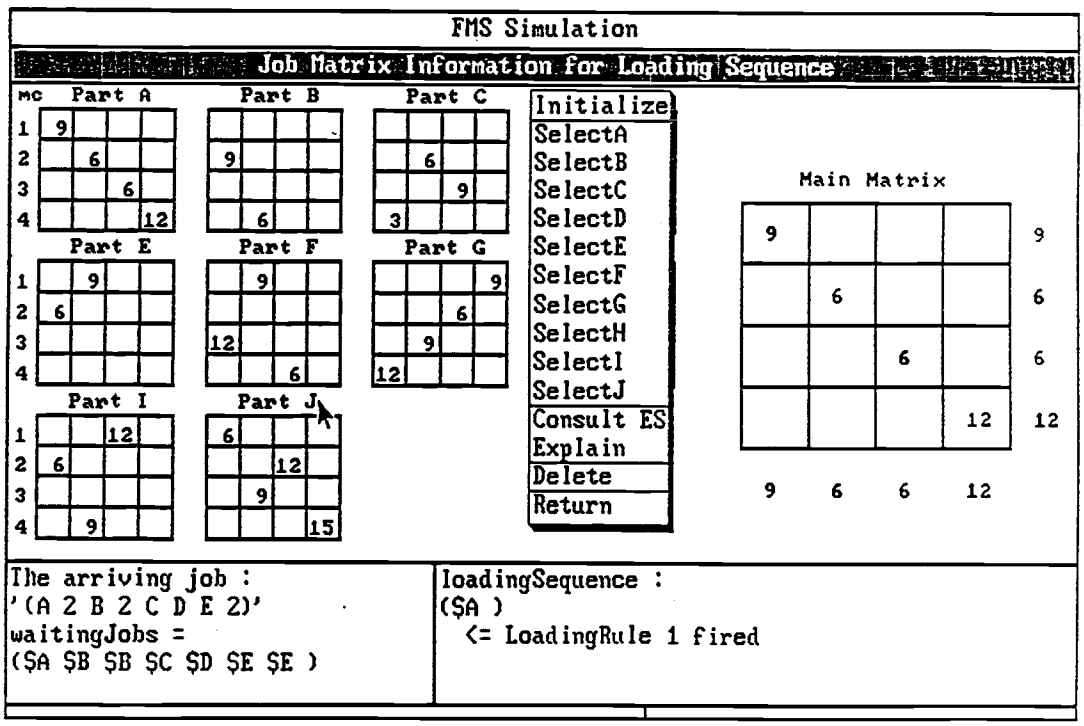


Figure 5.3 Job-matrix information window.

analysis. Each time the operator selects the next job, by either of these two methods, updated information on the selected jobs in sequence and the total workload at each machine in the main matrix are displayed. Figure 5.3 shows what the scheduler may see at one time while determining the loading sequence. The commands in the menu shown in Figure 5.3 are described in Table 5.12.

Table 5.12 Job-matrix information commands.

Initialize	Initializes the procedure and shows the arriving job in the lower left pane.
SelectA • • • SelectJ	Selects the corresponding job, part A through part J, one at a time, and updates the main matrix; if the operator uses these commands, the system asks the operator the reason for the selection.
Consult ES	Selects the next job by consulting the loading expert system.
Explain	Explains the reason why the last job was selected.
Delete	Deletes the last selected job and updates the main matrix.
Return	Returns to the main window.

After determining the loading sequence, the operator returns to the main window (Figure 5.4). Before the simulation can be run, the system asks the operator for a dispatching rule to be used in the simulation. A list of dispatching rules is shown in Table 5.13. As in the previous step, the operator can select a dispatching rule or ask the dispatching expert system to select one. In addition to the six dispatching rules, there are two other commands in Figure 5.4: "ExSys"

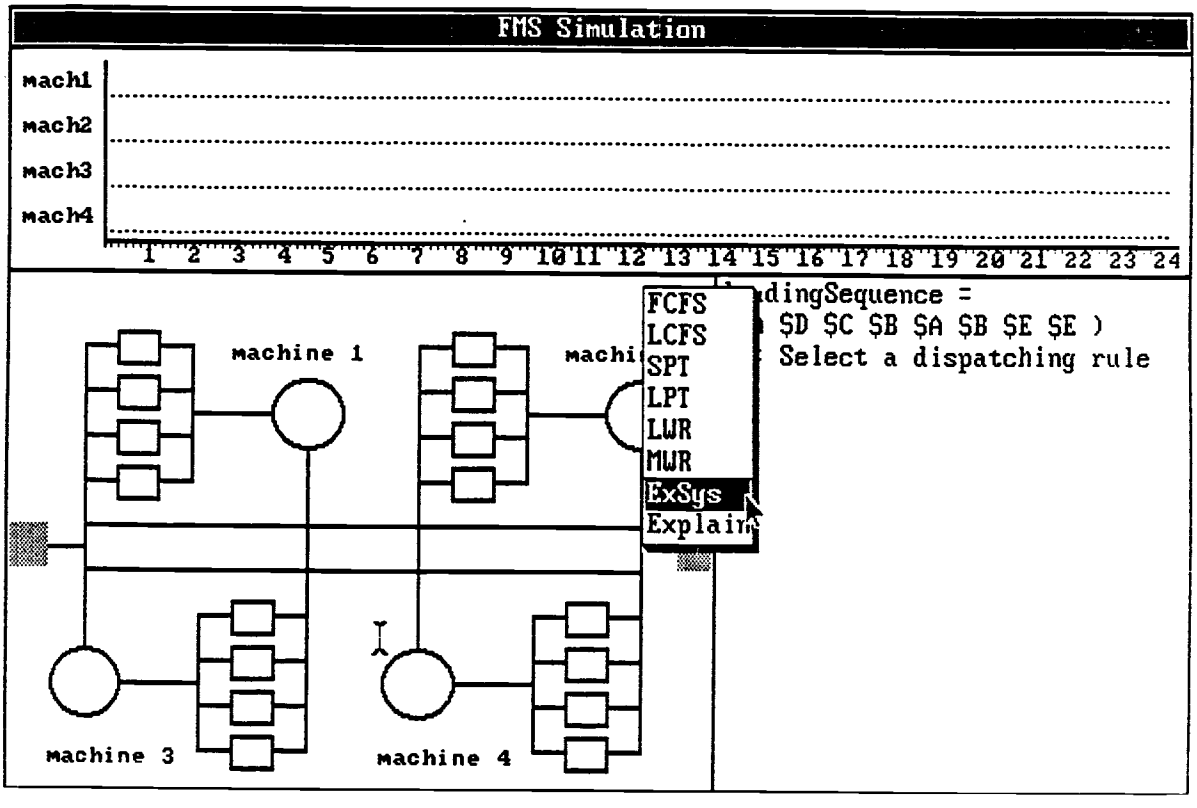


Figure 5.4 Selecting a dispatching rule.

Table 5.13 Dispatching rules in Figure 5.3.

FCFS	First Come First Served
LCFS	Last Come First Served
SPT	Shortest Processing Time job first
LPT	Longest Processing Time job first
LWR	Least Work Remaining job first
MWR	Most Work Remaining job first

consults the dispatching expert system and selects the one among the six rules which will usually minimize total processing time; "Explain" pops-up the explanation window and shows the reason why the dispatching rule was selected.

As the system runs the simulation, it shows the position of each part in the system layout pane and draws a Gantt chart in the Gantt chart schedule pane. At the same time it also prints the time and simulation event in the event-information pane. Figure 5.5 shows the end of the simulation. In the example, the total processing time was 620 minutes for the job order (A 2 B 2 C D E 2), with a loading sequence of (A D C B A B E E) using the MWR dispatching rule. The results of simulation runs using other dispatching rules for two other examples are shown in Table 5.14. Clearly,

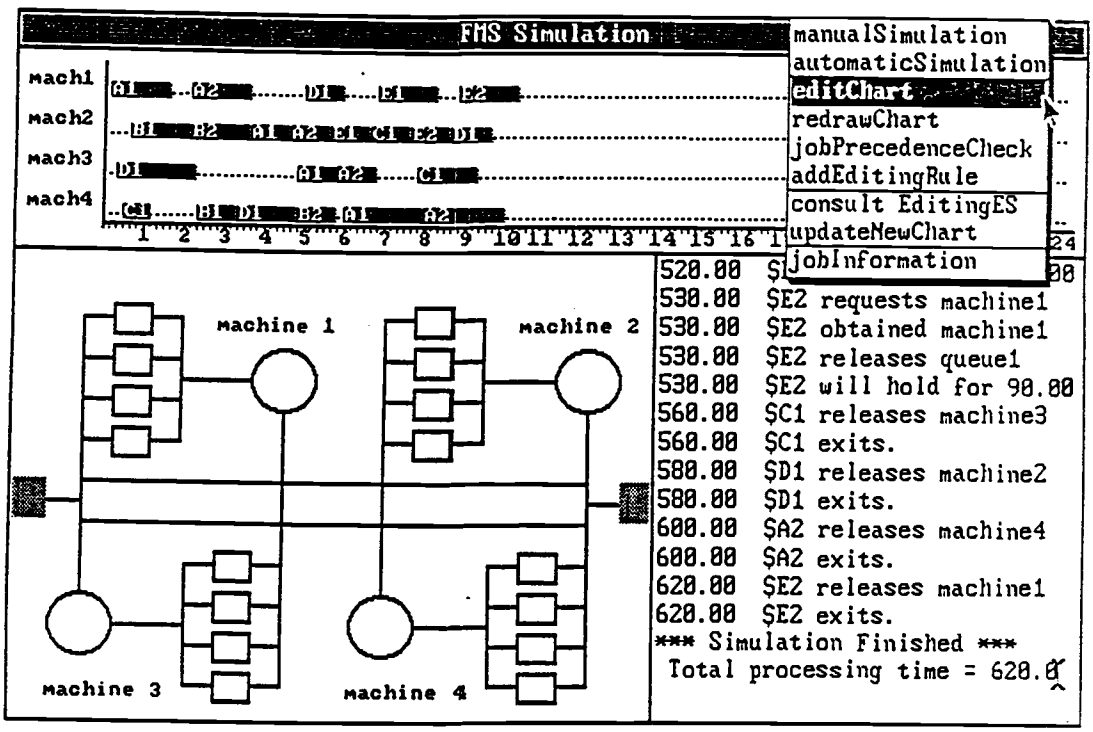


Figure 5.5 End of simulation.

Table 5.14 Results of simulation.

Job Order Loading Sequence	(A 2 B 2 C D E 2) (A D C B A B E E)	(D 2 G 3 I 2 J 2) (D G I J D G I J G)
FCFS	780	1110
LCFS	650	1310
SPT	750	1040 ← min
LPT	780	1210
LWR	780	1310
MWR	620 ← min	1110

selection of the best dispatching rule is dependent upon job characteristics.

The purpose of determining the loading sequence and selecting a better dispatching rule for a given job order is to generate a good initial schedule for the operator, since it will subsequently reduce the operator's effort when generating the final schedule. The initial schedule is shown in the Gantt chart schedule pane in Figure 5.5. Since this schedule was generated through the simulation, we can be certain that it is a feasible schedule. The Gantt chart schedule pane also contains nine commands in the menu, as shown in Figure 5.5. The function of each command is described in Table 5.15.

Table 5.15 Commands in Gantt chart pane.

manualSimulation	This command simulates the system according to the Gantt chart schedule as the scheduler moves the cursor by hand along the time axis. In this way, the operator can control the speed of the simulation.
automaticSimulation	Instead of manually updating the clock, the system automatically updates the clock and simulates the system according to the schedule. The purpose of simulation is to check the feasibility of new schedules.
editChart	This allows the operator to change the schedule. The scheduler can pick up a block of the schedule and move it to another place, using the mouse.
redrawChart	Once a schedule has been changed, the operator should redraw the Gantt chart to update its schedule. As it redraws, it also checks the overlaps of the blocks of jobs in the Gantt chart and locates them in the correct place. In this way, the operator need not worry about moving the block to the exact place; this command will do it for the operator.
jobPrecedenceCheck	The redrawChart command cannot check the job precedence constraint. This command checks all jobs for their precedence constraints in the schedule. If some job violates its precedence constraints, the command reports it to the event-information pane, e.g., "A3 - Job precedence constraint violated !!"
addEditingRule	Once selected, this command asks the operator the specific condition of the current state of the system and action the operator performed, information saved in a file, along with the general condition of the system.
consult EditingES	This command first opens another window for a new schedule and consults the editng expert system to update the existing schedule.
updateNewChart	This command updates the edited schedule and redraws it on the main window.
jobInformation	This command pops-up a separate window for job information in the event-information pane. The scheduler can access this information any time during editing.

By observing the Gantt chart, the operator can determine how to improve the initial schedule to minimize

total processing time. For example, since there are gaps between D1 and E1, and between E1 and E2 at machining center #1, the operator may check whether he can reduce or eliminate the gaps. To reduce the gaps, the operation of E1 and E2 at machining center #1 should be initiated earlier. This means that the previous operations of part E at machining center #2 should be performed earlier. Since job B2 does not affect total processing time whether it is processed earlier or later, the operator can delay the processing of part B and process other jobs ahead of it. This results in the final schedule shown in Figure 5.6.

While the operator edits the initial schedule, changes can be recorded in a file by selecting the "addEditingRule" menu. Then, the system asks the operator for the condition of the system and the action to be performed. The system stores these comments in a file with the current state of the system.

Instead of manual editing, the operator can consult the editing expert system by selecting "consult EditingES" (see Figure 5.5). Then, another window is opened for a new schedule and the operator can compare the new schedule with the old one as the editing expert system edits the Gantt chart schedule, as shown in Figure 5.7.

The simulation results show that the total processing time is 590 minutes, which is 30 minutes less

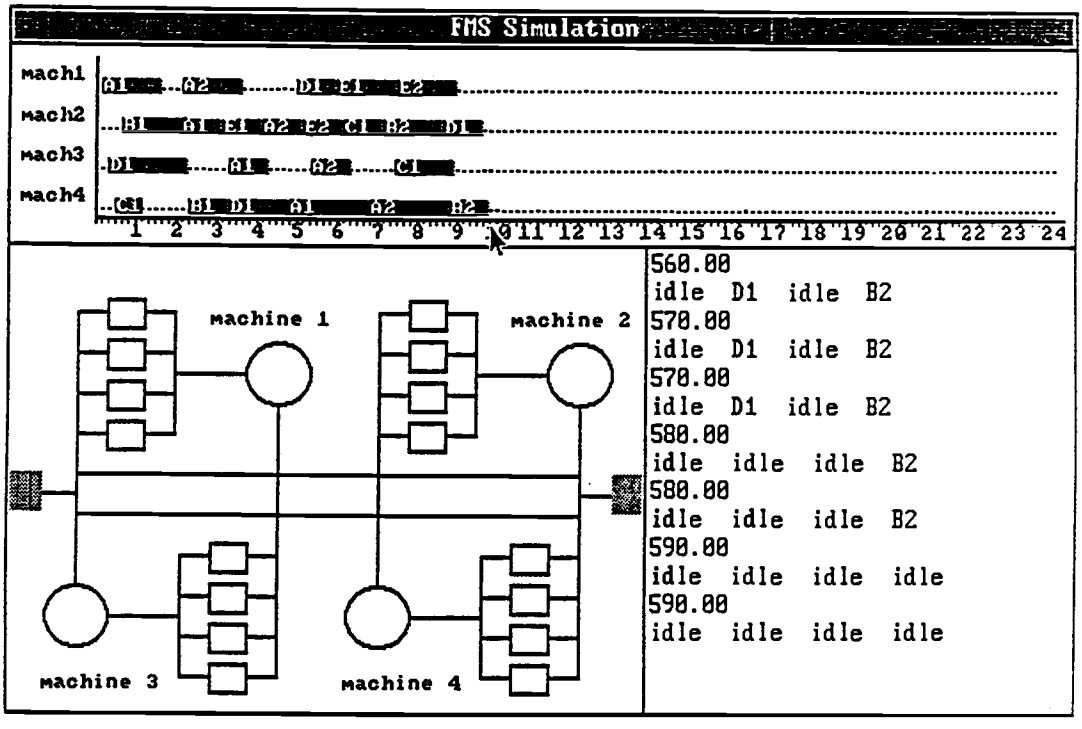


Figure 5.6 Final schedule.

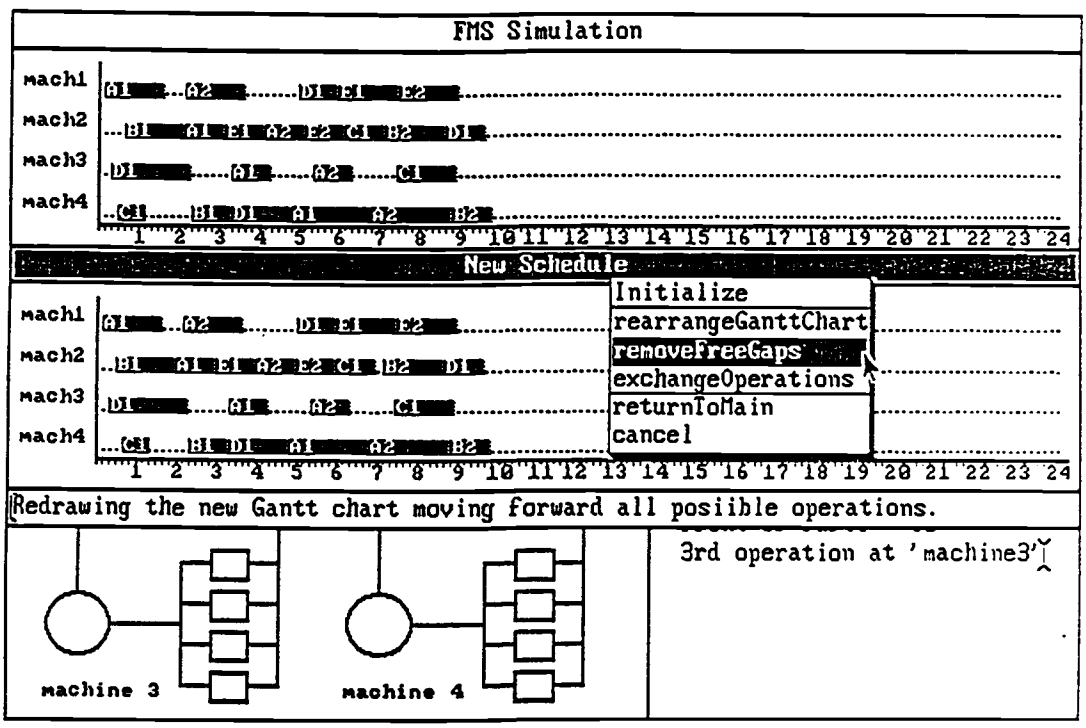


Figure 5.7 Consulting the editing expert system.

than the best result of using the MWR dispatching rule shown in Table 5.14.

5.4 Rules

Expert system rules for determination of initial loading sequence and for editing the Gantt chart were developed by having several graduate students in the Department of Industrial and Manufacturing Engineering at Oregon State University conduct trials with the "Schedule Generator" (Kim, 1988c).

The interactive graphic simulation model of the "Schedule Generator" (Kim et al., 1988a), which facilitated the knowledge acquisition process, was given to each subject, who in turn was asked to record actions to be taken in determining the loading sequence. Each time the subjects selected the next job among those waiting in the job order (see Figure 5.3), the system asked the subject to provide reasons for the selection. The typed-in reasons and current system status were saved in a text file for later analysis.

These comments were analyzed by the system developer, converting them into the form of production rules. For instance, most of subjects agreed that the job with the longest processing time should be selected first since the scheduling objective was to minimize total processing time and this method usually minimized

total processing time. This reason was converted to the first loading rule, "selectJobWithTotalProcessing Time", shown in Table 5.16. Once the first job was selected, subjects usually selected the next job to start its first operations at an idle machine. Reasons provided by the subjects were converted to the second loading rule, "selectJobForIdleMachine", shown in Table 5.16. Other rules were similarly developed and several key attributes in the data base dictionary for the loading expert system were considered. Their descriptions are shown in Table 5.17.

After the loading expert system was built, rules for selection of dispatching rules (see Table 5.18) were collected by running the simulation for different job orders based upon use of six different dispatching rules. For this purpose, the system was operated in automatic mode without the intervention of a human operator. Table 5.19 shows a small part of the results of this simulation, which were analyzed for consistency with other existing rules, generalized in production rule form, and added to the knowledge base of the dispatching expert system when found to be appropriate.

After the loading and dispatching expert systems were built, subjects were asked to find reasonable rules for adjusting and manipulating the Gantt chart schedule. It has been alleged that human operators,

Table 5.16 Rules for determining the loading sequence.

```
LoadingRule
   number:          1
   condition:       [ #(objective minimizeTotalProcessing Time)
                    isFact  & #(waitingJob notNil)isFact
                      & #(selectedJob nil) isFact ]
   action:          [ #selectJobWithTotalProcessingTime fireRule ]
   description:     'select the first job in the job order"

LoadingRule
   number:          2
   condition:       [ #(objective minimizeTotalProcessing Time)
                    isFact  & #(waitingJob notNil) isFact
                    & #(machineHasStartingJob
                       (status status status status)) isFact ]
   action:          [ #selectJobForIdleMachine fireRule ]
   description:     'select the next job starting at the machine of
                    machineHasStartingJob = false"

LoadingRule
   number:          3
   condition:       [ #(objective minimizeTotalProcessingTime)
                    isFact          & #(waitingJob notNil) isFact
                    & #(numberOfJobFinishedAtMachine
                       (number number number number)) isFact ]
   action:          [ #selectJobByLastOperation fireRule ]
   description:     'select the next job which finishes its last
                    operation at the machine having the maximum value
                    of array  numberOfJobFinishedAtMachine"

LoadingRule
   number:          4
   condition:       [ #(numberOfWaitingJobType 1) isFact
   action:          [ #selectAllRemainingJob fireRule ]
   description:     'select all the remaining job in the job order
                    since they are all same type"

LoadingRule
   number:          5
   condition:       [ #(waitingJob Nil) isFact ]
   action:          [ #returnToMainSystem fireRule ]
   description:     'all remaining jobs are selected"

LoadingRule
   number:          6
   condition:       [ #(waitingJob notNil) isFact
                    & #(ruleFired false) isFact]
   action:          [ #askScheduler fireRule ]
   description:     'ask the scheduler the next job since there is not
                    proper rule to select the next job"
```

Table 5.17 Key attributes in the data base dictionary for the loading expert system.

totalWorkLoad	Array in which each element shows the total work load at each machine.
machineHasStartingJob	Array in which each element shows a machine status whether the machine has starting job or not.
numberOfJobFinishedAtMachine	Array which shows how many jobs are finished at each machine.
numberOfWaitingJobs	Shows the number of waiting jobs in the job order.
numberOfSelectedJobs	Shows the number of selected jobs in the loading sequence.
numberOfPartA	Shows how many part A are in the job order.
numberOfPartJ	Shows how many part J are in the job order.

Table 5.18 Rules for selecting a dispatching rule.

```
DispatchingRule
   number:          1
   condition:       [ #(numberOfJob 2) isFact ]
   action:          [ #selectMWR fireRule ]
   description:     'select the MWR rule"

DispatchingRule
   number:          2
   condition:       [ #(numberOfJob 3) isFact ]
                    & #(numberOfPartBMoreThan 2) isFact ]
   action:          [ #selectLCFS fireRule ]
   description:     'if the job order includes more than two B and the
                    number of jobs is 3, then select the SPT rule"

DispatchingRule
   number:          3
   condition:       [ #(jobOrderInclude (A B D E)) isFact
   action:          [ #selectSPT ]
description:        'if the number of jobs is 4 and the job order
                    includes (A B D E), then select the SPT rule"

DispatchingRule
   number:          4
   condition:       [ #(jobOderInclude (B C C E)) isFact
   action:          [ #selectLCFS ]
   description:     'if the number of jobs is 4 and the job order
                    includes (B C C E, then select the LCFS rule"

DispatchingRule
   number:          5
   condition:       [ #(jobOrderInclude (C C E E)) isFact ]
   action:          [ #selectLCFS ]
   description:     "if the number of jobs is 4 and the job order
                    includes (C C E E), then select the LCFS rule"

DispatchingRule
   number:          6
   condition:       [ #(numberOfJob 4) isFact
                    & #(numberOfPartBMoreThan 2) isFact ]
   action:          [ #selectLCFS fireRule ]
   description:     'if the job order includes more than two B and the
                    number of jobs is 3, then select the SPT rule"

DispatchingRule
   number:          7
   condition:       [ #(jobOrderInclude (A B C C E)) isFact ]
   action:          [ #selectLCFS fireRule ]
   description:     'if the job order includes (A B C C E) and the
                    number of jobs is 5, then select the LCFS rule"

DispatchingRule
   number:          8
   condition:       [ #jobOrderInclude (A B C D E)) isFact ]
   action:          [ #selectLCFS fireRule ]
   description:     'if the jub order includes (A B C D E) and the
                    number of jobs is 5, then select the LCFS rule"
   .
   .
   .
DispatcingRule
   number:          94
   condition:       [ #(ruleSelected false) isFact ]
   action:          [ #selectMWR ]
   description:     'if rule is not selected, then select the MWR rule
                    as a default rule"

(Other rules are similar to the above rules.)
```

Table 5.19 Partial results of the simulation.

Job No.	Job order	Job sequence	FCFS	LCFS	SPT	LPT	LWR	MWR
147	(A B B C C C C)	(A C B C C C B)	670.0	610.0	620.0	830.0	800.0	670.0
148	(A B B C C C D)	(A D C B C C B)	660.0	590.0	630.0	750.0	750.0	620.0
149	(A B B C C C E)	(A C B C C B E)	680.0	620.0	650.0	680.0	770.0	640.0
150	(A B B C C D D)	(A D C B B D C)	580.0	560.0	560.0	570.0	660.0	560.0
151	(A B B C C D E)	(A D C B B C E)	720.0	590.0	690.0	720.0	720.0	590.0
152	(A B B C C E E)	(A C B B C E E)	740.0	580.0	700.0	740.0	740.0	610.0
153	(A B B C D D D)	(A D C B D D B)	670.0	790.0	790.0	670.0	670.0	670.0
154	(A B B C D D E)	(A D C B B D E)	630.0	590.0	590.0	630.0	630.0	580.0
155	(A B B C D E E)	(A D C B B E E)	690.0	590.0	650.0	690.0	690.0	560.0
156	(A B B C E E E)	(A C B B E E E)	710.0	580.0	680.0	710.0	710.0	610.0
157	(A B B D D D D)	(A D B D D D B)	750.0	810.0	870.0	750.0	870.0	750.0
158	(A B B D D D E)	(A D B D D B E)	660.0	720.0	780.0	660.0	660.0	660.0
159	(A B B D D E E)	(A D B B D E E)	650.0	550.0	560.0	650.0	650.0	590.0
160	(A B B D E E E)	(A D B B E E E)	680.0	580.0	620.0	650.0	650.0	620.0
161	(A B B E E E C)	(A B B E E E E)	700.0	570.0	670.0	700.0	700.0	660.0

utilizing some form of Gantt chart, can employ various heuristic adjusting procedures which give them an advantage over machine competition (Conway et al., 1967). Although the system made it possible for the subjects to group orders and easily distribute them over the available machines, while building schedules that were better than the initial schedule generated through the loading and dispatching expert systems, considerable time was still required for analysis of the Gantt chart schedules. Some of subjects suggested that the process would have been facilitated if the system had provided automatic functions or rules for the adjustment of existing schedules at the operator's request. Since the subjects usually checked whether the last operation in the schedule could be initiated

earlier, or whether the gap before the last operation could be reduced, in order to further minimize total processing time, a function was programmed to determine the gap and rearrange the Gantt chart ("rearrangeGantt Chart" in Table 5.20). Once the schedule was changed, new gaps developed. These were reduced or eliminated without constraints by development of the "removeFree Gaps" shown in Table 5.20.

The subjects frequently tried to exchange the sequence of operations at specific machines to determine whether or not this would reduce total processing time. This function was added to the expert editing system ("exchangeOperations" in Table 5.20) as rules which are fired at the operator's request. Even though it is relatively easy for a human operator to determine a pattern for firing a rule when using the Gantt chart schedule, it is extremely difficult to explicitly state the conditions for rule-firing.

Through further testing and experimentation some very specific firing rules were developed, including edit rules 4 and 5 in Table 5.20. These rules exchange the particular operations at a machine if specific conditions are met. Even though it is possible to generalize the kind of specific rules listed in Table 5.20, building explicit statements of reasonably general rules for manipulation of the Gantt chart is still a very difficult task. Additional experimentation and

Table 5.20 Rules for editing the Gantt chart.

```
EditRule
   number:          1
   condition:       [ #(objective minimizeTotalProcessing Time) isFact
                    & #(gapInSchedule true) isFact ]
   action:          [ #rearrangeGanttChart fireRule ]
   description:     'if the scheduling objective is to minimize the
                    total processing time and there is a gap in the
                    schedule, then check the gap whether it is
                    reducible or not"

EditRule
   number:          2
   condition:       [ #(gapReducible true) isFact ]
   action:          [ #removeFreeGaps fireRule ]
   description:     'if there is a reducible gap, reduce the gap and
                    rearrange the new Gantt chart"

EditRule
   number:          3
   condition:       [ #(gapReducible false) isFact
                    & #(switchableJob found) isFact ]
   action:          [ #exchangeOperations fireRule ]
   description:     'if there is no reducible gap, then ask operator
                    which two jobs will be exchanged and receiving an
                    input from operator, perform operator's direction"

EditRule
   number:          4
   condition:       [ #(objective minimizeTotalProcessingTime) isFact ]
                    & #(numberOfJob 7) isFact
                    & #(hasSequenceInMachine (part A part B machine 2))
                    isFact ]
   action:          [ #swtichOpeations fireRule ]
   description:     'if part A is processed before part B at machine 2,
                    switch the operation"

EditRule
   number:          5
   condition:       [ #(objective minimizeTotalProcessingTime) isFact ]
                    & #(numberOfJob 7) isFact ]
                    & #(hasSequenceInMachine (part D part B machine 4)
                    isFact ]
   action:          [ #switchOperations fireRule ]
   description:     'if part D is processed before part B at machine 4,
                    switch the operation"

(Other rules are similar to rules 4 and 5 and additional rules will be
                    generated through experimentation.)
```

study will be required to help operators articulate the condition state and generalize a group of specific rules.

6.0 SUMMARY, CONCLUSIONS, AND SCOPE FOR FUTURE RESEARCH

This paper presents an expert system for FMS scheduling problems, as well as an interactive graphics-based computer model for a knowledge acquisition method which effectively utilizes human pattern-recognition abilities. Through testing and implementation of the model, it was found that a human scheduler can obtain an optimum or near-optimum schedule in short periods of time, while gaining valuable experience and knowledge in scheduling problems. Furthermore, it was determined that this model can be a useful training device for inexperienced schedulers and a decision-making aid for expert schedulers.

Four different areas for future research have been identified. First, the relaxation of assumptions imposed in this study can be considered, namely:

1) The use of other scheduling objectives, such as minimizing the number of late jobs or minimizing average job tardiness, or the use of multiple objectives;
2) Expansion of the scheduling time horizon;

3) Consideration of other system constraints, such as the number of pallets, fixtures, tools, and the material handling system;

4) Allow flexible job processing sequences; and

5) Allow batch processing and job splitting.

Dealing with different types of constraints will generate new knowledge and alternative sets of rules.

Second, the model developed for this study can be expanded and be applied to the real-time control of an FMS. Operations in an FMS are initiated one-by-one, taking instructions from a planned schedule while progress is continuously checked. The actual progress data for operations are compared with the planned schedule each time a new operation is begun and is completed. If the difference exceeds a specified limit, or if certain abnormal events occur, such as a machine breakdown, a signal is issued alerting the scheduler and a new schedule is generated which considers current system status.

Third, an automatic rule generation program should be considered. At present, the rules are collected, analyzed, and added to the knowledge base of an expert system by the human operator. Developing a type of induction program capable of inducing general principles or rules from a set of specific examples, in order to automate the manual rule-adding process, will constitute a challenging research problem.

Finally, the approach implemented in this study can be applied to other problem domains, such as process planning, other production controls in addition to scheduling, and to fault diagnosis problems.

BIBLIOGRAPHY

Ammons, J.C., 1985, "Scheduling Models for Aiding Real Time Control," IEEE CH2253-3/85, pp185-189.

Baker, K.R., 1974, Introduction to Sequencing and Scheduling, John Wiley, New York.

Ballakur, A. and Steudal, H., 1984, "Integration of Job shop Control Systems: State-of-the-art Review,", J. Mfg. Sys. Vol. 3, No. 1, pp71-79.

Barr, A. and Feigenbaum, E.A.(eds), 1981, The Handbook of Artificial Intelligence, Vol. 1, William Kaufmann.

Bell, M.Z., 1985, "Why Expert Systems Fail," J. Opl. Res. Soc., Vol. 36, No. 7, pp613-619.

Bernold, T., 1985, Artificial Intelligence in Manufacturing, North Holland.

Blackstone, J.H., Jr, Phillps, D. T., and Hogg, G.L., 1982, "A State-of-the-art Survey of Dispatching Rules for Manufacturing Job shop Operations," Int. J. Prod. Res., Vol. 20, No. 1, pp27-45.

Blessing, J.A., and Watford, B.A., 1987, "INFMSS: An Intelligent FMS Scheduling System," IEE Conference Proceedings on World Productivity Forum, pp476-482.

Bowman, E. H., 1959, "The Schedule-Sequencing Problem," Operations Res. Vol. 7, pp621-624.

Bruno, G., Elia, A., and Laface, P., 1986, "A Rule-Based System to Schedule Production," Computer, Vol. 19, No. 7, July, pp32-39.

Bullers, W. I., Nof, S. Y., and Whinston, A. B., 1980, "Artificial Intelligence in Manufacturing Planning and Control," AIIE Trans., Dec., Vol. 12, No. 4, pp351-363.

Buchanan, B. G. and Shortliffe, E. H., 1985, Rule-Based Expert Systems, Addison-Wesley.

Buzacott, J. A., 1982, "Optimal Operating Rules for Automated Manufacturing Systems," IEEE Trans.: Automation and Control, Vol. AC-27, No. 1, pp80-86.

Buzacott, J. A. and Shanthikumar, J.G., 1980, "Models for Understanding Flexible Manufacturing Systems," AIIE Trans., Vol. 12, pp339-350.

Buzacott, J.A., and Yao, D. D., 1986, "Flexible Manufacturing Systems: A Review of Analytical Models," Management Science, Vol. 32, No. 7, July, pp890-905.

Carter, C. F., Jr, 1971, "Trends in Machine Tool Development and Application," Proceeding of the Second International Conference on Product Development and Manufacturing Technology," University of Strathclyde, pp125-141.

Chen, P. H. and Talavage, J., 1982, "Production Decision Support System for Computerized Manufacturing Systems," J. Mfg. Sys., Vol. 1, No. 2, pp157-166.

Clocksin, W. F. and Mellish, C. S., 1981, Programming in Prolog, Springer-Verlarg.

Conway, R. W., Maxwell, W. L., and Miller, L. W., 1967, Theory of Scheduling, Addison-Wesley, Reading, Massachusetts.

Cook, N. H., 1975, "Computer-Managed Parts Manufacturing," Scientific American, Feb., pp21-29.

Darrow, W. P., 1986, "A Survey of Flexible Manufacturing Systems Implementations," US Department of Commerce, National Bureau of Standards (NBSIR86-3413).

Day, J. E. and Hottenstein, M. P., 1970, "Review of Sequencing Research," Naval Research Logistics Quarterly, Vol. 17, pp11-40.

Denzler, D. R. Boe, W. J., 1987, "Experimental Investigation of Flexible Manufacturing System Scheduling Decision Rules," Int. J. Prod. Res., Vol. 25, No. 7, pp979-994.

Duda, R. O., Gashing, J. G., Hart, P. E., Konolige, K., Reboh, R., Barret, P., and Slocum, J., 1978, "Development of the PROSPECTOR Consultation System

for Mineral Exploration," Final Report, SRI project 5821 and 6415, SRI International Inc., Melno Park, CA, USA.

Dupont-Gatelmand, C., 1982, "A Survey of Flexible Manufacturing Systems," J. Mfg. Sys., Vol. 1, No. 1, pp1-16.

ElMaraghy, H. A., 1982, "Simulation and Graphical Animation of Advanced Manufacturing Systems," J. Mfg. Sys., Vol. 1, No. 1, pp53-63.

Falster, P. and Mazumder, R. B., 1984, Modelling Production Management Systems, NorthHolland.

Faught, W. S., 1986, "Application of AI in Engineering," Computer, Vol. 19, No. 7, July, pp17-31.

Ferguson, R. L. and Jones, C. H., 1969, "A Computer Aided Decision System," Management Science, Vol. 15, No. 10, June, pp550-561.

Fikes, R. and Kehler, T., 1985, "The Role of Frame-based Representation in Reasoning," CACM, Vol. 28, No. 9, Sep., pp904-920.

Forgy, C. L., 1981, "The OPS5 User's Manual," Technical Report CMU-CS-81-135, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA.

Fox, M. s., 1983, "Constraint-Directed Search: A Case Study of Job-Shop Scheduling," CMU-RI-TR-83-22/CMU-CS-83-161, Carnegie Mellon University, PhD Thesis.

Fox, M. S. and Smith, S. F., 1984, "A Knowledge-based System for Factory Scheduling," Expert Systems Journal, Vol. 1, No. 1, July, pp25-40.

French, S., 1982, Sequencing and Scheduling: An Introduction to the Mathematics of the Job-shop, John Wiley & Sons.

Godin, V. B., 1978, "Interactive Scheduling: Historical Survey and State of the art," AIIE Trans., Vol. 10, No. 3, Sep., pp331-337.

Godin, V. B. and Jones, C. H., 1969, "The Interactive Shop Supervisor," Industrial Engineering, Vol. 1, No. 11, Nov., pp16-22.

Goldhar, J. D. and Jelinek, M., 1985, "Computer Integrated Flexible Manufacturing: Organizational, Economic, and Strategic Implications," Interfaces, Vol. 15, No. 3, May-June, pp94-105.

Goldberg, A. and Robson, D., 1983, Smalltalk-80: The Language and Its Implementation, Addison-Wesley.

Graves, C. S., 1979, "A review of Production Scheduling: Theory and Practice," Technical Report 169, Operations Research Center, MIT.

Groover, M. P. and Zimmers, E. W., Jr., 1984, Computer Aided Design and Manufacturing, Prentice Hall.

Hart, A., 1985, "The Role of Induction in Knowledge Elicitation," Expert Systems Journal, Vol. 2, No. 1, Jan., pp24-28.

Hayes-Roth, F., Waterman, D. A., and Lenat, D. B., 1983, Building Expert Systems, Addison-Wesley.

Hodgson, T. J. and McDonald, G. W., 1981, "Interactive Scheduling of a Generalized Flowshop Part I: Success through Evolutionary Development," Interfaces, Vol. 11, No. 2, pp

Holloway, C. A. and Nelson, R. T., 1973, "Alternative Formulation of the Job-shop Problem with Due Dates," Management Science, Vol. 20, No. 1, Sep., pp65-75.

Hurrion, R. D., 1978, "An Investigation of Visual Interactive Simulation Methods Using the Job-Shop Scheduling Problem," J. Opl. Res. Soc., Vol. 29, No. 11, pp1085-1093.

Hurrion, R. D., 1980, "Visual Interactive (Computer) Solutions for the Travelling Salesman Problem," J. Opl. Res. Soc, Vol. 31, No. 6, pp537-539.

Ito, Y., 1981, "Japanes FMS - present and a future view," handout of seminar on FMS, R.O.C.

Jackson, J. R., 1957, "Networks of Waiting Lines," Ops. Res., Vol. 5, pp518-522.

Jaikumar, R., 1984, "Flexible Manufacturing Systems: A Managerial Perspective," working paper No. 1-784-078, Harvard Business School, USA.

Japanese Production Technology Investigation Society, 1981, Collection of Europe and American FMS, pp30-41.

Jones, C. V. and Maxwell, W. L., 1986, "A System for Manufacturing Scheduling with Interactive Computer Graphics," IIE Trans. Sep., pp297-303.

Johnson, S. M., 1954, "Optimal Two and Three Stage Production Schedules with Set-up Times Included," Naval Research Logistics Quarterly, Vol. 1, PP61-65.

Johnson, L. A. and Montgomery, D. C., 1974, Operations Research in Production, Scheduling, and Inventory Control, John Wiley and Sons, New York.

Kim, J., Funk, K. H., and Fichter, E. F., 1988a, "Towards an Expert System for FMS Scheduling: A Knowledge Acquisition Environment," Second International Conference on Expert System and Leading Edge in Production Planning and Control, May 3-5, Charleston, SC, USA.

Kim, J., Fichter, E. F., and Funk, K. H., 1988b, "Building an Expert System for FMS Scheduling," ASME conference, U.S.A.-Japan Symposium on Flexible Automation, July 18-20, Minneapolis, MN, USA.

Kim, J., 1988c, "Schedule Generator: Operator's Manual," Department of Industrial and Manufacturing Engineering, Oregon State University, Corvallis, OR, USA.

Kim, J., 1988d, Videotape (demonstration) for "Schedule Generator: Operator's Manual," Department of Industrial and Manufacturing Engineering, Oregon State University, Corvallis, OR, USA.

Kumara, S. R. T., Joshi, S., Kashyap, R. L., Moodie, C. L., and Chang, T. C., 1986, "Expert Systems in Industrial Engineering," Int. J. Prod. Res., Vol. 24, No. 5, pp1107-1125.

Kunz, J., 1978, "A Physiological Rule Based System for Interpreting Pulmonary Function Test Results," Heuristic Programming Project, Report No. HPP-78-19, Computer Science Department, Stanford University.

Kusiak, A., 1985, "Loading Models in Flexible Manufacturing Systems," in Raouf, A. and Ahmad, S. I. (eds), Flexible Manufacturing: Recent Developments in FMS, Robotics, CAD/CAM, CIM, pp119-132.

Lenstra, J. K. and Rinnooykan, A. H. G., 1978, "Complexity of Scheduling under Precedence Constraints," Operations Research, Vol. 26, No. 1, Jan-Feb. pp22-35.

Licklider, J. C. R., 1960, "Man-Computer Symbiosis," IRE Trans, on Human Factors in Electronics, HFE-I, Vol. 1, March, pp4-11

Lu, S. C-Y. and Komanduri, R. (eds), 1986, Knowledge-Based Expert Systems for Manufacturing, The American Society of Mechanical Engineers.

Manne, A. S., 1960, "On the Job-Shop Scheduling Problem," Operations Research, Vol. 7, pp219-223.

McDermott, J., 1980, "R1: A Rule Based Configurer of Computer Systems," Technical Report CMU-CS-80-119, Carnegie-Mellon University.

Minsky, M. 1975, "A Framework for Representing Knowledge," in Winston, P. (ed), The Psychology of Computer Vision, McGraw-Hill, New York.

Nau, D. S., 1983, "Expert Computer Systems," IEEE Computer, Feb., pp63-73.

Newell, A. and Simon, H. A., 1972, Human Problem Solving, Prentice-Hall, Englewood Cliffs, NJ

Nicholas, J. M. and Yazdanian, K., 1978, "Integrity Checking in Deductive Databases," in Gallaire, H. and Minker, J. (eds), Logic and Databases, Plenum press, New York.

Nilsson, N., 1971, Problem Solving Methods in Artificial Intelligence, McGraw Hill, New York.

Panwalker, S.S., and Iskander, W., 1977, "A Survey of Scheduling Rules," Operations Research, Vol. 25, No. 1, January-February, pp45-61.

Pople, H., 1977, "The Formation of Composite Hypotheses in Diagnostic Prolem SOlving - An Exercise in Synthetic Reasoning," Proceeding of IJCAI-5, Vol. 5, pp1030-1037.

Quillian, R., 1968, "Semantic Memory," in Minsky, M. (ed), Semantic Information Processing, MIT press, Cambridge.

Rachamadugu, R. and Stecke, K. E., 1986, "Classification and Review of FMS Scheduling Procedures," working paper, The University of Michigan, Graduate School of Business Administration, Ann Arbor, Michigan.

Ranky, P., 1983, The design and Operation of FMS, North-Holland

Reiter, R., 1978, "Deductive Question-Answering on Relational Database," in Gallaire, H. and Minker, J. (eds), Logic and Databases, Plenum pressm New York.

Rinnooykan, A.H.G., 1976, Machine Scheduling Problems: Classification, Complexity and Computations, Nijhoff, The Hague, Holland.

Shannon, R. E. and Phillips, D. T., 1983, "Comparison of Modelling Language for Simulation of Automated Manufacturing Systems," Autofact 5 conference proceedings, Society of Manufacturing Engineers.

Shanker, K. and Tzen, Y. J., 1985, "A Loading and Dispatching in a Random Flexible Manufacturing System," Int. J. Prod. Res., Vol. 23, No. 3, pp579-595

Shortliffe, E. H., 1976, Computer Based Medical Consultation: MYCIN, American Elsevier, New York.

Smith, M. L.,Ramesh, R., Dudek, R. A., and Blair, E. L., 1986, "Characteristics of US Flexible Manufacturing Systems - A Survey," in Stecke, K. E. and Suri, R. (eds), proceedings of the second ORSA/TIMS conferences on FMSs: Operations Research Models and Applications, Elsevier Science, Amsterdam, The Netherland, pp478-486.

Smalltalk/V: Tutorial and Programming Handbook, 1986, Digitalk Inc.

Solberg, J. J., 1977 "A Mathematical Models of Computerized Manufacturing Systems," proceedings, 4th international conference on Production Research, Tokyo, Japan.

Stecke, K. E. and Solberg, J. J., 1981, "Loading and Control Policies for an FMS," Int. J. Prod. Res., Vol. 19, No. 5, Sep-Oct., pp481-490.

Stecke, K. E., 1983, "Formulation and Solution of Non-linear Integer Production Planning Problems for FMS," Management Science, Vol. 29, No. 3, March, pp273-288.

Stefik, M., Aikius, J., and Balzer, R., 1982, "The Organization of Expert Systems, A Tutorial," AI, Vol. 18., pp135-173.

Steudel, H. J., 1986, "SIMSHOP: A Job Shop/Cellular Manufacturing Simulator," J. Mfg. Sys., Vol. 5, No. 3, pp181-189.

Suri, R. and Whitney, C. K., 1984, "Decision Support Requirements in Flexible Manufacturing," J. Mfg. Sys., Vol. 3, No. 1, pp61-69.

van Meile, W., Shortliffe, E. H., and Buchanan, B. G., 1981, "EMYCIN: A Domain Independent System that Aids in Constructing Knowledge Based Consultation Programs," Machine Intelligence, Infotech state of the art report 9, No. 3, pp281-287.

Wagner, H., 1959, "An Integer-Programming Model for Machine Scheduling," Naval Research Logistic Quartely, Vol. 6, pp131-139.

Waterman, D. A., 1986, A Guide to Expert Systems, Addison-Wesley.

Wiess, S. M. and Kulikowski, C. A., 1979, "EXPERT: A System for Developing Consultation Models," proceedings of IJCAI-6, pp942-947.

Winston, P.H., and Horn, B.K.S., 1984, LISP (2nd ed.), Reading, MA: Addison-Wesley Publishing Co.

APPENDICES

Appendix A

MPOS Program Code

```
S
1     MPOS VERSION 4.0        NORTHWESTERN UNIVERSITY
                  88/02/04.   04.26.24.          PAGE    1

          ****************************************
          *                                      *
          *              M P O S                 *
          *                                      *
          *            VERSION 4.0               *
          *                                      *
          * MULTI-PURPOSE OPTIMIZATION SYSTEM    *
          *                                      *
          ****************************************


 ***** PROBLEM NUMBER  1 *****

        BBMIP
        TITLE
        ...EXAMPLE JOB SHOP SCHEDULING PROBLEM...
        *
        * INTEGER VARIABLES
        *
        INTEGER
        YAB2 YAB4
        YAC2 YAC3 YAC4
        YAD1 YAD2 YAD3 YAD4
        YBA2 YBA4 YBC2 YBC4 YBD2 YBD4
        YCA2 YCA3 YCA4 YCB2 YCB4 YCD2 YCD3 YCD4
        YDA1 YDA2 YDA3 YDA4 YDB2 YDB4
        YDC2 YDC3 YDC4
        *
        * VARIABLES
        *
        VARIABLES
        TA1  TA2  TA3  TA4
        TB1  TB2  TB3  TB4
        TC1  TC2  TC3  TC4
        TD1  TD2  TD3  TD4
        FMAX
        *
        * OBJECTIVE FUNCTION
        *
```

```
     MINIMIZE
     FMAX
     *
     * CONSTRAINTS
     *
     CONSTRAINTS
     *
     * STARTING TIME OF LAST OPERATION + PROC. TIME > FMAX
     *
 1.  -TA4 + FMAX .GE. 12
 2.  -TB4 + FMAX .GE. 6
 3.  -TC3 + FMAX .GE. 9
 4.  -TD2 + FMAX .GE. 6
     *
     *
     * NONINTERFERENCE CONSTRAINTS
     *
     * I=A, J=B
 5.  109YAB2 + TA2 - TB2 .GE. 9
 6.  106YAB2 - TB2 + TA2 .LE. 100
 7.  106YAB4 + TA4 - TB4 .GE. 6
 8.  112YAB4 - TB4 + TA4 .LE. 100
     * I=A, J=C
 9.  106YAC2 + TA2 - TC2 .GE. 6
10.  106YAC2 - TC2 + TA2 .LE. 100
11.  109YAC3 + TA3 - TC3 .GE. 9
12.  106YAC3 - TC3 + TA3 .LE. 100
13.  103YAC4 + TA4 - TC4 .GE. 3
14.  112YAC4 - TC4 + TA4 .LE. 100
     * I=A, J=D
15.  106YAD1 + TA1 - TD1 .GE. 6
16.  109YAD1 - TD1 + TA1 .LE. 100
17.  106YAD2 + TA2 - TD2 .GE. 6
18.  106YAD2 - TD2 + TA2 .LE. 100
19.  112YAD3 + TA3 - TD3 .GE. 12
20.  106YAD3 - TD3 + TA3 .LE. 100
21.  109YAD4 + TA4 - TD4 .GE. 9
22.  112YAD4 - TD4 + TA4 .LE. 100
     * I=B, J=A
23.  106YBA2 + TB2 - TA2 .GE. 6
24.  109YBA2 - TA2 + TB2 .LE. 100
25.  112YBA4 + TB4 - TA4 .GE. 12
26.  106YBA4 - TA4 + TB4 .LE. 100
     * I=B, J=C
27.  106YBC2 + TB2 - TC2 .GE. 6
28.  109YBC2 - TC2 + TB2 .LE. 100
29.  103YBC4 + TB4 - TC4 .GE. 3
30.  106YBC4 - TC4 + TB4 .LE. 100
     * I=B, J=D
31.  106YBD2 + TB2 - TD2 .GE. 6
32.  109YBD2 - TD2 + TB2 .LE. 100
33.  109YBD4 + TB4 - TD4 .GE. 9
34.  106YBD4 - TD4 + TB4 .LE. 100
```

```
     * I=C, J=A
35.  106YCA2 + TC2 - TA2 .GE. 6
36.  106YCA2 - TA2 + TC2 .LE. 100
37.  106YCA3 + TC3 - TA3 .GE. 6
38.  109YCA3 - TA3 + TC3 .LE. 100
39.  112YCA4 + TC4 - TA4 .GE. 12
40.  103YCA4 - TA4 + TC4 .LE. 100
     * I=C, J=B
41.  109YCB2 + TC2 - TB2 .GE. 9
42.  106YCB2 - TB2 + TC2 .LE. 100
43.  106YCB4 + TC4 - TB4 .GE. 6
44.  103YCB4 - TB4 + TC4 .LE. 100
     * I=C, J=D
45.  106YCD2 + TC2 - TD2 .GE. 6
46.  106YCD2 - TD2 + TC2 .LE. 100
47.  112YCD3 + TC3 - TD3 .GE. 12
48.  109YCD3 - TD3 + TC3 .LE. 100
49.  109YCD4 + TC4 - TD4 .GE. 9
50.  103YCD4 - TD4 + TC4 .LE. 100
     * I=D, J=A
51.  109YDA1 + TD1 - TA1 .GE. 9
52.  106YDA1 - TA1 + TD1 .LE. 100
53.  106YDA2 + TD2 - TA2 .GE. 6
54.  106YDA2 - TA2 + TD2 .LE. 100
55.  106YDA3 + TD3 - TA3 .GE. 6
56.  112YDA3 - TA3 + TD3 .LE. 100
57.  112YDA4 + TD4 - TA4 .GE. 12
58.  109YDA4 - TA4 + TD4 .LE. 100
     * I=D, J=B
59.  109YDB2 + TD2 -TB2 .GE. 9
60.  106YDB2 - TB2 + TD2 .LE. 100
61.  106YDB4 + TD4 - TB4 .GE. 6
62.  109YDB4 - TB4 + TD4 .LE. 100
     * I=D, J=C
63.  106YDC2 + TD2 - TC2 .GE. 6
64.  106YDC2 - TC2 + TD2 .LE. 100
65.  109YDC3 + TD3 - TC3 .GE. 9
66.  112YDC3 - TC3 + TD3 .LE. 100
67.  103YDC4 + TD4 - TC4 .GE. 3
68.  109YDC4 - TC4 + TD4 .LE. 100
     *
     * PRECEDENCE CONSTRAINTS
     *
69.  -TA1 + TA2 .GE. 9
70.  -TA2 + TA3 .GE. 6
71.  -TA3 + TA4 .GE. 6
72.  -TB2 + TB4 .GE. 9
73.  -TC4 + TC2 .GE. 3
74.  -TC2 + TC3 .GE. 6
75.  -TD3 + TD4 .GE. 12
76.  -TD4 + TD1 .GE. 9
77.  -TD1 + TD2 .GE. 6
     *
```

```
*
BOUNDS
YAB2,YAB4,YAC2,YAC3,YAC4,YAD1,YAD2,YAD3,YAD4 .LE. 1
YBA2,YBA4,YBC2,YBC4,YBD2,YBD4 .LE. 1
YCA2,YCA3,YCA4,YCB2,YCB4,YCD2,YCD3,YCD4 .LE. 1
YDA1,YDA2,YDA3,YDA4,YDB2,YDB4,YDC2,YDC3,YDC4 .LE. 1
PRINT
OPTIMIZE
```

Appendix B

Results of MPOS

```
1     MPOS VERSION 4.0        NORTHWESTERN UNIVERSITY
                  88/02/04.   04.26.24.          PAGE   46

 **********************
 * PROBLEM NUMBER  1 *
 **********************

 USING BBMIP
 ...EXAMPLE JOB SHOP SCHEDULING PROBLEM...


 --- NEW INTEGER-FEASIBLE SOLUTION ---
```

SUMMARY OF RESULTS AT ITERATION 194

OBJECTIVE FUNCTION = 36.0000000000

VARIABLE TAG	NAME	BASIS/ BOUNDS	INTEG/ CONTIN	ACTIVITY LEVEL	OPPORTUNITY COST
2	YAB4	IF	I	0.0000000	0.0000000
3	YAC2	IF	I	1.0000000	0.0000000
5	YAC4	IF	I	0.0000000	0.0000000
8	YAD3	IF	I	0.0000000	0.0000000
13	YBC4	IF	I	0.0000000	0.0000000
19	YCB2	IF	I	0.0000000	0.0000000
22	YCD3	IF	I	0.0000000	0.0000000
24	YDA1	IF	I	0.0000000	0.0000000
25	YDA2	IF	I	0.0000000	0.0000000
28	YDB2	IF	I	0.0000000	0.0000000
30	YDC2	IF	I	0.0000000	0.0000000
32	YDC4	IF	I	0.0000000	0.0000000
31	YDC3	IF	I	1.0000000	0.0000000
17	YCA3	IF	I	0.0000000	0.0000000
26	YDA3	IF	I	1.0000000	0.0000000
29	YDB4	IF	I	0.0000000	106.0000000
14	YBD2	IF	I	1.0000000	0.0000000
15	YBD4	IF	I	1.0000000	0.0000000
12	YBC2	IF	I	1.0000000	0.0000000
1	YAB2	IF	I	0.0000000	0.0000000
7	YAD2	IF	I	1.0000000	0.0000000

4	YAC3	IF	I	1.0000000	0.0000000
6	YAD1	IF	I	1.0000000	0.0000000
9	YAD4	IF	I	0.0000000	109.0000000
10	YBA2	IF	I	1.0000000	0.0000000
11	YBA4	IF	I	1.0000000	0.0000000
16	YCA2	IF	I	0.0000000	0.0000000
18	YCA4	IF	I	1.0000000	0.0000000
20	YCB4	IF	I	1.0000000	0.0000000
21	YCD2	IF	I	1.0000000	0.0000000
23	YCD4	IF	I	1.0000000	0.0000000
27	YDA4	IF	I	1.0000000	0.0000000
49	FMAX	B	C	36.0000000	—
-2	—SLACK	B	C	21.0000000	—
-65	—SLACK	B	C	73.0000000	—
36	TA4	B	C	24.0000000	—
-5	—SLACK	B	C	3.0000000	—
-6	—SLACK	B	C	88.0000000	—
-7	—SLACK	B	C	9.0000000	—
-8	—SLACK	B	C	85.0000000	—
-9	—SLACK	B	C	94.0000000	—
-73	—SLACK	B	C	15.0000000	—
-11	—SLACK	B	C	91.0000000	—
-12	—SLACK	B	C	3.0000000	—
-13	—SLACK	B	C	21.0000000	—
-14	—SLACK	B	C	76.0000000	—
-15	—SLACK	B	C	76.0000000	—
-16	—SLACK	B	C	15.0000000	—
-17	—SLACK	B	C	82.0000000	—
-18	—SLACK	B	C	12.0000000	—
-19	—SLACK	B	C	6.0000000	—
-20	—SLACK	B	C	82.0000000	—
-22	—SLACK	B	C	91.0000000	—
-23	—SLACK	B	C	88.0000000	—
-24	—SLACK	B	C	3.0000000	—
-25	—SLACK	B	C	85.0000000	—
-26	—SLACK	B	C	9.0000000	—
-69	—SLACK	B	C	3.0000000	—
-28	—SLACK	B	C	9.0000000	—
-29	—SLACK	B	C	6.0000000	—
-30	—SLACK	B	C	91.0000000	—
-31	—SLACK	B	C	70.0000000	—
-32	—SLACK	B	C	21.0000000	—
-33	—SLACK	B	C	94.0000000	—
-34	—SLACK	B	C	0.0000000	—
-36	—SLACK	B	C	94.0000000	—
-37	—SLACK	B	C	3.0000000	—
-38	—SLACK	B	C	91.0000000	—
-39	—SLACK	B	C	76.0000000	—
-40	—SLACK	B	C	21.0000000	—
-41	—SLACK	B	C	9.0000000	—
-42	—SLACK	B	C	82.0000000	—
-43	—SLACK	B	C	91.0000000	—
-44	—SLACK	B	C	6.0000000	—

-45	--SLACK	B	C	88.0000000	—
-46	--SLACK	B	C	6.0000000	—
-47	--SLACK	B	C	15.0000000	—
-48	--SLACK	B	C	73.0000000	—
-49	--SLACK	B	C	85.0000000	—
-50	--SLACK	B	C	12.0000000	—
-51	--SLACK	B	C	15.0000000	—
-52	--SLACK	B	C	76.0000000	—
-53	--SLACK	B	C	12.0000000	—
-54	--SLACK	B	C	82.0000000	—
-55	--SLACK	B	C	82.0000000	—
-56	--SLACK	B	C	6.0000000	—
-57	--SLACK	B	C	91.0000000	—
-58	--SLACK	B	C	0.0000000	—
-59	--SLACK	B	C	21.0000000	—
-60	--SLACK	B	C	70.0000000	—
-75	--SLACK	B	C	3.0000000	—
-62	--SLACK	B	C	94.0000000	—
-63	--SLACK	B	C	6.0000000	—
-64	--SLACK	B	C	88.0000000	—
-74	--SLACK	B	C	3.0000000	—
-66	--SLACK	B	C	15.0000000	—
-67	--SLACK	B	C	12.0000000	—
-68	--SLACK	B	C	85.0000000	—
34	TA2	B	C	12.0000000	—
35	TA3	B	C	18.0000000	—
-27	--SLACK	B	C	82.0000000	—
40	TB4	B	C	9.0000000	—
42	TC2	B	C	18.0000000	—
43	TC3	B	C	27.0000000	—
48	TD4	B	C	15.0000000	—
45	TD1	B	C	24.0000000	—
46	TD2	B	C	30.0000000	—
39	TB3	NB	C	—	0.0000000
-1	--SLACK	NB	C	—	0.0000000
-3	--SLACK	NB	C	—	0.0000000
-72	--SLACK	NB	C	—	1.0000000
47	TD3	NB	C	—	0.0000000
37	TB1	NB	C	—	0.0000000
-61	--SLACK	NB	C	—	1.0000000
-77	--SLACK	NB	C	—	1.0000000
-21	--SLACK	NB	C	—	0.0000000
-70	--SLACK	NB	C	—	0.0000000
41	TC1	NB	C	—	0.0000000
-71	--SLACK	NB	C	—	0.0000000
-76	--SLACK	NB	C	—	1.0000000
33	TA1	NB	C	—	0.0000000
38	TB2	NB	C	—	1.0000000
-35	--SLACK	NB	C	—	0.0000000
-4	--SLACK	NB	C	—	1.0000000
-10	--SLACK	NB	C	—	0.0000000
44	TC4	NB	C	—	0.0000000

Appendix C

Results of the Proposed Expert System Approach for the Example in Chapter 3, Section 3.2.4