ON-LINE SCHEDULING AND CONTROL OF RANDOM FLEXIBLE MANUFACTURING SYSTEMS WITHIN AN OBJECT ORIENTED FRAMEWORK

By

CHUDA BAHADUR BASNET

Bachelor of Engineering
University of Poona
Poona, India
1973

Master of Science in Industrial and
Management Engineering
Montana State University
Bozeman, Montana
1987

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
DOCTOR OF PHILOSOPHY
July, 1991

ON-LINE SCHEDULING AND CONTROL OF RANDOM FLEXIBLE MANUFACTURING SYSTEMS WITHIN AN OBJECT ORIENTED FRAMEWORK

Thesis Approved:

Joe H. Mize
Thesis Adviser

M. P. Jewell

John W Nazemetz

C. M. Bacon

K E Case

Norman N. Durham
Dean of the Graduate College

ACKNOWLEDGEMENTS

I wish to express my sincere appreciation to the many individuals who helped me in the course of this work. I am particularly indebted to Dr. Joe H. Mize, my major adviser. He has provided superb guidance, but what I appreciate even more is his unfailing moral encouragement and support. I am also grateful to the other committee members Dr. Charles M. Bacon, Dr. Kenneth E. Case, Dr. John W. Nazemetz and Dr. M. Palmer Terrell for their advisement.

Special thanks go to Dr. Wayne C. Turner for his support of my graduate endeavors. I also appreciate the support and the intellectual environment provided by members (past and present) of the Center for Computer Integrated Manufacturing at OSU.

Finally, I wish to thank the members of my family for their confidence, understanding and sacrifices.

TABLE OF CONTENTS

Chapter Page

(*) Available in a separate volume in the School of Industrial Engineering and Management Library at Oklahoma State University

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

The goal of the research described herein was the development of a comprehensive framework in which automated manufacturing systems may be operated. This framework facilitates the decisions that need to be taken at the shop floor as the manufacturing processes are carried out.

Automation is increasingly being adopted in the manufacturing sector on account of the advantages of rapid turnaround, high quality, low inventory costs, and low labor costs. Flexible manufacturing system (FMS) is one of the options for automation in the discrete manufacturing industry. A FMS consists of a number of computer numerically controlled (CNC) workstations which can process a variety of parts. These parts are mounted on pallets which are transported by automatic transportation devices. The CNC machines have tool magazines on which some tools may be stored for automatic changeover as needed. Some systems even permit automatic conveyance of the tools to the machine magazines. The whole system is directed by a supervisory computer.

The research problems raised by the industrial espousal of FMS could be broadly classified into two problem areas: design problems and operation problems. At the design stage, one is interested in specifying the system so that the desired performance goals are achieved. Discrete event simulation has been the traditional mainstay of this endeavor. Recently, 'rough-cut' approaches based on queueing theory have been developed to narrow down the choices rapidly. The slower simulation analysis can then be applied to the 'short list'.

The operation problems are aimed at making decisions related to the planning, scheduling, and control of a given FMS. These problems differ on the basis of the planning horizon being considered. The long term planning decisions can be made with the traditional Master Production Schedule (MPS) and Material Requirements Planning (MRP) concepts. The problem of short term scheduling and control of FMS, however, has still not been answered adequately. The very flexibility of the FMS makes the choices of a scheduler too numerous to handle easily. The large number of system components and the corresponding constraints make the problem still more difficult.

Various methodologies have been suggested in the research literature to help the decision making process as faced by the operator of a FMS. These approaches are based on Operations Research, Artificial Intelligence, and other problem solving techniques. The approaches, however, either make too many restrictive assumptions, or take an inordinate amount of time, or are overly simplistic to be of practical use to the shop floor operator.

An ideal scheduling tool should take into account all the constraints of the FMS. It should permit convenient interaction between the supervisory computer and the human operator. The scheduler should help in making the decisions as the actual system conditions occur: machine failure, unavailability of material, change of priorities, quality problems, etc.

FMS scheduling is an active area of research. Various approaches are continually being offered. But the stringent requirements of the problem have made it difficult to bring about a complete, ideal solution. However, the high investment required for a FMS and the potential of FMS as a strategic competitive tool make it worth while to pursue a solution of the problem.

Like some other researchers, the position taken in this research is that discrete event simulation can be a practical tool to help in this decision making process. In as much as the developed model is a valid model of the real system, the schedules generated

by a simulator are bound to be feasible. Further, evaluation of a limited number of alternatives is possible within limited time bounds by using multiple passes of the simulation.

This approach is feasible on account of the increasingly high performance of the desktop computers. These computers are cheap enough to be solely dedicated to the scheduling task, and fast enough to provide adequate decision support. This trend is progressively favorable. Further, these computers provide excellent graphics hardware that can be utilized to develop user friendly interfaces for the operator of the FMS.

This simulation environment and the decision making framework is implemented in an Object Oriented Programming (OOP) language: Smalltalk-80. The chief merit of OOP from the viewpoint of the task in hand is that OOP makes it convenient to decompose complex systems. Each of the components of the system can be defined individually and its behavior can be described in isolation.

The goal of this research was to develop a conceptual framework in which the aforementioned tasks can be carried out. An event based approach was followed in which the framework reacts to the events as they occur on the shop floor. In the proposed framework, knowledge based simulation is used as a decision making tool. The system is designed to solve problems on-line, interactively. The framework has been implemented for 'proof of concept' purpose. Finally, the prototype has been evaluated using measures of effectiveness developed as part of this research effort.

CHAPTER II

STATEMENT OF THE PROBLEM

Introduction

Industrial Engineering researchers have devoted a lot of attention to scheduling problems. Over the years, much effort has been expended on finding a mathematical optimization formulation (and solution) of the problems. Owing to the combinatorial explosion, an optimal solution of practically sized problems is difficult to find.

Recent attention given to Flexible Manufacturing Systems has generated new interest in scheduling problems. The considerable investment required for FMS makes it essential that these systems be operated effectively. However, the variety and flexibility of these systems pose difficult problems for the operational system designer. FMS's are widely different. Almost every FMS needs its own specific scheduling system tailored for itself. The flexibility of FMS opens up many choices that need to be resolved.

Dupont-Gatelmand (1982) reported on a survey of FMS installations and showed the wide variety in the extant FMS's. It appears that automation of the machines and diversity of parts are the two criteria that are used commonly to justify the use of the designation - 'flexible'. She was able to classify the FMS's into three broad categories:

1. Flexible modules and units: These could be a single machining center with multiple head changer and automatic parts input/output system. They may have an automatic tool changer and pallet storage magazine. Some of these centers may be combined into machining cells with robots. A separate computer may or may not be used to control these systems. The flexibility is in the variety of parts machined.

2. Flexible conveyor lines: These systems are similar to automatic transfer lines. But they are multi-functional, and the parts do not necessarily have to go to all the machines. The part transfer may be done by roller conveyor, or shuttle conveyor, or wire guided carts.

3. Unaligned flexible systems: These systems are analogous to job shops. They are invariably controlled by computers. The parts transfer is done usually by wire-guided carts. The carts have on-board microcomputers.

The variety in FMS installations shows the difficulty in designing scheduling and control systems. Since this control is done as the process evolves on the shop floor, the control system has to be intimately tied to the actual system under consideration.

Rachamadugu and Stecke (1989) suggest another dichotomy for the classification of FMS's which is useful from the scheduling point of view: flexible flow systems (FFS) and general flexible machining systems (GFMS). In the flow systems, the parts follow one sequence of machines although different operations may be performed on them. Two types of these are the flexible assembly systems and flexible transfer lines.

The GFMS is more like a job shop - the processing is nonserial. Two modes of operation can be identified for these: dedicated and nondedicated. In the dedicated mode of operation, the FMS processes a fixed set of part types usually in a fixed ratio. The FMS is tooled for these particular parts, and often even the routing is pre-determined once the set of parts and their production ratio is known. Since the parts are made to stock, at a specific ratio, a stable schedule is usually sought for this type of FMS.

The other GFMS mode of operation is nondedicated. In this mode, the number of parts processed simultaneously by the FMS is much higher than that of the dedicated FMS. The set of parts is not fixed, nor is the production ratio of the parts. The flow of parts is controlled by customer order. Thus the production control is more JIT like, and the flow of materials in the FMS is like that of a traditional jobshop. The order size of a part type may vary as the external or the internal requirements change over time.

In a survey of 95 FMS's in the U.S.A. and Japan, Jaikumar (1986) found that most FMS applications in the U.S.A. were in the dedicated mode: producing only a few parts simultaneously. The average number of parts produced was 10; this number was 93 for Japan. Jaikumar asserts that the U.S. companies used the FMS the wrong (inflexible) way - for high volume production of a few parts, not for high variety production of many parts at low cost per unit. This is also borne out by the annual volume per part - 1727 for U.S. and 258 for Japan.

Thus, to really gain the competitive advantage of FMS, it is necessary to use it in the random mode: low volume, high variety of parts. To enable this, decision support methodologies need to be developed. The research described herein makes a contribution to this goal.

Most FMS scheduling research has also been targeted at the dedicated mode of operation of FMS. Little has been done to schedule the nondedicated (also called random) FMS. The scheduling problem for this type of FMS is also the most difficult - since it resembles the jobshop scheduling problem closely. Indeed, it is even more complex than jobshop scheduling on account of more alternatives and constraints.

Most of the FMS scheduling approaches fail to consider the effect of part transportation devices and the time required for this activity. Rachamadugu and Stecke (1989) point out that this could be a substantial issue since the processing time and the transportation times are comparable. Further, scheduling procedures typically neglect the limited buffer space available at the machining centers.

In conformity with the dedicated role of FMS, parts are assumed to be made to stock and the objective in the scheduling formulation is usually the maximization of production rate or machine utilization. But this objective is not suitable for random FMS, where the scheduling is order driven. In this application, due date should be the primary criterion for choosing among scheduling alternatives.

Statement of the Problem

To sum up, the extant models do not take into account the totality of the scheduling problem in FMS:

1. Material handling constraints.
2. Buffer capacity.
3. Alternate routing of parts.
4. Tool transport and tool changes.
5. Due date and priority.
6. Material availability.
7. Fixtures and pallets availability.
8. Tool availability and tool life.
9. Machine failures.
10. Dynamic production environment.
11. Scheduling and control decisions must be made on-line.

Although the current approaches do address some aspects of these, they ignore the others. The research problem addressed by this research may be stated as:

Existing scheduling techniques do not take into account all the relevant opportunities, constraints, and realities existing in a random flexible manufacturing system.

Outline of the Proposed Approach

The framework developed in this research presents a comprehensive methodology for scheduling and controlling random FMS. The salient points of this approach are:

Dynamic

In the traditional approach, everything is assumed to be started ab initio. Jobs are available at the start of the planning period. Once the schedule is decided upon, the

schedule is carried out in toto. Needless to say, this seldom happens in practice. Disruptions and unexpected changes continually occur on the shop floor. In the proposed approach, the supervisory computer continually updates the schedule in response to the events in the shop floor.

Reactive

The literature distinguishes between predictive and reactive scheduling. Predictive scheduling involves advance planning - prediction of events. Reactive scheduling reacts to the disruptions in the shop floor, and tries to bring the shop floor back on schedule.

In the proposed approach, like in discrete event simulation, all scheduling activities are event driven. A unified, reactive viewpoint is taken for all the events. The schedule is always updated in response to events in an effort to meet the goals of the scheduler. Arrival of new jobs is treated in the same fundamental way as the failure of a machine - as an event.

Comprehensive

The proposed approach considers major constraints that exist on the floor.

Tooling. Traditional approaches overlook the existing tooling of the machines. It is assumed that the machines will be completely re-tooled every scheduling period. In the event driven scheduling proposed here, the tools already on the tool magazines are taken into account as a planning factor. The tool changes are planned as necessitated by the events.

AGV. It is said that the transportation devices are not usually bottlenecks in the operation of FMS's. But they still need to bear on the scheduling of FMS's.

Buffer Space. Although it is not a major concern, the storage for WIP is limited in an FMS. Thus, FMS scheduling is constrained by the available buffer space.

Object-Oriented

The object-oriented paradigm has been employed because of its superior capability of modeling complex systems. It is used for the implementation of discrete event simulation, embedded expert systems, and for control algorithms.

CHAPTER III

BACKGROUND OF THE STUDY

Introduction

This chapter presents a review of the pertinent literature on the problem to be addressed by this research. Production scheduling has had a long history as an object of production management research. It is not the intent here to discuss all the results of this research effort. This review focuses on scheduling problems of Flexible Manufacturing Systems (FMS) with more emphasis placed on the specific problem areas and on the approaches closest to this research.

Early scheduling researchers spent a lot of effort on finding algorithmic solution of the particular problems of their interest. With the realization of the intractability of the general problem, the emphasis has been placed on the analysis of complexity of the problem. Algorithmic approaches are now being actively sought for only very small problems - one or two machines or other problems of particular structures.

The general job shop scheduling problem is known to be NP-hard (there is no known algorithm to solve it, that solves it in number of steps which is a polynomial of the size of the problem). The FMS scheduling problem has more alternatives and constraints than the job shop problem, and can be conjectured to be NP-hard too. Various versions of the FMS scheduling problem have been shown to be NP-hard [Hwan and Shogun, 1989]. On account of this intractability, researchers have suggested many heuristical or other non-optimal solutions to the problem.

FMS scheduling literature could be classified in many ways. For the purpose of this review, the following taxonomy, based on the basic approach, is used.

1. Mathematical programming approach
2. Heuristics oriented approach (dispatching rules)
3. Control theoretic approach
4. Simulation based approach
5. Artificial Intelligence (AI) based approach
6. Interactive approach

There is some cross fertilization among these approaches. For example, some AI based approaches use simulation to generate or evaluate schedules. Similarly, an interactive approach may use any of the other five methods. In the following discussion, the approaches are classified on the basis of their main emphasis.

Mathematical Programming Approach

In this approach, the researchers have cast the problem into an Operations Research model. Buzacott and Yao (1986) present a comprehensive review of the analytical models developed for the design and control of FMS up until 1984. They strongly advocate the analytical methods as giving better insight into the system performance than the simulation models. The analytic models let the modeler identify the key parameters and their influence on the performance of FMS.

Stecke (1983) appears to have done comprehensive work on the operational aspects of dedicated FMS. She points out the opportunities provided by the FMS as the reasons why managing production for an FMS is more difficult - the versatility of machines, simultaneous processing of multiple part types, and alternate routing of the parts.

To manage the complexity of the problem, Stecke and many other authors who have followed her divided the FMS operation problem into two subproblems: preproduction setup and production operation. In this view, a FMS is prepared beforehand for the given part mix: loading the tools, allocating the operation to the machines, allocating the

pallets and fixtures to the different part types. After this preparatory planning phase, the remaining problems are called operational problems and solved later. It should be noted, however, that there is no clear boundary between the planning and the scheduling problems, and one approach may place most of the burden on the planning stage, while another approach may do the opposite. Stecke (1983) places stress on pre-production setup of the FMS. This is to be carried out frequently, as the part mix changes. To carry out a complete setup, a FMS manager would solve 5 problems:

1) Part type selection problem. This problem determines the part types to be produced in the FMS out of the total production requirement of the company.
2) Machine grouping problem. Stecke would partition the machines in the FMS so that machines in a group can all perform the same operations.
3) Production ratio problem. This problem is related to problem 1 -determine the ratio of the parts selected to be manufactured in the FMS.
4) Resource allocation problem. This problem determines the allocation of pallets and fixtures to the part types.
5) Loading problem. The solution to the problem will simultaneously allocate operation of the part types and the corresponding tools to the machine groups.

The five problems could be solved successively, or in an iterative fashion. Stecke (1983) then goes on to describe models for the grouping and loading problems. For this problem, the major constraint is the capacity of tool magazines of each machine tool. This is complicated by the different number of slots taken up by the tools, and the commonality of the tools requirements of different part types.

Assume a total of m machines are to be grouped into a total of M machine groups, and let

x_{il} = 1 if operation i is assigned to machine group l, 0 otherwise

b = total number of operations to be assigned

and q_i = maximum number of machine groups operation i can be assigned to

Then,

$$1 \leq \sum_{l=1}^{M} x_{il} \leq q_i, \quad i = 1, ..., b$$

The tool magazine capacity constraints may be simply stated as

$$\sum_{i=1}^{b} d_i x_{il} \leq t_l, \qquad l = 1, \ldots, M$$

where d_i is the number of tool slots taken up by operation i, t_l is the tool magazine capacity of the machine tools in group l. The above formulation is overly simplistic because it ignores the common tools for the different operations assigned to a machine group. This becomes complicated on account of the set exclusion and inclusion operation that needs to be done to take care of commonality. If $w_{i_1 i_2 i_3 \ldots}$ is the count of slots occupied by the tools contained in the intersection of the sets of tools required by the operations $i_1, i_2, i_3, ...$, then the tool magazine capacity constraint may be written as

$$\sum_{i=1}^{b} d_i x_{il} - \sum_{i_1=1}^{b-1} \sum_{i_2=i_1+1}^{b} w_{i_1 i_2} x_{i_1 l} x_{i_2 l} + \sum_{i_1=1}^{b-2} \sum_{i_2=i_1+1}^{b-1} \sum_{i_3=i_2+1}^{b} w_{i_1 i_2 i_3} x_{i_1 l} x_{i_2 l} x_{i_3 l} - \ldots \ldots \leq t_l$$

The minimum number of machines, M, required to cover all operations is calculated. This is done by initially considering each individual machine (there are m in all) as a group and posing the problem

$$\text{Maximize} \sum_{j=1}^{m} \gamma_j sl_j$$

subject to

$$sl_j = t_j - \sum_{i=1}^{b} d_i x_{ij} + \sum_{i_1=1}^{b-1} \sum_{i_2=i_1+1}^{b} w_{i_1 i_2} x_{i_1 j} x_{i_2 j}$$
$$- \sum_{i_1=1}^{b-2} \sum_{i_2=i_1+1}^{b-1} \sum_{i_3=i_2+1}^{b} w_{i_1 i_2 i_3} x_{i_1 j} x_{i_2 j} x_{i_3 j} + \ldots \ldots$$

The parameter γ_j is a parameter to weight the slack in tool magazines, sl_j. The above formulation is subject to the other constraints defined earlier. The object of this formulation is to pack as many tools as possible in few machine tools, at the same time making enough tool allocations to cover all the part types. The above problem gives the number of groups M needed. If there are more machines than the number of groups, the additional machines are tooled identical to some of the ones that are grouped. This way, the machines are pooled to allow maximum flexibility.

In Stecke's methodology, the operations and corresponding tools are then assigned (loaded) to the machine groups. She suggests 6 different objectives to optimize during the loading phase.

1. Balance the assigned machine processing times. The relative workload r_j assigned to machine j may be defined as

$$r_j = \sum_{i=1}^{b} a_i p_{ij} x_{ij}, \qquad j = 1, \ldots, m$$

where a_i is the proportion of part type i relative to the other part types, (that is, the part types are in the ratio $a_1 : a_2 : \ldots : a_b$), p_{ij} is the processing times of part type i on machine j. (There is a change in notation here. The previous notation for operations

is now used for part types). Then, to balance the assigned machine processing times, she suggests many objectives, one of which is given below:

$$\text{Minimize} \sum_{j=1}^{m-1} \sum_{h=j+1}^{m} |r_j - r_h|^{\gamma}, \qquad \gamma > 0$$

2. Minimize the number of movements from machine to machine. This objective may be stated as

$$\text{Minimize} \sum_{i=1}^{b-1} \sum_{j=1}^{m} (x_{ij} - x_{i+1,\,j})^2$$

3. Balance the workload per machine for a system of groups of pooled machines of equal sizes. Here, instead of the machines, it is sought to balance the workload across groups. If s_l is the number of machines assigned to group l, and r_l is the load on group l, then this objective may be stated as

$$\text{Minimize} \sum_{i=1}^{M-1} \sum_{k=l+1}^{M} \left[\frac{r_l}{s_l} - \frac{r_k}{s_k}\right]^2$$

4. Unbalance the workload per machine for a system of groups of pooled machines of unequal sizes.This objective stems from earlier results of Stecke and Solberg (1982) that recommends unbalancing the workload for each machine when the pooled group sizes are unequal in order to obtain maximum production rate. This objective may be stated as follows:

$$\text{Minimize} \sum_{l=1}^{M} (r_l - X_l^*)^2$$

X_l^* in the above expression is the theoretical optimal workload that should be assigned to machine group l to maximize expected production [Stecke and Solberg, 1982].

5. Fill the tool magazines as densely as possible. The rationale of this objective is to allow the greatest number of alternate routings. This objective is similar to the earlier formulation to determine the minimum number of machine groups M, except it is sought to place as many tools as possible.

$$\text{Minimize} \sum_{j=1}^{m} sl_j$$

6. Maximize the sum of operation priorities. Here, again, most operations are sought to be assigned to multiple machines to permit alternate routings.

$$\text{Maximize} \sum_{i=1}^{b} \sum_{j=1}^{m} w_i x_{ij}$$

The weight w_i on the operations is used to favor more critical operations such as bottleneck operations.

The formulations of Stecke (1983) lead to large nonlinear mixed integer problems. She suggests various linearization schemes. Stecke's planning problems place much of the scheduling problem in the setup stage. Once the setup is done as per the five specific sub-problems, most of the resource allocation is already complete. The setup is carried out for a particular part mix.

It is not clear when one of the six loading objectives is to be favored over the others. In some cases, where the machine tools are separated over a long distance, the choice is obvious. In other cases the answer is hard to discern. The grouping problem does not consider the production ratio a_i. Thus, it could give an answer which is not

desirable from the view point of maintaining the production ratio. Another problem with the formulation is the large number of variables and constraints that result from the linearization of the problems. That makes the approach computationally expensive.

Stecke's approach is explained above at length because other mathematical modeling approaches build upon this foundational work. Lashkari et al. (1987) developed a formulation of the loading problem. Their formulation considered refixturing and limited tool availability. Refixturing of the parts may be required when a different machine is used, or when a different machining operation is performed in the same machine tool. Besides this problem, they place an upper bound on the number of tools that may be assigned. Their approach assumes that between all machine transfers, a part necessarily passes through a central storage. They consider two objectives:

1. Minimization of total transportation requirements of the parts. This could be an important objective where the distance between machining centers is large relative to the operation time. This is Stecke's second objective.

Define $X_{i,j,k} = 1$ if operation k of part type i is assigned to machine j

$= 0$ otherwise

O_i = number of different operations to be performed on part type i

M = number of machine tools in the FMS

Then the distance traveled by part i for the first operation

$$= \sum_{j=1}^{M} (L_j \cdot X_{i,j,1}),$$ where L_j is the loading distance to machine j

Similarly the distance traveled by part i for the last operation

$$= \sum_{j=1}^{M} (U_j \cdot X_{i,j,(O_i)}),$$ where U_j is the unloading distance from machine j

The total distance traveled by part i for inter-machine transfers

$$= \sum_{k=1}^{O_i-1} \sum_{j=1}^{M} (X_{i,j,k} - X_{i,j,k+1})^2 (U_j \cdot X_{i,j,k} + L_j \cdot X_{i,j,k+1})$$

And, finally, if there is a refixturing, it needs a transfer to the central storage and back even if the operation is carried out in the same machine tool. This distance

$$= \sum_{k=1}^{O_i-1} \sum_{j=1}^{M} (X_{i,j,k} \cdot X_{i,j,k+1}) \cdot F(i,k+1,j,j) \cdot (U_j + L_j)$$

where $F(i,k,p,q) = 1$ if refixturing is needed by part type i for operation k, and kth operation is done on machine q and $(k\text{-}1)$th operation is done on machine p, 0 otherwise. The sum of these 4 terms is the total distance D_i for a part type i. To minimize the transportation load, the objective is

$$\text{Minimize } Z = \sum_{i=1}^{R} D_i, \text{ where R is the total number of part types in the FMS}$$

2. Minimization of refixturing requirements. Number of refixturings needed by part type i for operation k

$$= Q(i,k) = \sum_{p=1}^{M} X_{i,p,(k-1)} \left[\sum_{q=1}^{M} (X_{i,q,k} \cdot F(i,k,p,q)) \right]$$

To minimize the total number of refixturings for all the part types produced in the FMS, the objective is to minimize Z, where Z is given by

$$Z = \sum_{i=1}^{R} \sum_{k=2}^{O_i} Q(i,k)$$

They restrict the allocation of part types to single machines. This results in the constraint

$$\sum_{j=1}^{M} X_{i,j,k} = 1 \text{ for each pair } i, k$$

When desirable, certain operations may always be grouped together on a machine tool.

$$\sum_{j=1}^{M} \prod_{G} X_{i,j,k} = 1 \text{ where G is the set of operations to be grouped.}$$

Let $Y_{j,H} = 1$ if machine tool j is equipped with tool of type H.

$= 0$ otherwise.

If the number of available tools of type H is h, then

$$\sum_{j=1}^{M} Y_{j,H} \leq h \text{ for each tool type H}$$

The constraint arising out of the capacity of the tool magazines is expressed in the same way as Stecke (1983), already explained above.

The above objective functions and the constraints have products of 0-1 integer variables. Lashkari et al. (1987) linearize the formulation to solve the problem using linear integer programming code. Their computational experience shows that even for small problems (2 to 5 part types, 2 to 5 operations per part), the problem size becomes considerably high. In order to reduce the search, they suggested dividing the problem into two sub-problems, the result of which could be used as an upper bound for the original problem.

Unlike Stecke, Lashkari et al. will permit only one allocation of a machine to an operation. This would curtail some flexibility at the operation control level. Their modeling is suitable only when the parts must always traverse to and from a central storage for every inter-machine transfer. Further, the objective function lacks the relative weighting for the different part types.

Wilson (1989) used simpler and more straight forward formulation of the constraints to solve the same problem as discussed by Lashkari et. al. (1987). He was able to do this for the reason that not all $X_{i,j,k}$ are permissible to begin with. Thus all tool allocations need not be constrained. The non-linear terms in the first objective function could also be eliminated by restructuring the expression. He demonstrated substantial savings in computational effort using his modeling of the constraints and the objective function.

Shanker and Rajamarthandan (1989) present a similar model with the objective of part movement minimization. In contrast to Lashkari et al. (1987), they do not require the parts to go to a central storage after every operation. Also, they are not interested in the distance traveled: only the number of movements is of concern. This changes their objective function to (in terms of the notation of Lashkari et al.)

$$\text{Minimize} \sum_{i=1}^{R} \sum_{k=1}^{O_i-1} \sum_{j=1}^{M} (X_{i,j,k} - X_{i,j+1,k})^2$$

They also consider the same constraints. Like Wilson (1989), they exploit the particular structure of the problem to obtain linearization of the problem. They also reported that high computational effort was required.

Han et al. (1989) address the setup and scheduling problem in a special type of FMS: where all the machines are of the same type, and tools are 'borrowed' between machines and from the tool crib as needed. In their model, the number of tools is lim-

ited. The purpose of their model is to assign tools and jobs to machines so that the 'borrowing' of tools is minimized while maintaining a 'reasonable' workload balance.

Define:

a_{jt} = 1 if part type j requires tool type t, 0 otherwise

x_{it} = 1 if tool type t is assigned to the magazine of machine tool i, 0 otherwise

y_{ij} = 1 if part j is assigned to the machine tool i, 0 otherwise

Then the number of tool types required by a job j is given by

$$r_j = \sum_{t=1}^{l} a_{jt},$$

where l is the number of tool types.

If the job j is assigned to the machine tool i, the number of tools available there for job j,

$$e_{ij} = \sum_{t=1}^{l} a_{jt}\, x_{it}$$

To minimize the number of tools to be borrowed the objective may be written:

$$\text{Minimize} \sum_{i=1}^{m} \sum_{j=1}^{n} (r_j - e_{ij})\, y_{ij}\,;$$ where m is the number of machines and n is the number of jobs.

There is a limited number (c_t) of each type t of tools available.

$$\sum_{i=1}^{m} x_{it} \le c_t \quad \text{for } t = 1, 2, \ldots, l$$

The tool magazine of machine tool i has limited capacity s_i :

$$\sum_{t=1}^{l} x_{it} \le s_i \quad \text{for } i = 1, 2, \ldots, m$$

Each part type is assigned to one machine tool

$$\sum_{i=1}^{m} y_{ij} = 1 \quad \text{for } j = 1, 2, \ldots, n$$

If total processing time of job j is p_j, then the load on each machine i is

$$\sum_{j=1}^{n} p_j y_{ij}$$

For a perfect balance of the workload, the load on any machine should be

$$b = \frac{\sum_{j=1}^{n} p_j}{m}$$

The objective of workload balance is framed as a constraint. The workload imbalance is defined by α, and the constraint may be written,

$$b(1 - \alpha) \leq \sum_{j=1}^{n} p_j y_{ij} \leq b(1 + \alpha), \quad i = 1, 2, \ldots, m$$

The problem as posed above is a nonlinear integer programming problem, and is computationally expensive. To solve the problem efficiently, the authors propose to decompose the problem. The two sub-problems each have the same objective as shown above. But the constraints are divided. The first problem finds an optimum tool allocation x_{it}, given the job allocation y_{ij}. This will need the first two constraints only. The second problem finds an optimal job allocation y_{ij}, given the tool allocation x_{it}. Phrased in this way, both problems become linear. The first problem is a capacitated transportation problem, and the second is a generalized assignment problem. It is suggested to solve the two problems iteratively.

They also propose a greedy heuristic to solve the same problem. Their heuristic does not consider the workload. The jobs are assigned to the machines with the largest processing times assigned first. Then the number of times each tool type is required at a

machine tool is determined. A machine tool with the highest requirement is allotted the tool. This process is repeated until there are no more of both machine tools and tool types left to assign.

Han et al. (1989) also carried out a simulation study of throughput performance under two different setup procedures (decomposition, heuristic), two queue formation methods (common queue of incoming parts, individual queue of machine tools), two tool return policies (return borrowed tools when finished, return only when needed) , and four dispatching rules (longest processing time, least number of tool movements, shortest processing time, random). They recommended their heuristic tool loading method together with a policy of not returning a tool until needed. The differences between the dispatching rules were not found significant.

The FMS investigated by Han et al., is special. All machine tools are assumed identical. Consequently, the jobs remain at one machine, and the tools are moved to the machines as needed.

Random FMS

The dedicated FMS problem assumes a fixed part mix. As seen above in Stecke (1983), the part mix is selected from the total production requirement of the company. When the machines in the FMS are grouped, and loaded with the parts, the operation of the parts is allocated to the machines. Then until the production allocation is changed again, the FMS is operated in the same way as a job shop since the allocation of operation and tooling of the machines is taken care of. If the parts visiting the machine are not selected in advance, the operations need to be allocated as the parts arrive and the machines need to be tooled correspondingly. This type of FMS is called "random FMS".

Hutchison et al (1989) provide a mathematical formulation of the problem. Their formulation is a static one in which N jobs are to be scheduled on M machines. The objective is to minimize the makespan:

Minimize T_{max}

subject to

$$X_{i(g+1)k(m)} - X_{igk(m)} + H\,(1 - V_{i(g+1)k(m)}) \geq P_{i(g+1)k(m)}$$

$$\text{for } i = 1, \ldots, N;\ g = 1, \ldots, Q_i - 1;\ k = 1, \ldots, Z_{i(g+1)}$$

where $X_{igk(m)}$ is the completion time of the kth option of operation g on machine m of job i; H is an arbitrarily large number; $V_{igk(m)} = 1$ if the kth option of operation g on machine m of job i is used, 0 otherwise; $P_{igk(m)}$ is the processing time of kth option of operation g on machine m of job i; and T_{max} is the largest completion time for the last operation of all the jobs; Z_{ig} is the number of alternative machine options for operation g of job i.

This constraint assures the precedence relations of the operations.

$$X_{igk(m)} - X_{jhq(m)} + H\,(1 - Y_{igk(m)jhq(m)}) \geq P_{igk(m)}\ V_{igk(m)}$$

$$X_{jhq(m)} - X_{igk(m)} + H\,(Y_{igk(m)jhq(m)}) \geq P_{jhq(m)}\ V_{jhq(m)}$$

$$\text{for } i = 1, \ldots, N;\ g = 1, \ldots, Q_i;\ m = 1, \ldots, M$$

where $Y_{igk(m)jhq(m)} = 1$ if the kth option of operation g on machine m of job i precedes qth option of operation h on machine m of job j, 0 otherwise; Q_i is the number of operations in job i.

These constraints ensure that no two operations are processed on a machine simultaneously.

$$X_{iGk(m)} \leq T_{max} \quad \text{for } i = 1, \ldots, N;\ k = 1, \ldots, Z_{ig}$$

This constraint defines T_{max} as the largest of all operation completion times.

$$X_{i1k(m)} \geq P_{i1k(m)}\ V_{i1k(m)} \quad \text{for } i = 1, \ldots, N;\ k = 1, \ldots, Z_{i1}$$

This ensures that the completion time of the first operation must be equal to or greater than its processing time.

$$X_{igk(m)} \leq H V_{igk(m)} \quad \text{for } i = 1, ..., N;\ g = 1, ..., Q_i;\ k = 1, ..., Z_{il}$$

This constraint sets the unused operation completion time to zero.

$$\sum_{k=1}^{Z_{ig}} V_{igk(m)} = 1 \quad \text{for } i = 1, ..., N;\ g = 1, ..., Q_i$$

This is needed to select only one machine option.

This is a mixed integer 0-1 programming formulation. They solve this problem by a branch and bound scheme. As can be seen, a single formulation solves the allocation of the operations to the machines and the timed sequence of the operations. However, their study assumes that material handling devices, pallets, buffers, and tool magazines do not constrain the system. Further, at most one alternative is allowed for any operation.

An alternative approach to the above problem is to decompose it into two sub-problems. The first problem is the allocation of the jobs to the machines in the routings. The second problem is the time bound sequencing of the jobs, the standard job shop problem.

Hutchison et al (1989) report on a comparison of the performance of the above two methodologies and another methodology which was based on dispatching rule (SPT). A novel feature of their simulation experiment is their use of a measure of flexibility: probability of an alternate machine option for any operation. This measure was set at nine levels in their experiment.

They concluded that the programming formulations produced substantial improvement in makespan over the dispatching rules. However, as compared to the decomposed problem, the unified formulation did not produce significant improvement in makespan to justify the additional computational effort required.

In the above approach, the tool magazines do not constrain the system. Hence the first subproblem of the decomposition can allocate all the jobs to their machines. However, when the tool magazine is considered restraining, it may not be possible to allocate

all the jobs for one tooling setup. Then this subproblem resolves to a selection problem. Out of the pool of waiting jobs, jobs are selected to be processed in the next planning period (part type selection problem) . The selected parts are then sequenced. The process is repeated period by period. In this approach, it is assumed that at the beginning of each planning period all the tools are reassigned and replaced in the tool magazine.

Shanker and Tzen (1985) propose a mathematical programming approach to solve the part selection problem for random FMS. Their approach is similar to [Stecke, 1983]. Stecke assumes the part ratio as given and the planning horizon as indefinite whereas Shanker and Tzen consider individual parts and a fixed planning horizon. They have a constraint on the tool magazine capacity which is very similar to Stecke's. They constrain the model to find a unique routing for each part type (in contrast to Stecke).

$$\sum_{G \varepsilon B(i,k)} x_{ikG} \leq 1, \quad i = 1, 2, \ldots m; k = 1, 2, \ldots y_i$$

Here, $B(i,k)$ is the set of machines on which operation k of job i can be performed. $x_{ikG} = 1$ if operation k of job i is performed on machine G, 0 otherwise. y_i is the number of operations for job i.

They do not allow splitting of the job between planning horizons.

$$\sum_{k=1}^{y_i} \sum_{j=1}^{n} x_{ikj} = x_i y_i, \; i = 1, 2, \ldots m$$

where $x_i = 1$ if job i is selected for production in this planning period, 0 otherwise.

The overload and underloads on the machines may be specified as a constraint.

$$\sum_{i=1}^{m} \sum_{k=1}^{y_i} p_{ikj} x_{ikj} + U_j - O_j = H, \quad j = 1,2, \ldots .. n; \; U_j, O_j \geq 0; \; U_j O_j = 0$$

where p_{ikj} is the processing time of operation k of job i on machine j; U_j and O_j are the underloads and overloads for machine j; and H is the length of the planning horizon. Two objectives are considered:

1. Balancing the workload. The objective is to minimize the sum of both the overload and the underload of all the machines. This will attempt to load the machines as close to capacity as possible.

$$\text{Minimize } Z_1 = \sum_{j=1}^{n} w_{oj} O_j + \sum_{j=1}^{n} w_{uj} U_j$$

where w_{oj} is the weight on overload O_j and w_{uj} is the weight on underload U_j for machine j.

2. Balancing the workload and minimizing the number of late jobs.

$$\text{Minimize } Z_2 = \sum_{j=1}^{n} w_{oj} O_j + \sum_{j=1}^{n} w_{uj} U_j - \sum_{j=1}^{m} \frac{w_{di} x_i}{\max (D, R_i - 2H)}$$

To minimize the late jobs, w_{di} weights the job i, R_i is the remaining time on job i. D is a parameter with small value that influences the selection of late jobs. Jobs due within two planning horizons will be given higher priority.

The resulting problems are, again, non-linear integer problems. They suggest linearization schemes. Even after linearization, the problems are computationally too expensive, and they further propose two heuristics corresponding to the two objectives. For balancing the workload, they propose essentially a greedy heuristic which attempts to allocate to the most lightly loaded machine the longest operation first. For the second

objective, the same heuristic is modified to include the overdue jobs with the highest priority. Their computational experience showed that the analytical formulations would be too formidable to be of practical use, and they suggested further research to obtain better heuristics.

In the above approaches for random FMS, the scheduling of the FMS is decomposed into two problems: part type selection, and sequencing of jobs. The sequencing is done using one of the dispatching rules. Of course, some (e.g. branch and bound) search could be used to solve the sequencing problem too. Hwan and Shogun (1989) present the part selection problem for a random FMS with machines of a single general purpose type capable of producing all part types. They include the due date and the quantity of parts needed to be produced in their formulation. By ignoring the tool overlapping (cf. Stecke, 1983), they considerably simplify the tool magazine constraint. Their objective is to maximize the number of part types selected over a planning horizon, a surrogate for maximizing the production rate. They take care of due dates by weighting on the selected part types. By assuming a single machine type, their problem essentially boils down to maximizing the utilization of the tool slots in the tool magazines. They report computational experience on two Lagrangian relaxation techniques they used to solve the problem. Their heuristics and Lagrangian methods obtained solutions close to optimal solutions found by the Branch and Bound method. The CPU times required by the three methods are successively order of magnitudes higher.

Jaikumar and Wassenhove (1989) propose a hierarchical planning and scheduling decomposition of FMS operation problems. In the first level, an aggregate production model is used. This is a linear programming model that chooses parts to be produced in a FMS during the next planning period. The remaining parts are assumed to be produced elsewhere at a cost difference. The objective is to maximize the cost difference while allowing for the inventory cost for work in process. The essential constraints are the demand for the parts and the machine capacity. The second level objective will organize

the production requirements into part families to minimize family setup cost and tool setup cost while the production requirement determined in level one acts as a constraint. Put simply, the objective of the second level is to minimize tool changeover. The production requirements and the tool and machine allocation are determined in levels one and two. All that remains in the third level is to determine a feasible schedule that will fulfill the above requirements. Detailed requirements such as buffer requirements, and material handling constraints, are taken care of at this level. Jaikumar and Wassenhove recommend simulation using some dispatching rule to carry out this level. If a feasible schedule cannot be obtained, the planning process is reiterated. They discuss the application of their framework in an existing FMS and point out that the primary problem is at the first level - selection of parts. Once this is decided upon, the other two problems can be solved by simple heuristics.

Planning of a FMS is a problem with multiple criteria. Lee and Jung (1989) formulate a part selection and allocation problem using goal programming. Their model considers the goals of 1) meeting production requirements, 2) balancing of machine utilization, and 3) minimization of throughput time of parts. Deviational variables representing the under- and over- achievement for each of the goals are used to measure the deviation from the goal. The model casts even the technological constraints into goal constraints. The goal programming model of Lee and Jung can provide the decision maker with a satisficing solution for given goals and their prioritization. But even with restrictive assumptions, the model is computationally expensive for practical use.

It has been pointed out in the course of the above discussion that the mathematical programming formulation of the problem tends to get computationally expensive as the problem size increases. The only way to use these models is to use the heuristics which are offered as approximations to the models. But then they do not optimize anymore. The models make simplifying assumptions which are not always valid in practice. The assumptions, of course, change with the models: some models assume automatic tool

transport, some others will neglect delays caused by AGV's, still others will assume that tool magazines, pallets and fixtures do not constrain the models in any way, and so on. The models also take a static view of the shop floor. It is assumed that all the planned activities will be carried out exactly, or the disruptions are infrequent enough that periodic solution of the problems will be practical.

Hueristics Oriented Approach

To counter the difficulties mentioned above, use of simple heuristics or dispatching rules has been suggested. Given that a resource is idle, the heuristics will quickly yield the operation it should carry out. Given that an operation is finished, the heuristics will say where the job needs to go. Extensive study of these dispatching rules have been carried out in the general job shop context [Conway, 1965; Conway, 1965b; Gere, 1966; Panwalker and Iskander, 1982]. In the same vein, numerous simulation studies of dispatching rules have been carried out in the FMS area. A comprehensive survey is presented in [Gupta et al., 1989]. Some significant results are discussed below.

Nof et al. (1979) carried out a study of different aspects of planning and scheduling of FMS. They explore the part mix problem, part ratio problem, and process selection problem. In the scheduling context, they report on three part sequencing situations:

1. Initial entry of parts into an empty system
2. General entry of parts into a loaded system
3. Allocation of parts to machines within the system (dispatching rules)

They examined three initial entry control rules, two general entry rules, and four dispatching rules. Their conclusion was that all these issues were interrelated: performance of a policy in one problem is affected by choices for other problems.

Stecke and Solberg (1981) investigated the performance of dispatching rules in a FMS context. They experimented with five loading policies in conjunction with sixteen dispatching rules in the simulated operation of an actual FMS. Under broad criteria, the

shortest processing time (SPT) rule has been found to perform well in a jobshop environment [Conway,1965 ; Conway, 1965b]. Stecke and Solberg, however, found that another heuristic - SPT/TOT, in which the shortest processing time for the operation is divided by the total processing time for the job - gave significantly higher production rate compared to all the other fifteen rules evaluated. Another surprising result of their simulation study was that extremely unbalanced loading of the machines caused by part movement minimization objective gave consistently better performance than balanced loading.

Buzacott and Shantikumar (1980) consider the control of FMS as a hierarchical problem: a) Pre-release phase, where the parts which are to be manufactured are decided, b) Input or release control, where the sequence and timing of the release of jobs to the system is decided, and c) Operational control level, where the movement of parts between the machines is decided. Their relatively simple models stress the importance of balancing the machine loads, and the advantage of diversity in job routing. Buzacott (1982) further stresses the point that operational sequence should not be determined at the pre-release level. His simulation results showed that best results are obtained when:1) For input control, the least total processing time is used as soon as space is available, and, 2) For operational control, the shortest operation times rule is used.

In the study of Shanker and Tzen (1985), the formulation of the part selection problem is mathematical; but its evaluation was carried out in conjunction with dispatching rules for scheduling the parts in the FMS. Further, on account of the computational difficulty in the mathematical formulation, they suggested heuristics to solve the part selection problems too. They used four dispatching rules: FIFO, SPT, LPT, MOPR (most operations remaining first). They felt that they had not done enough simulation runs to conclude either way about the dispatching rules. But they conjectured that MOPR would perform better (in terms of machine utilization) when the workload is balanced. On the average, SPT performed the best.

Moreno and Ding (1989) take up further work on heuristics (for part selection) as mentioned above, and present two heuristics which reportedly give better objective values than the heuristics in [Shanker and Tzen, 1985]. This, however, they are able to do by increasing the complexity of the heuristics. Their heuristic is 'goal oriented' - in each iteration, they evaluate the alternate routes of the selected job to see which route will contribute most to the improvement of the objective. Otherwise, their heuristic is the same as that of Shanker and Tzen.

Chang et al. (1989) report on a heuristics based beam search technique designed to solve the random FMS scheduling problem. Beam search is a breadth first search technique in which an evaluation function is used to retain only a certain number of nodes to sprout at every level. The number of retained nodes is called the beam width of the search. The quality of the solution, of course, depends on the evaluation function. The root of their search tree has no operation scheduled. They progressively go along the time line and schedule more and more operations until at the final leaf, all the operations are scheduled. At each node, to evaluate the schedule, they carry out a simulation using the SPT rule. This SPT rule identifies the critical path in the schedule. This is analogous to the Critical Path Method (CPM). For the first machine in the critical path, they evaluate all the possible alternate assignments. Only a certain number (beam width) of assignments is then fixed depending on the makespan obtained.

A contribution of Chang et al. is a measure of flexibility of the manufacturing system. This is called a flexibility index. It denotes the average number of workstations able to process an operation. Flexibility index is 1 for the conventional job shop. For various values of the flexibility indices, they compare their algorithm against several dispatching rules. As can be expected, their algorithm gives better results than the dispatching results at the cost of increased computational effort. It can also be seen that as the flexibility of the FMS increases, even a beam width of 1 gives very good results.

Chang et al. do not consider the tool magazine as restraining. They do not consider the pallets, fixtures, and transportation devices either. The only constraints considered are the capacity constraints, precedence constraints, and the routing (which may have alternate machines). This limits the usefulness of their approach.

Donath (1988) developed a heuristic based hierarchical methodology to schedule a FMS in near real-time. In his approach, at every point of decision, e.g. completion of a job, a program called 'SCHEDULE' is run. This makes decisions on the next assignment of assignable operations. His decomposition has two main subproblems. In the first, a cost of assigning an operation to a machine is calculated on the basis of process time, idle time, and the average time for that operation. Secondly, a generalized assignment problem is solved to assign the jobs to the machines. All the pending operations are assigned even if they were assigned already (but not carried out). The runtime of SCHEDULE is said to be near real time (about a minute). However, the tool magazine capacity of the machines is not considered in this methodology.

Slomp et al. (1988) consider three quasi on-line procedures for scheduling FMS's. These procedures are essentially heuristic rules for the selection of a workstation, a transport device, and an operator. The selections are made hierarchically, and the three procedures differ in the way these selections are placed in the hierarchy. In the Function Sequential Scheduling (FSS) procedure, the selections of workstation, transport device, and the operator are made for each operation sequentially. The Function Integrated Scheduling (FIS) makes all the three assignments simultaneously. In the Function Phased Scheduling (FPS) procedure, the workstation assignments are completed first, in phase one; then, the transport device and operator assignments are made in phase two. Simple rules, analogous to dispatching rules, were used for all the selections. The workstation that can start the earliest is assigned. The transport device that can be available earliest is used. Similarly, the operator that can finish his activities at the earliest moment is selected for the operation. For the selection of jobs, four dispatching rules

were used: SPT, SPT/TOT, SPT*TOT, and EFTA (Earliest finishing time with alternatives considered). When the makespan is used as the criterion, the SPT/TOT rule performed the best. This result is the same as that of Stecke and Solberg (1981), although their criterion was the production rate. Slomp et al. concluded that FPS performed worse than FIS and FSS, and that FIS is to be favored when there is heavy workload on transport devices and operators, otherwise FSS is recommended.

Heuristic rules are excellent for dynamic problems. Some of them, for instance, SPT, have very little computational overhead, and still give good results. A scheduler will need to only decide on the heuristics to be used for each type of decision: resource allocation, job allocation, tool change, etc. Then the system can run automatically using the rules. However, the scheduler has to live with suboptimal results. The performance of the rules change considerably depending on the system state. Thus it is not always easy to select appropriate rules.

Control Theoretic Approach

Gershwin et al. (1986) present a control theoretic perspective on the production control aspects of FMS. Kimemia and Gershwin (1983) presented a closed loop hierarchical formulation of the FMS scheduling problem. Akella et al. (1984) describe the performance of a simulated model of an actual facility using this hierarchical policy. A FMS is considered where parts are manufactured to meet a certain demand which could be varying over time. There is a penalty for exceeding the demand as well as not meeting it. Thus it would be best to produce exactly at the same rate as the demand; but this cannot be done on account of the failure of the machines. Stochastic machine failures are considered, which are smoothed by providing buffers of the parts.

The heart of this control theoretic scheduling policy is to maintain a steady safety buffer of the parts produced in the FMS, as long as it is feasible to do so. The operational state of the FMS can be defined by a vector α, the ith component of which indi-

cates the number of operating machines of type *i*. The production rate **u** is defined as a vector whose *j*th component represents the instantaneous rate of production of part type *j*. A characteristic of the framework is that it is constrained to find a solution within the production capacity of the FMS. For each machine state $\underline{\alpha}$, a capacity state $\underline{\Omega}(\underline{\alpha})$ can be defined which is the set of possible production rate vectors $\underline{u}$. For each $\underline{\alpha}$, a safety buffer level $H_j(\underline{\alpha})$ is defined for each part type *j*. At any point in time, the production rate vector $\underline{u}$ is found by solving the linear program.

Linear Program:

Minimize $\underline{c}\,\underline{u}$

subject to

$$\sum_j \tau_{ij} u_j \leq \alpha_i \text{ for all } i \text{ (feasible production rate)}$$

$$\underline{u} \geq \underline{0}$$

where $\underline{c}$ is a vector of cost coefficients for each part type, and τ_{ij} is the load on machine *i* due to a unit vector of production rates. Thus, depending on the vector $\underline{c}$, and the machine state $\underline{\alpha}$, a particular production rate $\underline{u}$ is chosen. As an approximation, for a given buffer level x_j, c_j is estimated by

$$c_j(x_j) = A_j(\underline{\alpha})(x_j - H_j(\underline{\alpha}))$$

where $A_j(\underline{\alpha})$ is a positive quantity indicating the relative value and vulnerability of part type *j*.

Their hierarchy is based on the frequency of events. Decisions about events of higher frequency is made at a lower level of hierarchy.Three levels of hierarchy are suggested. The frequency of events at a particular level is an order of magnitude smaller than that at a lower level. The top level of the hierarchy calculates the vector H_j for each machine state $\underline{\alpha}$. As an approximation, Akella et al. (1984) suggest

$$H_j = d_j \frac{T_j}{2}$$

where $\underline{d}$ is the vector of demands for the parts, and T_j is the average mean time to repair of all the machines part j visits. A higher hedging point H_j is required if more time is required to repair the machines, and if there is a high demand for a part type j. The safety stock needs also to be higher if the part is more vulnerable to failure. They suggested

A_j = number of machines that part type j visits.

These parameters are suggested only as approximations. More complicated formulations are available. As can be noticed, the need to determine these occurs only if the configuration or the part mix of the FMS changes.

At the middle level, calculations need to be done more frequently. From the parameters given by the top level, the vector of cost coefficients $\underline{c}$ is calculated, and the linear program is solved. This is to be done on-line. This results in a vector of production rates $\underline{u}$. The lowest level of the hierarchy dispatches parts in such a way that the flow rates $\underline{u}$ established at the middle level are achieved.

A rigorous formulation of the above hierarchical framework is provided by Gershwin (1989). The simulation results of Akella et al. show that their hierarchical scheduling methodology produces high output with low work in process. It is able to track the demand on the system very closely while coping with disruptions due to machine failure.

As can be seen, the closed loop control policy is tailored for a dedicated FMS producing a particular part mix. The tooling of the FMS, buffer capacity and other constraints are not considered. It is assumed that the input of a part is a sufficient control decision, and the (alternate) routing, possible deadlocks, blocking, etc. need not be considered. Further, the possible effect of long total processing times of parts in the FMS on the feedback loop is ignored.

Simulation Based Approach

Recently some authors have presented discrete event simulation as a scheduling tool. Basically, simulation is proposed as a tool to evaluate the dispatching rules. This is not an entirely new approach: the study by Conway (1965, 1965b) was based on simulation. What is new is that the authors suggest using data from the actual FMS for simulation. Thus a simulation model of the 'real production system' is built. The simulation model is initialized to the exact current state of the factory. The dispatching rules are then tested on this model. Obviously, a large amount of data gathering is necessitated by this approach to initialize the model.

Although not specifically targeted for a FMS, FACTOR [Anon., 1990] is an example of a commercial software product based on this approach. It takes over where MRP leaves off, and it carries out detailed finite capacity scheduling of the factory using simulation. Choices of sequencing rules are provided. Simulation can be carried out using any of these rules and various other scenario - shift changes, maintenance, failure of machines, etc. Detailed performance reports for all of these scenarios, as well as compatible information are presented. Gantt charts, and shop orders are also generated. FACTOR [Grant, 1989] provides two standard interfaces: a modeler's interface for the person who builds and maintains the simulation model, and a scheduler's interface for the operator using the model on a daily basis. FACTOR may be used both as a scheduling tool and for 'what if?' analysis of scheduling alternatives. It may be used to reschedule in the event of unforeseen events on the shop floor. Practical experience with FACTOR is described in [Robbins, 1986].

Yancey and Peterson (1989) report on the synthesis of expert systems with FACTOR. Two expert system modules have been incorporated into the FACTOR system. Output Analysis System (OAS) is an expert system shell which generates rulebases for analyzing a schedule. These rulebases then detect problems and suggest improvements to

the schedule generated by FACTOR. The solutions are incorporated into a new revised schedule. The shell generates an inference engine which is embedded in the rulebase. This makes for efficient running of the expert system. Rulebases are created as needed by the modeler, with a particular purpose in mind - for example, to detect late orders, or to suggest solutions for late orders, etc. Similarly, Site Specific Tailoring (SST) is used to create rulebases for making decisions during simulation. The rulebases implement sequencing decisions, resource selection, etc. SST rulebases tend to be small (less than ten rules). SST provides a customized inference engine for each rulebase. When a rulebase is invoked, it retrieves context sensitive data from FACTOR and after running its rules, returns its inference.

Grant et al. (1989) propose a framework that carries out adaptive and predictive scheduling in real-time. This approach derives from FACTOR, and is based on five components. SCHEDULER generates a schedule of the operations of the factory. Like FACTOR, this is done by simulating a detailed model of the plant. MONITOR is their next module: it tracks the on-going progress of the schedule. This consists mainly of data sampling, and communications with the processes. Performance measures are calculated. COMPARATOR compares the performance with the planned schedule. This module will signal when the performance is beyond control levels, called "performance tolerance fences". RESOLVER uses Expert Systems technology to determine what, if any, action needs to be taken. It decides how cost effective rescheduling is going to be. In the event of rescheduling, it determines the method to do so. ADAPTOR uses discrete event simulation again to determine a new schedule to patch up with the original schedule. An expert system is used to select the heuristic, time horizon, and the components involved for this adaptive scheduling. Simulation experiments carried out to determine the feasibility of this methodology have reportedly yielded encouraging results.

Davis and Jones (1989) propose concurrent simulation to carry out production scheduling. In their scheme, multiple simulators of a production facility are initialized to

the latest state of a FMS. These simulators are stopped after some time. The simulations are then analyzed as terminating simulations to decide on the best rule to use.

Synergism between expert systems and simulation is used in an on-line scheduling system called ESS (Expert System Scheduler). Jain et al. (1989) describe the development of a scheduling system which communicates on-line with the factory control system, generating schedules in real-time. The scheduling decisions are based on the expertise of an experienced scheduler. The system is based on LISP, and uses object oriented concepts for both the expert systems and simulation. It is possible to run the simulation backward in time to obtain starting time-windows for jobs. The major reason for implementing backward simulation was implementation of JIT concepts. With this concept the job can be started at the latest possible time. Conflicts are resolved by shifting individual jobs in the schedule forward or backward. The system reacts interactively with the user, and permits solicitation of more information by the user, or changing of the schedule. At the time this article was written, the system had been controlling production at an automated manufacturing facility for several months.

Manivannan and Banks (1989) propose a framework for a knowledge-based on-line simulation system (KBOLS) to control the manufacturing shop floor. The main component in their scheme is a knowledge based controller (KBC) which is modeled after blackboard systems in AI. The blackboard system was originally proposed in HEARSAY-I speech understanding project [Barr and Feigenbaum, 1981]. It has multiple 'knowledge sources' (KS) , which are expert systems, each with their own field of expertise. KS's are activated under specified conditions. A 'scheduler', which is itself a specialized knowledge source sequences the different knowledge sources. These KS's work cooperatively to solve the problem at hand. KS's communicate with each other through generally accessible messages - hence the name 'blackboard'. Blackboard architecture based planners are particularly suitable [Young, 88] for factory scheduling: 1)

they can be driven by external events posted on the blackboard; 2) independent knowledge sources lend themselves to ease of modifications.

Four independent knowledge bases (KB) are proposed for KBC. The factual KB contains historical knowledge and the current system status. The procedural KB has algorithms and procedures for loading, routing, and scheduling of parts and processing stations. The temporal KB keeps track of time of event occurrences on the shop floor. The on-line simulation KB has rules to determine when to execute the simulation and to set the simulation parameters. KBC monitors the shop floor activities and when a fault occurs, it collects all the data from the concerned cell. A fault diagnosis is carried out, and if the fault has occurred before, a learning module (which has learned the previous response) provides the steps to be carried out. If it is a new fault, all the factual, procedural, and temporal knowledge are collected. The on-line simulation KB is called to determine whether a simulation is necessary, and what parameters are needed for simulation. The KBC then calls the on-line simulator to determine the best control activity by way of simulation. This decision is then passed both to the learning module, and the shop floor cell.

The manufacturing simulator proposed by Manivannan and Banks for KBOLS system has software to model shop floor activities and to perform resimulations. It has the capability to interface with the rulebase, the human operator, and the KBC. The simulation results are analyzed by the on-line simulation rulebase. The rulebase then selects the control decision to be carried out. All the above activities are carried out on-line. Since this particular system is in the planning stage, it is too early to discuss its performance.

Wu and Wysk (1989) report on a multi-pass expert control system (MPECS) which uses discrete-event simulation for on-line control and scheduling in flexible manufacturing systems. In their system, simulation is used to evaluate dispatching rules. An expert system is employed to compile the set of candidate dispatching rules [Wu and

Wysk, 1988]. This expert system has a learning module to learn from past decisions. The expert system generates the candidate set on the basis of current system objectives, system status, and the characteristics of on-going operations. A 'Flexible Simulation Mechanism' (FSM) collects all the data on the current system status. A simulation model is then generated based on this data. A series of simulation runs is carried out starting from the current state using each of the candidate dispatching rules for the next short time period (dt), selected by the user. FSM provides performance measures for each of the runs. The rule that results in the best performance is used to generate a series of commands to the real-time control system of the FMS. The FMS is then run for time dt under the 'best' dispatching rule.

Compared to single-pass heuristic scheduling, Wu and Wysk report an improvement of 2.3%-29.3% under different simulation windows (= dt) and measures of performance. Selection among waiting jobs for operation in a machine is, however, just one of the decisions that need to be made on the shop floor. Although Wu and Wysk's control system addresses flexible manufacturing, it is not known how or if other decisions in FMS, e.g. routing selection, tool change, AGV selection, etc. are handled in this system.

Simulation is certainly more tractable than mathematical programming formulations of FMS scheduling problems. With simulation, there is no concern about feasibility, since there is no need to make any simplifying assumptions. The simulation model can be built as close to reality as one needs to. However, if the simulation is carried out with just one rule for each type of decision, then simulation does not serve any decision support purposes. Then, the only purpose of simulation would be prediction - when a job can be expected to be completed, what machine utilizations can be expected, etc. Simulation can work as a decision support tool when there is the possibility to simulate under different decision alternatives. Then informed decisions could be made by looking at the simulation results. When considered as a candidate system for on-line scheduling, response time of the scheduling system is a major concern. The response time would

also depend on the number of candidate rules evaluated. This issue can only be resolved by more investigations into this new method of scheduling.

Artificial Intelligence Based Approach

Artificial Intelligence (AI) appears to be particularly suited to solving scheduling problems because AI was developed to solve problems similar to scheduling - problems involving a large search space, and where human expertise can find reasonable solutions pretty fast. Many researchers have sought to utilize this similarity by using AI methodology to solve scheduling problems. There are four application areas where AI has been used with success: computer vision, natural language processing, expert systems, and planning. For scheduling applications, one is primarily interested in expert systems and planning.

Expert systems consists of three components [Gevarter, 1984]:

1) A knowledge base of facts and heuristics related to the domain of interest. These are expressed in the form of rules of the form - If (conditions) then (actions). When 'conditions' are right in the database, the 'actions' alter the database.

2) A working memory for keeping track of the problem status, the input data for a particular problem, and modifications to the data.

3) An inference engine, which is independent of the domain of interest and selects the rules to be applied as the problem solving process is continued.

It might seem that expert systems are just a collection of If-then rules, and as such, they are similar to conventional computer programs. But the main advantage of expert systems is their decomposition into the three components as mentioned above. Thus, the knowledge about the problem and the methods for using the knowledge are completely separate, which is not the case with conventional programming. The knowledge associated with an expert system can be changed very easily by changing the rules.

Planning, also called problem solving, concerns itself with situations where there is a goal, and different actions have to be planned to achieve the goal. The main emphasis is on the task of stringing together sequences of actions. Given a goal, the planner can find actions to obtain the goal. To carry out the actions, more actions may be needed. A basic problem in planning is the conflict in the actions. One action may adversely impact the effect of another action. In another case, the sequence of actions may be feasible action by action but not as a whole. Typically, planning programs use a search process to find a feasible plan that is in some sense, "good". Hierarchical planning is a common method of planning in which a high level plan is formed first, and then this plan is more and more elaborated until a final, feasible, satisfactory plan is obtained. Nonhierarchical planning generates all the actions at the lowest level without regard to stringing them together, and then tries to resolve the conflict among the actions. In script based planning, previously prepared outlines of plans are used. These skeletal plans are then fleshed out so that the resulting plan meets the current goal. In opportunistic planning, the plan is developed piecewise, and then linked together as opportunities arise. This paradigm is said to be the followed by humans in solving problems.

As pointed out by Gevarter (1984), at first the researchers tried to solve the planning problems without regard for the domain of the problem. But this proved inadequate for the "real world" complex planning tasks. Planning has had to rely more and more on the domain knowledge; and ideas from expert systems were used to capture the knowledge. Thus, the distinction between expert systems and planning systems became less prominent. Now, they can both be called knowledge based systems.

Steffen (1986) has presented a survey of AI based scheduling systems. These systems were developed to schedule production systems, not necessarily a FMS. He discusses the surveyed systems from four perspectives: historical, methodological, application, and implementation. The AI paradigms used by systems historically tracked the development of AI ideas. As better ideas were developed in AI, they were incorporated

into production scheduling. Steffen found that many AI approaches were currently used by the system builders but most approaches were rule based. He points to the common misconception that AI is solely concerned with imitating human behavior. The goal of scheduling research is efficiency not behavioral. Thus, many scheduling systems did not use the human scheduler as the model for emulation. As can be expected from the complexity of the problem, jobshops were the most popular subject for research on AI-based scheduling systems. It is interesting to observe that only 2 of the 51 systems surveyed were operational in the factory environment. This is attributed to implementation difficulties. Kusiak and Chen (1988) have also reviewed a number of AI-based scheduling approaches. They address individual approaches while Steffen (1986) provides a general survey.

Bullers et al. (1980) advocate the use of AI in manufacturing planning and control. They suggest using predicate calculus to solve planning and control problems, particularly at the operational level, in automated manufacturing. It is argued that traditional off-line analyses are too slow and may result in costly mistakes in real-time environments. These real-time decisions should be made by automated control systems having knowledge of the system as well as its status. They present predicate forms that can represent the static and the dynamic states of a production system. For example, MCH-PART (mch, part, t) is a dynamic assertion that states that a certain machine has a part at time t. With these statements in the data base, logic programming can be used to ask various questions from the data base. Logic programming uses unification and/or resolution to come up with an answer to the questions. Examples of such questions are: "Were any parts in the system at time t_1 due before t_2?" , "Given the current time and an SPT scheduling algorithm, when will part p complete its last operation?". Their research is, however, only exploratory and gives simple instances of what could be done. It is not clear if a decision support system was ever based on these ideas or what success it may have had. Considering the slow process of back tracking used in the resolution of logic

programming, it is doubtful that these ideas can result in a practical system that offers real time support.

ISIS is a knowledge based system to schedule production. Its main emphasis is on the constraints of the production system being modeled (Fox et al, 1982; Bourne and Fox, 1984). Although theoretically the search space in scheduling problems could be very large, they found that it was severely curtailed by various constraints. They found that human schedulers spent 80%-90% of their time determining the current constraints and 10%-20% of the time actually working on the production schedules. Categories of constraints defined in ISIS are: 1) organizational goals e.g. due dates, cost, quality; 2) operator's preferences e.g. particular machines for some operations; 3) gating constraints such as operation precedence, resource requirements; and 4) physical constraints, for instance, the size of a machine, life of a tool. Some constraints determine the admissibility of a schedule - these are hard constraints which, if violated, render the schedule infeasible. Other constraints determine the acceptability of a schedule - these rate the schedule on the basis of their desirability.

ISIS is constraint directed in the sense that constraints are used to identify the next state to go to and are also used to evaluate the current state. If the constraints overly constrain the search and progress cannot be made, these constraints are relaxed. ISIS follows a hierarchical planning paradigm. There are four levels in this hierarchy: order selection, capacity analysis, resource analysis, and resource assignment. Each level is composed of three phases: a presearch analysis phase which constructs the current problem, a constraint directed search, and a post search analysis phase which determines the acceptability of the solution. Level 1 selects an order to be scheduled based on the category of the order and its due date. Its output is a prioritized list of orders to be scheduled. The level 2 phase produces constraints for the next level based on the capacity of the plant. Its output is the earliest start time and latest finish time for each operation of the selected order as determined by the order's release and due dates. In level 3,

resources are selected to produce an order. A beam search is carried out in the space of alternative partial schedules. This search is carried out with the help of constraints developed earlier. Level 3 still does not completely commit the resource to a particular time. It merely fixes the reservation time bounds on the machines. Level 4 finally fixes the actual times of the operations with a view to minimize the work-in-process.

The beam search and the constraint based approach of ISIS do limit the search space greatly. But the symbolic manipulations inherent in AI technology are time consuming. In comparison to an optimal search, it is not known how good the solutions produced by ISIS were or how fast (or slow) it was. Although the time bounds are not fully committed until constraints cause them to be bound, ISIS uses horizontal loading - one highest priority order is entirely scheduled, then the next priority is picked up, and so on. It is known that this approach creates "holes" in the schedule - a machine sits idle, even if a job is available, because a more important job is coming (Vollmann et al., 1988). It is not clear how much this problem is alleviated by ISIS's least commitment approach.

Another production scheduling system based on the constraint directed approach is OPAL (Bensana et al., 1988). In this system, a constraint propagation module (CBA) sequences operations of jobs according to the precedence constraints, release dates, and due dates. These constraints for each operation are through all the operations of a job. If the conflicts are all resolved by these constraints, a schedule is produced, and the problem is solved. But if there are conflicts not cleared by following the constraints, OPAL activates a decision support module (DS) to force an ordering on the contending jobs. This decision support module is rule-based. This module is based on the fuzzy set methodology. Each rule is assigned an index which can be regarded as a grade of membership of the rule in the fuzzy set of rules relevant to the goal. These indices function like certainty factors and attach weights to the rules. The rules could be simple priority rules such as SPT or rules linked to the utilization of auxiliary resources or rules pertain-

ing to slack times of operations. A supervisor module guides the search process by alternately calling the CBA and DS modules. Each time the CBA module stops, a new node is generated. The DS module determines the branching of the node. Then, the CBA module is called again to propagate the decisions forced by the DS module. The search is a depth-first, back-tracking type. As can be seen, OPAL is a static scheduling tool. It does not follow the dynamic changes in the shop floor.

Bruno et al. (1986) present a rule-based system to schedule production in a FMS. They use expert systems to capture knowledge about the domain, and queueing network analysis for performance evaluation. The expert system uses rules to select production lots to introduce into the FMS. Primarily, the lots are selected on the basis of the dispatching rule of critical ratio (CR). A lot with highest priority may not be scheduled if a constraint is violated. Production constraints such as release time, needed fixtures, maintenance, etc. are checked. Capacity constraints such as system congestion and throughput are checked by a heuristic based on the mean value analysis of closed queueing network. This module calculates the machine utilization, average queue lengths, and lot throughputs. A simulation model is used to obtain the system state trajectory using the rule base and the performance analyzer. This trajectory is the resulting schedule.

An interesting feature is that the expert system is written in OPS5, a rule-based production system language, while the queueing network analyzer is written in FORTRAN-77. Because of the different data structures used in the two modules, data needs to be translated back and forth. It is not known how much this translation penalizes the system performance.

It is well known that mean value analysis calculates steady state performance. However, a FMS is a dynamic entity where the operating conditions are continually changed by the very actions of the scheduler and by the vagaries of nature. Thus the validity of the results of mean value analysis for use in decisions about production lot introduction is open to question.

A nonlinear planning algorithm for FMS scheduling is proposed by Shaw (1988). Here, the term 'nonlinear' is not to be taken in the mathematical programming sense. It is an AI planning approach where the plans are not formed in a serial fashion, one after the other. The plans are formed in parallel (least commitment) until some constraints force the actions to be serial. This approach is based on the A* search, where one starts from an initial state and by applying successive operators (from a rule base), the goal state is finally reached. It is a heuristic best-first search procedure directed by an evaluator which evaluates a current node on the basis of the estimated cost of the path from the initial state to the goal state (Nilsson, 1980). If this evaluator is always correct, there is obviously no search: one directly gets the path from the initial state to the goal state. But even if the estimate always errs on the conservative side, one is guaranteed to get the optimum result. The operator to be applied at a node is specified as follows:

<Action-name> : <list-of-arguments>

<Precondition> : <list-of-precondition-literals>

<Add list> : <list-of-add-list-literals>

<Delete list> : <list-of-delete-list-literals>

<Resource> : <resource-name>

<Duration> : <length-of-duration>

These operators are available in the rule base. In this methodology, the jobs are individually scheduled using this search procedure. Of course, these schedules are not going to be feasible, due to the simultaneous contentions on the resources. A list (Alternate-list) of operators that are in conflict is then prepared. A plan-revision procedure is used to resolve the contentions. This procedure schedules the operators on an alternate resource as far as possible. A forward chaining procedure propagates the changes. If an alternate machine is not available, then the operators just wait, again necessitating a change propagation. This plan-revision step is a unique feature developed with production scheduling in mind, and is not found in other AI planning literature.

Four evaluators were tested for the A* algorithm mentioned above:

1) f0: Height of the search tree

2) f1: Cumulative processing time + estimated total remaining time

3) f2: Cumulative processing time + imminent operation time

4) f3: processing time + number of operations left

Shaw found that a) good heuristic knowledge is important for improving the computation efficiency of the scheduling algorithm; b) a global heuristic is better than a local heuristic; and c) a domain specific heuristic is better than a general heuristic. As can be expected, the size of the search tree was significantly higher when the number of alternative machines for an operation or the number of machines in the system was higher. This is on account of the higher branching in the search tree. When his A* algorithm was modified for due-date targets, he concluded that there is not a single rule that would dominate in every situation. He suggests selecting the rule dynamically, based on the system state.

Unlike many other FMS scheduling methodologies, this methodology explicitly considers alternate job routing, and incorporates it in the optimization. This approach attempts to obtain the globally minimal make-span. It is claimed that the scheduling system can perform dynamic scheduling, adapting to changes in the FMS environment. Although it will use AI heuristics to limit the search, the search space is still very large and may make it prohibitively expensive to use in practical scheduling problems.

SCORE (Shop-floor Contingency Rescheduling Expert) is an on-line scheduling system which is based on blackboard concepts of AI [Chiodini, 1986; Chiodini, 1989]. It carries out both predictive scheduling (a priori determination of future events) and reactive scheduling (alter the schedule in response to changing shop floor status). Predictive scheduling is done top-down; given the master production schedule, it generates a schedule for future events. Reactive scheduling is carried out bottom-up. This principle aims at localizing the disruptions at the lowest level in the shop floor hierarchy. First,

the sub-assembly schedule is adjusted; if it does not absorb the effect of the disruption, the final assembly schedule is revised. If the effect of the perturbation is high enough, it may cause changes on the master production schedule.

SCORE is event driven. As events occur, they trigger SCORE to take action. The SCORE supervisor is the focus of action in Chiodini's system. It prioritizes the active events posted in the Agenda (a data structure similar to the blackboard), dispatches the tasks (similar to Knowledge Sources) that handle the specific event, allocates an execution time slice to the task to be performed, and coordinates concurrent access to the database by the running tasks. An interesting feature of SCORE is its management of task execution times, which is handled by the SCORE supervisor. Predictive scheduling is carried out in the background, and is scheduled during periods of reduced activity of production lines. The goal in predictive scheduling is to generate a schedule that satisfies as many objectives of the master production schedule as possible, while remaining within the capacity constraints. The activation of this task has lower priority compared to the real-time reactive scheduling.

Shop floor activities are continually monitored and the data base is updated. If an error status is reported, a corrective action request is posted on the Agenda, which triggers reactive scheduling. Reactive scheduling performs a search in the space of alternate partial schedules to find a schedule that still meets the original schedule with the least disruption. When tested under a simulated environment, SCORE has reportedly performed satisfactorily and is now under field test.

Many ideas in AI could be used beneficially in FMS scheduling. Heuristic search is one method that should prove useful. The separation of control, operator, and data is another idea that helps build flexible software for large systems. However, AI methods have, by definition, intensive symbolic manipulation. This raises concerns about their use in real-time systems.

Interactive Approach

All the approaches mentioned in earlier sections can be implemented in an interactive environment. What makes interactive scheduling different is that the scheduling decisions are mainly made by a human operator with the aid of the computer. The main focus of interactive scheduling has been on the computer generation and display of Gantt charts.

Godin (1978) presents a review of interactive scheduling. He describes various earlier attempts at computerization of production scheduling and scheduling of other systems. He reports the consensus that interactive scheduling combines the best of both humans and machines. He offers some hypotheses on why interactive scheduling systems have not fulfilled their promise: dynamic nature of production systems, computer ignorance, unavailability of suitable hardware and software, uniqueness of each scheduling situation, etc. Some of the above impediments have now been removed with the introduction of hardware and software highly conducive to user oriented computer utilization. Correspondingly, some applications have been reported in the literature.

Adelsberger and Kanet (1989) provide a more recent review of the state of art in interactive scheduling. They describe the main components of an interactive scheduler:

1. Graphics interface capable of providing a pictorial representation of the schedule. The interface gives the status of each resource over time. Ideally, it should provide the facilities of zooming, panning, and scrolling to view all the schedule at the same time. It should also permit viewing the schedule from the perspective of the jobs. The authors describe a large number of existing and planned systems with these capabilities.

2. A schedule editor for manually generating and manipulating schedules. Minimally, the system should enable adding or deleting an operation one by one. More sophisticated systems let the user i) change the completion time of an operation, ii)

modify the quantity in the production order, iii) change the allocated resource for a job and iv) split or combine operations. Advanced editors test the schedules for violation of constraints dynamically to alert the user. Some systems depict the effects of a schedule change in an animated fashion. As one operation is 'dragged' across the screen, its effect on other operations and machines is animated. A nice feature of some systems lets the user 'undo' a decision. The ultimate goal is to have AI based software which recommends specific changes for the scheduler to try.

3) Database manager for accessing information. This module retrieves data from the production planning system, engineering database, and the shop floor status information system.

4) Schedule evaluator for measuring the performance of schedules. Many current interactive schedulers do not have this module. An evaluator should provide feedback to the scheduler on a number of performance measures: due date performance, work in process, lead time, machine utilization etc.

5) Automatic schedule generator. Many systems reviewed by the authors do not have this module. At the minimum, this module should provide a feasible schedule, so that the operator can improve on it. Of course, any of the scheduling methodologies described in the earlier sections, e.g. mathematical programming or AI could be used to generate the schedule.

Many of the developers described by Adelsberger and Kanet are incorporating expert systems technology into their systems. An expert system may be used to implement constraints, or preferences or just sound advice. A user could add or delete the rules as needed.

An interactive scheduler developed for actual industrial use is described by Jackson and Browne (1989). This scheduler has some of the components described by Adelsberger and Kanet, namely, a graphics interface, a schedule editor, a database manager, and a schedule generator. This scheduler is built on the premise that it is almost

always impossible to obtain optimal solutions to most real-life scheduling problems, and good solutions can be generated by a human operator by editing , with the help of a computer, a schedule built by one pass of a scheduling heuristic. A number of dispatching rules are provided for the purpose of creating the schedule. This system was developed with ergonomic considerations in mind. An application keypad provided many functions for viewing and editing the schedule. Provision is made for scrolling and panning.

Conway and Maxwell (1986) describe an interactive scheduling system called LLISS (Low Level Interactive Scheduling System) developed by them in collaboration with Hewlett Packard Laboratories. Their view on scheduling optimization is instructive:

> Scheduling is inherently an exceedingly complex process. There are simply too many variables, and too many possible solutions, for there to be any hope of obtaining optimal solutions to any non-trivial scheduling problems. Anyone who claims otherwise either does not understand the problem, or is being carefully deceptive in the use of the term.

Their system assists the human scheduler in organizing the information, permitting selective retrieval and display of the information. It communicates automatically with the external entities and predicts the implication of each individual scheduling decision. LLISS communicates with: 1) Manufacturing engineering, for the types of available machines, and the work they can perform; 2) Production planning, to obtain the tasks to be performed, to acknowledge when a task is completed or is to be completed; 3) Material control, to communicate the raw material requirements; 4) Maintenance control, to obtain machine availability, and the repair status; and 5) Machine control, to obtain the information on the status of a job, to obtain failure information and to communicate scheduling orders.

LLISS permits schedule editing. Tools are available to reschedule using different heuristics - due date, priority, lateness etc. Tasks may be split or combined; quantities

and priorities may be changed. The schedule may be viewed by time, machine, task, or status.

As pointed out earlier, computer aided scheduling is now feasible with the introduction of powerful microcomputers with large amount of core memories and high performance. These machines also have very good graphics capability. These qualities make for good interactive programs. However, there are not many published accounts of interactive scheduling systems. Most of the systems reported in [Adelsberger and Kanet, 1989] were developed in Europe.

Summary

Six different approaches for scheduling a FMS were reviewed. Many authors are of the opinion that mathematical optimization is too intractable for practical FMS scheduling applications. Heuristic based approaches are appealing from the view point of their simplicity, but may adversely affect efficiency on account of their short time horizons. Work on approaches based on control theory, AI, and simulation have been all going on for some time, but none have established their prominence in the field. There is not much reported work done on interactive scheduling.

On account of the on-line requirements of dynamic scheduling, complicated iterative models are not likely to succeed in random FMS. From this perspective, discrete event simulation offers some promise, since it is not iterative. However, a single pass simulation does not help because it does not evaluate alternatives. Thus a very limited evaluation of some scheduling alternatives is desirable, even within simulation. This may be done with knowledge based simulation where the alternatives are chosen by an embedded expert system. This is the rationale underlying the research approach described in the following chapters.

CHAPTER IV

GOALS, SPECIFIC OBJECTIVES AND ASSUMPTIONS

Research Objectives

The main goal of this research was the development of a comprehensive methodology for scheduling and controlling random FMS that is capable of generating consistently good solutions in near real-time. Two requirements are placed on this methodology: on-line processing, and consideration of all the specified constraints.

With this goal in mind, the following objectives for the research have been identified.

Objective 1. Methodology

Develop the general outline of a comprehensive methodology for scheduling and controlling random FMS that is capable of generating consistently good solutions. This outline should state the main components (or modules) of the scheduling system and their interactions. It should show how these interactions will lead to the solution of the problem.

Objective 2. Object-Oriented Representation

The objective here is to determine the classes and subclasses of objects required for representation and expression of the random FMS scheduling problem in the object oriented paradigm. The resulting representation should, ideally, i) facilitate concise and

intuitive representation of the problem, and, ii) facilitate the implementation of the solution methodology defined in fulfillment of objective 1.

Objective 3. Development of Framework

Develop an object oriented framework for the interactions of the components within the system being developed that is capable of carrying out the logic of the solution methodology while operating in a dynamic environment, on-line. The framework will include: i) the messages needed to be passed between the objects, ii) the methods these messages will invoke, resulting in scheduling and control decisions, and, iii) explicit representation of the methodology's response to the events occurring in the real environment.

Objective 4. Measures of Merit

Develop and validate relevant measures of merit for evaluating alternative scheduling and control methodologies for random FMS. These measures may be qualitative and/or quantitative.

Objective 5. Evaluation

The fifth objective is to evaluate the scheduling methodology developed in this research. To do so, the ideas will be implemented in an object oriented environment. The implemented framework will then be evaluated on the basis of measures developed in fulfillment of objective 4 and compared to alternate methodologies.

Objective 6. Further Research

Finally, one objective of this research is to identify what further work needs to be done, as an extension of this effort or otherwise, to bring about a solution of the problem: scheduling and control of nondedicated FMS.

Research Assumptions

Unfortunately, scheduling approaches tend to have a very limited domain of application. The methodology will work only for the particular type of problem to which it is addressed. This is on account of the detailed nature of the solution offered.

This research is addressed to a random FMS: production of parts in limited quantities, where the part mix is not fixed in advance. The frequency of orders, or the production control policy is such that parts are made only to order.

The configuration of the FMS is assumed to consist of a loading station, a number of versatile processing equipments, and an unloading station, all tied together by AGV's. An operation on a part can potentially be carried out by one or more alternate machines provided the machine is loaded with the right tool. Limited buffers exist where parts awaiting processing can be stored.

The FMS is assumed to be controlled by an operator, assisted by a supervisory computer. The supervisory computer obtains the parts requirement, due dates, and available dates from a host computer. It is assumed that preliminary planning of these jobs is done such that a feasible loading on the FMS results - inordinate queues do not build up at the loading station.

The supervisory computer directs the AGV's and other equipments through programmable logic controllers (PLC) or computers associated with them. Thus, the function of the supervisory computer is only to determine the next steps to be taken by these controllers. The actual real-time process control will be carried out by the PLC's or other computers associated with the processes. Thus, although the process needs to be controlled in real time, the run time criterion is not so tight for the supervisory computer. This research is directed to the needs of the supervisory computer.

CHAPTER V

PROPOSED METHODOLOGY

Introduction

The problem addressed in this research is a dynamic problem in which orders arrive randomly to a FMS. The scenario could be described as follows. The FMS consists of a number of numerically controlled machines with limited tool capacity in their tool magazines. The tool magazines are supplied from a tool crib which has a limited number of copies of each tool type. Each NC machine has an input and output buffer of finite capacity. The machines fail randomly and require random repair time for repair.

There is a load/unload station for the whole FMS which serves as a material interface. Figure 1 on next page shows a schematic of the physical arrangement. All incoming materials arrive at this station and all finished parts are dispatched from here. Each part requires a particular type of pallet. There is a limited number of copies of each pallet type. A part is loaded into its requisite pallet at the load/unload station. The pallets are transported by an automatic vehicle to the selected machines in their routings. When the processing on the part is finished, it is finally transported to the load/unload station. There are a limited number of these vehicles, and there is a time matrix for the time required for inter-station transport.

Each order arriving at the FMS is for a batch of parts of a specific type. The part requires a number of processes to be carried out. The number of processes needed is random for each order. These processes could have one or more random alternate machines on which they can be carried out. Each process, however, requires the same

random processing time and needs a particular type of tool. The orders can have one of two priorities: high and low. Each order also has a due date assigned to it.

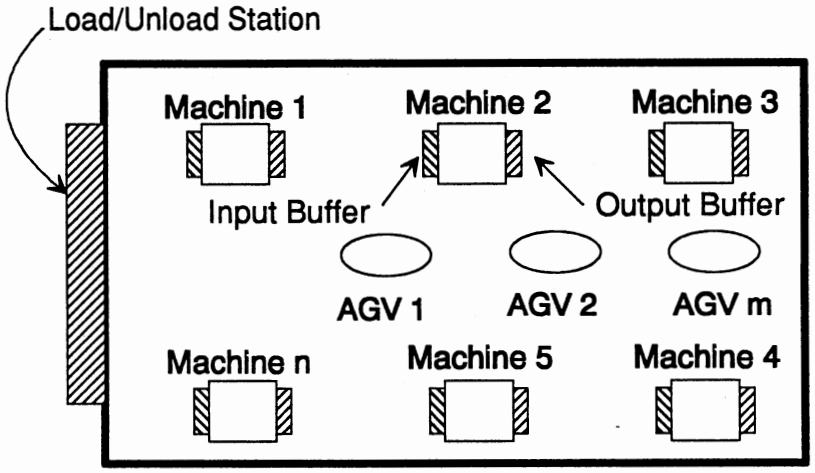

Figure 1. Flexible Manufacturing System Configuration

A machine can perform a process only if it is up (not failed), if it is one of the alternate machines in the routing, and if it has the requisite tool. The normal flow of a work part is from the load/unload station to the input buffer (queue) of a selected alternate machine for its first operation. Then the operation is carried out and it moves to the output buffer of the machine. If the output buffer of a machine is full, the part blocks the machine from further processing until there is space in the output buffer. The work part then moves on to other machines in its routing and finally moves back to the load/unload station. When all the work parts for an order are finished, the order is completed.

A setup of the FMS involves stopping of the NC machines and moving unneeded tools to the tool crib while bringing in needed tools from the tool crib. It may also involve moving a tool from one machine to another machine.

The problem is to plan the release of the orders as they continue to arrive, the allocation of the operations to one of the alternate machines, the sequencing of the parts once they are released, and the planning of the setup of the tools.

Some previous researchers (e.g. Stecke, 1983) have addressed a selection question, in which there are a number of job types with their associated loads which continue to arrive at the plant in question. The problem is to select some of these jobs that could be allocated to the FMS. (The others might go to some other department or to subcontractors). In so doing, the choice among the alternates is simultaneously made, and the machines are tooled with the requisite tools. The problem is then solved at least until the product mix is changed. This has been called dedicated mode of operation by some authors.

Another problem concerns the situation where the job types are not preselected, and various random jobs (at least of a large variety) continue to arrive at the FMS. This means that the machines have to be tooled and retooled again as the jobs arrive. This type of FMS, as described above in detail, has been called a random FMS.

This chapter describes some solution approaches and heuristics developed to address the problem described above. Then these approaches and heuristics are incorporated into a proposed methodology for scheduling and controlling a random FMS.

Solution Approaches

The solution approaches could be broadly classified as simultaneous solution and hierarchical solution. Some discussion of these appears in the literature review (Chapter III). This section and the next present a more focused treatment of them as well as some heuristics proposed in this research.

Simultaneous Solution

Hutchison et al. (Chapter III, page 24) proposed an approach in which mathematical programming is used to simultaneously allocate the machines (from the routings) and the time windows when the process is carried out. In this approach, however, no limit is placed on the tool capacity, the number of pallets, the AGV's, the number of buffer spaces etc. The formulation is also static: it considers a number of jobs at a time, not a stream of incoming jobs. They found that this approach did not offer significant improvement over a simpler decomposition approach where machine allocations are carried out separately from the machine scheduling. Furthermore, the branch and bound search is expensive in terms of computer resources, particularly for real time application. For these reasons, this simultaneous approach was not investigated in this research.

Hierarchical Solution

Shanker and Tzen (Chapter III, page 26) proposed an approach in which the problem is decomposed into two sub-problems, release and dispatch:

1) Release. From among the arrived jobs, select the jobs to be scheduled in the next planning horizon (the others will wait until the next selection). The selection simultaneously carries out the allocation of the processes to the machines. In what follows, this decision process is called "releasing". This approach assumes that a planning cycle exists in which the orders are selected for each cycle. These orders are selected so that the machines' tool magazine constraints are not violated. A tooling setup is then incurred: the machines are tooled for the selected orders. The released orders are then completed and the cycle begins again.

2) Dispatch. The released jobs are then scheduled on the machines (allocating time windows for the operations). This problem, of course, is the traditional job shop

problem with N jobs, M machines. A mathematically rigorous search approach or a simple priority rule based approach may be used.

Their objective is to minimize the overload/underload from the scheduling period (or planning horizon), and to minimize the tardiness. They proposed a mathematical programming formulation for sub-problem # 1 (Chapter III, page 26). As can be expected, the program is too complex for on-line application, and they suggested heuristics to solve the releasing problem. They use priority rules to solve the second sub-problem. It might be observed that this approach takes care of incoming streams of jobs directly: jobs that arrive after the release of selected jobs simply wait for the next release cycle with other unselected jobs.

Release Heuristics

This section describes some heuristics that can be used for the release part of the hierarchical solution described above. Many heuristics could be used for this purpose. Essentially, they select some orders to be introduced into the FMS from the load/unload station. In doing so, they attempt to attain some specified objective.

Fixed Period Release

Moreno and Ding (1989) investigated the same hierarchical problem as studied by Shanker and Tzen (1985). They presented another heuristic for the combined objective of workload balance and minimizing tardiness for the release of work parts into the random FMS. They reported that their heuristic gave better results than that of Shanker and Tzen.

Heuristic 1. The following is the pseudocode for a heuristic based on the heuristic (#2) of Moreno and Ding. Notation is as listed below.

N	Number of orders not yet released
D_i	Due date of order i
p_{ik}	Processing time of operation k for each work part of order i
A_{ik}	Vector of alternate machines on which operation k of job i can be performed
$Slot_j$	Tool slot capacity of machine j
P_i	Priority of order i
H	Length of scheduling period
L_j	Current load on machine j
op_i	Number of operations for order i
n_i	Number of processes for order i
Pend_List	Prioritized list of the orders which have not been released
T	Current time
$STATUS_j$	Status of machine j, $STATUS_j$ = {Up, Down}
$Tool_{ik}$	Tool required for operation k of order i
$Tools_j$	Set of tools mounted in machine j
$AvTools_t$	Number of available tools of type t
LoadBal	Sum of overload/underload of all the machines
Route	A vector of machine indices. For order i, Route(r) ε A_{ir}
Rel_List	Final list of orders to be released
t_{ik}	Tool type required for operation k of order i
Utilization	Vector of loads on the machines in the FMS
o_i	Number of work parts for order i
M_{ik}	Machine allocated for operation k of order i

1. Initialize all the variables with correct values;

$$\text{LoadBal} := \sum_{j} H$$

2. For i = 1 to N do

A. Let $SLACK = D_i - T - \sum_{k=1}^{op_i} p_{ik} * n_i$

B. Insert i into Pend_List in descending order of the priority P_i, placing orders with the same priority in ascending order of SLACK

3. While there are elements in Pend_List

A. Let i = First element in Pend_List

B. Remove i from Pend_List

C. Let numRoute = $\prod_{k=1}^{n_i}$ size (A_{ik})

D. Let chosenRoute = nil; testLoad = LoadBal

E. For n = 1 to numRoute do

i. Let freq = n; newLoad = 0

ii. Initialize the vector Route

iii. For k = 1 to n_i do

1. Let numAlternates = size (A_{ik})

2. Let position = (freq - 1) mod numAlternates + 1

3. Let route (k) = A_{ik} (position)

4. Let freq = (freq - 1) div numAlternates + 1

{Check if route is feasible}

iv. feasible = true

v. For k = 1 to n_i do

1. Let j = Route (k)

2. If not (($STATUS_j$ = Up) and

((($Slot_j$ - size ($Tools_j$) > 0) and ($AvTools_{t_{ik}}$ > 0)) or (t_{ik} ε $Tools_j$))) then

feasible = false

vi. if feasible then

1. For k = 1 to n_i do

A. newUtilization = Utilization (Route (k)) + p_{ik} * o_i

B. If (newUtilization > H) then

newLoad = newLoad + newUtilization - H

else

newLoad = newLoad + H - newUtilization

2. if newLoad < testLoad then

A. Let chosenRoute = Route

B. Let testLoad = newLoad

F. If not (chosenRoute = nil) then

i. Add i to the Rel_List

ii. For k = 1 to n_i do

1. Let j = chosenRoute (k); M_{ik} = j

2. Utilization (j) = Utilization (j) + p_{ik} * o_i

3. if (t_{ik} ∉ $tools_j$) then

A. Add t_{ik} to $Tools_j$

B Let $AvTools_{t_{ik}}$ = $AvTools_{t_{ik}}$ - 1

4. If (Utilization (j) > H) then

LoadBal = LoadBal + Utilization (j) - H

else

LoadBal = LoadBal + H - Utilization (j)

4. Return the Rel_List as the list of jobs to be released, and M_{ik} as the list of allocated machines

Basically, this heuristic carries out the following steps.

1. Put all the waiting jobs into a list in descending order of their priorities and ascending order of slacks.

2. While there is a job in the list

a. Pick the first job.

b. For all the possible machine allocations for this job:

A. If the allocation is feasible, calculate the machine loadings. (feasible = the machine is up and there is a tool slot available in the tool magazine of the machine, or the tool is already in the tool magazine)

B. Select the allocation that causes the minimum overload/underload for the scheduling period.

The constraints of pallets, material handling, and buffer space are not explicitly accounted for in this formulation. Using the above heuristic, jobs are selected for the next planning period, machines are retooled, and the jobs are dispatched through the FMS using a dispatching rule of choice. When all the jobs are done, the cycle begins again.

Although pallets, material handling devices, and buffer space limits are not considered in the above, these constraints can be handled via the same priority rule that is

used to sequence the parts for a machine. For instance, after a work part is released, it queues up for the pallet and when a pallet of the correct type is available, a work part is selected as per the priority rule.

The above described approach for controlling a random FMS is called "fixed period release".

Variable Period Release

The number of buffer spaces in the FMS can be brought to bear on the scheduling process by considering the congestion of the FMS. When too many parts are released into the FMS, it is congested and the machines are blocked frequently. This downgrades the efficient operation of the FMS. This is especially noticeable in an FMS because most FMS's have very small amounts of buffer space at the machines. A variation on the fixed period release is to release a number of parts which is a factor of the total number of pallets which can be accommodated in the FMS. Determination of this factor is not an on-line problem. It can be determined at the planning stage of the FMS by simulation. Using this factor, the planning cycle will be variable. The planning period in the Fixed Period release is quite arbitrary anyway, and, because of queueing delays, all of the released jobs are never finished in the planning period. It is conjectured that by selecting just enough jobs to be released to avoid congestion, better performance can be obtained. Three heuristics for this release are described below.

Heuristic 2. The selection of the orders and the allocation to the machines is done using a heuristic whose pseudocode is given below. This heuristic, developed in this research, is similar to heuristic # 1 of Moreno and Ding. The differences are in using variable periods, where they use fixed periods, and in checking for feasibility only at the last stage of the algorithm. Besides the notation given earlier, the following notation is used.

NUM	Target number of work parts to be released to avoid congestion
Proc_List	Ordered list of the process times
Order_List	List of orders corresponding to the order of the process times
Op_List	List of operation stages corresponding to the order of the process times

1. Initialize all the variables with correct values
2. For i = 1 to N do

   A. Let SLACK = $D_i - T - \sum_{k=1}^{op_i} p_{ik} * n_i$

   B. Insert i into Pend_List breaking ties in the following order:

      i. orders with negative slack always come before orders with positive slack

      ii. orders with higher priority come before orders with lower priority

      iii. orders with lesser slack come before orders with higher slack
3. Let selectedNum = 0
4. While there are elements in Pend_List and selectedNum < NUM

   A. Let i = First element in Pend_List

   B. Remove i from Pend_List

   C. Insert i into Rel_List at the last position

   D. Let selectedNum = selectedNum + o_i
5. Let N = size (Rel_List)
6. For i = 1 to N do

   A. For k = 1 to n_i do

      i. Let load = ($p_{ik} * o_i$)

      ii. Insert load into Proc_List in descending order of magnitude

         1. Let ind = index of load in Proc_list

      ii. Let Order_List (ind) = i; Op_List (ind) = k
7. For j = 1 to size (Proc_list) do

   A. Let i = Order_List (j) ; k = Op_List (j); chosenMachine = A_{ik} (1)

   B. For m = 2 to size (A_{ik}) do

      i. If Utilization (m) < Utilization (chosenMachine) then

         Let chosenMachine = m

   C. Let Utilization (chosenMachine) = Utilization (chosenMachine) + Proc_List (j)

   D. Let M_{ik} = chosenMachine

{check for feasibility}

8. Let feasible = true; i = 1
9. While $i \leq N$ and feasible
    A. For $k = 1$ to n_i do
        i. $j = M_{ik}$
        i. If $STATUS_j$ = Down then
            1. Let feasible = false
        else
            If $t_{ik} \notin Tools_j$ then
                If $AvTools_{t_{ik}} = 0$ or size ($Tools_j$) = size ($Slot_j$) then
                    Let feasible = false
                else
                    1. Let $AvTools_{t_{ik}} = AvTools_{t_{ik}} - 1$
                    2. Insert t_{ik} into $Tools_j$
    B. Let $i = i + 1$
10. If not feasible then
    A. Remove last item from Rel_List
    B. Go to step 5
else
    Return Rel_List as the list of orders to be released and M_{ik} as the list of machine allocations

Basic steps in this heuristic are:

1. Arrange the jobs into a list in order of the slack time and priority.
2. Pick the first jobs in the list so that total number of parts to be introduced is equal to the desired number to be introduced.
3. Arrange all the total process times of the selected jobs into a list with the largest one at the top.
4. Pick the first process, and among the machines it could be assigned to, assign it to a machine that has the least loading so far. Update the loading of the machines.
5. If all the jobs are not assigned, go to step 4.
6. Now check for the feasibility of the assignment.

    If the assignment is not feasible,

remove the last job in the list of selected jobs and go to step 3.

If feasible,

end of heuristic.

The above heuristic gives an evenly balanced load on the machines. Hereafter, this release heuristic is called Balanced Release. Since the feasibility check is done only at the last point, this heuristic saves many computer expensive feasibility checks where the tool constraint is not very restrictive.

Heuristic 3. If the machines are, however, very constrained (small tool space, or few tools) many jobs will be rejected and the number of parts will be much less than the congestion limit. A third heuristic can be used to get more jobs released (although it does not achieve as good a balance). This heuristic assumes the existence of a function coeffOfVar that calculates the coefficient of variation (standard deviation / mean) given a vector of numbers. This heuristic, developed in this research, differs from heuristic 1 in that it tries to minimize the coefficient of variation of the utilization of machines, and not overload/underload of the machines, as originally proposed.

1. Initialize all the variables with correct values.
2. For i = 1 to N do

   A. Let SLACK = $D_i - T - \sum_{k=1}^{op_i} p_{ik} * n_i$

   B. Insert i into Pend_List breaking ties in the following order:

      i. orders with negative slack always come before orders with positive slack.

      ii. orders with higher priority come before orders with lower priority

      iii. orders with lesser slack come before orders with higher slack.

3. While there are elements in Pend_List

   A. Let i = First element in Pend_List

   B. Remove i from Pend_List

C. Let numRoute = $\prod_{k=1}^{n_i}$ size (A_{ik})

D. Let chosenRoute = nil; testVariation = coeffOfVar (Utilization)

E. For n = 1 to numRoute do

i. Let freq = n; newLoad = 0

ii. Initialize the vector Route.

iii. For k = 1 to n_i do

1. Let numAlternates = size (A_{ik})

2. Let position = (freq - 1) mod numAlternates + 1

3. Let route (k) = A_{ik} (position)

4. Let freq = (freq - 1) div numAlternates + 1

{Check if route is feasible}

iv. Let feasible = true

v. For k = 1 to n_i do

1. Let j = Route (k)

2. If not (($STATUS_j$ = Up) and

((($Slot_j$ - size ($Tools_j$) > 0) and ($AvTools_{t_{ik}}$ > 0)) or (t_{ik} ε $Tools_j$))) then

Let feasible = false

vi. if feasible then

1. For k = 1 to n_i do

A. Let j = Route (k)

A. Let newUtilization (j) = Utilization (j) + p_{ik} * o_i

2. Let newVariation = coeffOf Var (newUtilization)

3. if newVariation < testVariation then

A. Let chosenRoute = Route

B. Let testVariation = newVariation

F. If not (chosenRoute = nil) then

i. Add i to the Rel_List

ii. For k = 1 to n_i do

1. Let j = chosenRoute (k); M_{ik} = j

2. Utilization (j) = Utilization (j) + p_{ik} * o_i

3. if (t_{ik} ∉ $tools_j$) then

A. Add t_{ik} to $Tools_j$

B Let $AvTools_{t_{ik}} = AvTools_{t_{ik}} - 1$

4. Return the Rel_List as the list of jobs to be released, and M_{ik} as the list of allocated machines

The basic steps in this heuristic are:

1. Put all the remaining jobs into a list in order of their slacks and priorities.
2. While there is a job in the list
    a. Pick the first job.
    b. For all the possible machine allocations for this job
        A. If the allocation is feasible, calculate the new machine loadings.
        B. If the allocation reduces the coefficient of variation of the machine loadings,
            Select the allocation that reduces the variation the most.
        If it does not reduce the variation, reject the job.

This heuristic is hereafter called Feasibility Release, since it checks for feasibility at every stage.

Heuristic 4. When the FMS under consideration is severely constrained by the transport system, minimizing the transport requirements is more important than balancing the load. A heuristic, developed in this research, to reduce the transportation needs of the released parts is presented below. This differs from the above heuristic (# 3) in that it attempts to minimize the number of transfers; and uses variation of utilizations only as a tie breaker. It assumes the existence of a function transfers, which calculates the number of part transfers needed when given a vector of the machines to be visited.

1. Initialize all the variables with correct values.
2. For i = 1 to N do

A. Let $SLACK = D_i - T - \sum_{k=1}^{op_i} p_{ik} * n_i$

B. Insert i into Pend_List breaking ties in the following order:

i. orders with negative slack always come before orders with positive slack.

ii. orders with higher priority come before orders with lower priority

iii. orders with lesser slack come before orders with higher slack.

3. While there are elements in Pend_List and selectedNum < NUM

A. Let i = First element in Pend_List

B. Remove i from Pend_List

C. Let numRoute = $\prod_{k=1}^{n_i}$ size (A_{ik})

D. Let chosenRoute = nil; testVariation = coeffOfVar (Utilization)

E. For n = 1 to numRoute do

i. Let freq = n; newLoad = 0

ii. Initialize the vector Route.

iii. For k = 1 to n_i do

1. Let numAlternates = size (A_{ik})

2. Let position = (freq - 1) mod numAlternates + 1

3. Let route (k) = A_{ik} (position)

4. Let freq = (freq - 1) div numAlternates + 1

{Check if route is feasible}

iv. Let feasible = true

v. For k = 1 to n_i do

1. Let j = Route (k)

2. If not (($STATUS_j$ = Up) and

((($Slot_j$ - size ($Tools_j$) > 0) and ($AvTools_{t_{ik}}$ > 0)) or (t_{ik} ε $Tools_j$))) then

Let feasible = false

vi. if feasible then

1. For k = 1 to n_i do

A. Let j = Route (k)

A. Let newUtilization (j) = Utilization (j) + p_{ik} * o_i

2. Let newVariation = coeffOf Var (newUtilization)

3. if chosenRoute = nil then

A. Let chosenRoute = Route

B. Let testVariation = newVariation

C. Let numTransfers = transfers (Route)

else

i. Let newTransfers = transfers (Route)

ii. If newTransfers < numTransfers then

A. Let chosenRoute = Route

B. Let testVariation = newVariation

C. Let numTransfers = newTransfers

else if newTransfers = numTransfers then

if newVariation < testVariation then

A. Let chosenRoute = Route

B. Let testVariation = newVariation

C. Let numTransfers = newTransfers

F. If not (chosenRoute = nil) then

i. Add i to the Rel_List

ii. For k = 1 to n_i do

1. Let j = chosenRoute (k); M_{ik} = j

2. Utilization (j) = Utilization (j) + p_{ik} * o_i

3. if (t_{ik} ∉ $tools_j$) then

A. Add t_{ik} to $Tools_j$

B Let $AvTools_{t_{ik}}$ = $AvTools_{t_{ik}}$ - 1

iii. Let selectedNum = selectedNum + o_i

4. Return the Rel_List as the list of jobs to be released, and M_{ik} as the list of allocated machines

The basic steps in this heuristic are given below.

1. Put all the waiting jobs into a list in order of their slacks and priorities

2. While there is a job in the list and there is no congestion

a. Pick the first job.

b. For all the possible machine allocations for this job

A. If any allocation is feasible, select the job.

1. calculate the new machine loadings.

2. Among the feasible allocations to machines, select the one with the least number of transfers. In case of tie,

a. Select the allocation that reduces the variation of machine loading the most.

In what follows, this heuristic is called Transport Release, since its focus is on reducing the transport requirement.

Heuristic 2, 3 or 4 can be applied to the list of waiting orders (unreleased orders). The focus of these is on the variation of workload, compared to the focus of heuristic 1 which is on providing workload close to a target planning period. As such they can be conjectured to provide a more balanced load on the machines. The planning period will, however, change for every pass of this heuristic. In what follows, this approach will be called "variable period release". In this approach, some account of the limited buffers and transportation is taken by limiting the number of work parts admitted to the system at one time to avoid congestion. The allocation of pallets, buffers, and material handling devices to the work parts is handled through queueing and priority rules. The sequencing of the parts through the FMS is also done using priority rules.

Priority Release

A simple method to solve both the release and dispatch problems is to use dispatching rules for both purposes. Then no scheduling period need to be considered. At every opportunity where a part can be released, release one or more parts on the basis of the dispatching rule. This opportunity occurs whenever an order is finished on the FMS since the tools needed specifically for this order are no longer required, and more jobs can be released after retooling the FMS. Obviously, this approach will incur a penalty of setups.

Heuristic 5. The following heuristic was developed in this research for this release method.

1. Initialize all the variables with correct values.

2. For i = 1 to N do

A. Insert i into Pend_List, ordering the items as per the selected priority rule.

3. feasible = true

4. While there are elements in Pend_List and feasible = true

A. Let i = First element in Pend_List

B. Remove i from Pend_List

C. Let numRoute = $\prod_{k=1}^{n_i}$ size (A_{ik})

D. Let chosenRoute = nil; testVariation = coeffOfVar (Utilization)

E. For n = 1 to numRoute do

i. Let freq = n; newLoad = 0

ii. Initialize the vector Route.

iii. For k = 1 to n_i do

1. Let numAlternates = size (A_{ik})

2. Let position = (freq - 1) mod numAlternates + 1

3. Let route (k) = A_{ik} (position)

4. Let freq = (freq - 1) div numAlternates + 1

{Check if route is feasible}

iv. Let feasible = true

v. For k = 1 to n_i do

1. Let j = Route (k)

2. If not (($STATUS_j$ = Up) and

((($Slot_j$ - size ($Tools_j$) > 0) and ($AvTools_{t_{ik}}$ > 0)) or (t_{ik} ε $Tools_j$))) then

Let feasible = false

vi. if feasible then

1. For k = 1 to n_i do

A. Let j = Route (k)

B. Let newUtilization (j) = Utilization (j) + p_{ik} * o_i

2. Let newVariation = coeffOf Var (newUtilization)

3. if chosenRoute = nil then

A. Let chosenRoute = Route

B. Let testVariation = newVariation

else

if newVariation < testVariation then

A. Let chosenRoute = Route

B. Let testVariation = newVariation

F. If (chosenRoute = nil) then

Let feasible = false

else

i. Add i to the Rel_List

ii. For k = 1 to n_i do

1. Let j = chosenRoute (k); M_{ik} = j

2. Utilization (j) = Utilization (j) + p_{ik} * o_i

3. if (t_{ik} $\notin$ $tools_j$) then

A. Add t_{ik} to $Tools_j$

B Let $AvTools_{t_{ik}}$ = $AvTools_{t_{ik}}$ - 1

4. Return the Rel_List as the list of jobs to be released, and M_{ik} as the list of allocated machines

The basic steps in this heuristic are:

1. Put all the remaining jobs into a list, ordering them by the selected priority rule.

2. While there is a job in the list

a. Pick the first job.

b. For all the possible machine allocations for this job

A. If it is not feasible to introduce the job, reject the job and break from the loop

B. If only one allocation is feasible, use the allocation.

C. If more than one allocation is feasible,

select the allocation that reduces the variation the most.

The allocation of pallets, buffers, material handling devices to the work parts in this approach is still handled through queueing and priority rules. The sequencing of the parts through the FMS is also done using priority rules. Table I presents a summary of the heuristics presented above.

TABLE I

SUMMARY OF THE HEURISTICS

Heuristic	Objective	Load limit	Load balancing	Complexity
Heuristic 1	Min overload/ underload	Up to scheduling period	fair	medium
Heuristic 2	Min variation of utilization	Up to preset congestion limit	best	most
Heuristic 3	Min variation of utilization	Up to preset congestion limit	good	medium
Heuristic 4	Min transport, then Min var of utilization	Up to preset congestion limit	fair	medium
Heuristic 5	Introduce jobs, then Min var of utilization	No limit	fair	least

Proposed Methodology

The above sections discussed the hierarchical solution of the random FMS problem, and heuristics that can be used to implement the release phase of the solution. But the problem addressed in this research is even broader than can be addressed by the hierarchical approach. For example, it does not address machine failures and material handling constraints. This section describes the methodology proposed and investigated in this research. The approach is based on rules, heuristics, and simulation. The basic strategy is i) to use rules to handle events as they occur, ii) to use variable period release

as a release mechanism, and iii) to use simulation for selection of the appropriate dispatching policy. These elements of the methodology are now described.

Rules

It is apparent from the study of literature that complex mathematical programming formulations developed for minimizing makespan in the context of static jobshop are not suitable for practical use in the context of dynamic jobshop with random arrival of jobs. Simple priority rules, such as SPT, have been developed for use in the later context. Random FMS is a jobshop with some more complexity added. It can be conjectured that random FMS will also engender suitable rules in the dynamic situation. In addition to the usual jobshop priority rules, rules are needed for the pallets, material handler, etc.

In the proposed methodology, at any point where a control decision is needed, a rule is invoked to make the decision. This rule may be an expert system, or a simple rule such as a priority rule. The rule may call for the execution of an algorithm or the rule may be about other rules to use in the given situation. The decision points and the rules invoked are given below. All the rule bases created in this research are presented in Appendix B.

Releaser. This rule base is invoked at the beginning of a release cycle. At these times, work parts are released to the limit of avoiding congestion. Call this limit point A. When the load on the FMS decreases substantially (but not empty), more jobs are released. Call this lower limit point B. The determination of these two points is not the object of on-line decision making. In a system as complex as the one under consideration, analytical performance modeling cannot (yet) give guidance on the number of work parts to be circulated in order to avoid congestion. Simulation should be used off-line in the planning phase to decide these two points: point A, number of work parts in the FMS

to trigger release, and point B, number of work parts to stop release. Then, in the real time control phase, these numbers are merely used as rules of thumb. At the point when a fresh release is initiated, a small expert system is used to choose the heuristic to use for the release of specific work parts to the FMS. Any of the heuristics mentioned above may be chosen by this rule base. Heuristic 2 is suitable when the FMS is not overly constrained. When the FMS is overly constrained by the tools, and tool magazine capacities, heuristic 2 is computationally expensive, and results in under loading of the FMS. In that case heuristic 3 may be used. Heuristic 4 is suitable when transportation is a bottleneck in the system. Some examples of the Releaser rulebase is given below in Figure 2 on next page.

Priority rule. The rules are selected by prior simulation (see below). These rules are invoked when a resource such as a pallet or a machine is idle. At this decision point, a simple dispatching rule is used for the selection of the work part. The machines in the FMS use a priority rule. But the transport device uses the closest transport/closest part rule: when a transport device is available, pick up the nearest part; when a work part needs transport, use the nearest available transport.

Machine failure or repair. At the time when a machine fails, or when a machine is up after repair, new allocations or new releases may be called for. Rules in the controller program handle these events. These rules call for a search for available alternate machines (suitably tooled) in case of machine failure. If such machines cannot be found, the affected work part is sent back to the load/unload station. During the next release cycle, an alternate machine is tooled for these parts if the routing and tool availability permit. After machine repair, another reallocation of jobs is made and any waiting parts are released.

Order arrival. At the time of arrival of an order, a new release may be initiated if this order is of high priority, and the material is available. A new release is, however, not initiated if the FMS is highly loaded. This decision is made by invoking a rule base.

```
ampleToolsMeansFeasible
        "If there are lots of tools then the FMS is not very constrained"

        if: (ampleTools > 0.8)
                then:
                        [feasible is: true withCertainty: 0.5]
        else:
                        [feasible is: true withCertainty: -0.5]

congestedIfBlocked
        "If the resources in the FMS are blocked, then release fewer
                orders at a time"

        if: ( blocked)
                then:
                        [congestion is: 'high' withCertainty: 0.9.]
                else:
                        [congestion is: 'low' withCertainty: 0.9]

balancedReleaseIfFeasible
        "Where there is ample tool space use balancedRelease because
        it gives more balanced release"

        if: (feasible)
                then:
                        [releaseAlgorithm is: 'balancedRelease' withCertainty: 0.75]
                else:
                        [releaseAlgorithm is: 'feasibilityRelease' withCertainty: 0.75]
```

Figure 2. Example of Releaser Rule Base

Simulation

The complexity of dynamically scheduling and controlling a random FMS can be captured adequately only by a discrete event simulation model. It is conjectured that the only means of examining any decision choice in this environment is to carry out detailed simulation. The methodology of using simulation as a real-time decision making tool is explained by Harmonosky (1990): (See Figure 3, next page, from [Harmonosky, 1990]).

> A computer simulation of a CIM system is linked with the actual physical system. Once the link between the simulation and the system is established, the simulation logic will be controlled by the actual system communication signals, dictating start and stop of robot movement, equipment processing, and cart movement. The simulation will always reflect the current system status, and it will be in effect monitoring the system. Then, when a system production control decision is needed, the starting condition for the simulation is the actual system status. For each different control decision option, a simulation run may be executed for some period of time. The future impact upon the system due to different decisions may be evaluated by analyzing simulation statistical results. In this mode, the simulation model is used as a real-time production control tool with look-ahead system assessment capabilities.

In the proposed methodology, at every epoch of release to the FMS, a fresh selection of priority rule is made. Thus, the priority rule used in the FMS can change from one release cycle to another. A simple rule base (with about 10 rules) selects the dispatching rule to be used. If there is a congestion in the system, SPT is selected. If the load is light, SLACK is used. In these cases, simulation is not required for the selection of priority rule. In other cases, deterministic simulations are carried out for the time period until the next anticipated release cycle using a few rules selected by the rule base. The rule that gives the highest total utilization of the machines is selected for the next release period. A few examples of the rules are given in Figure 4 on page 83.

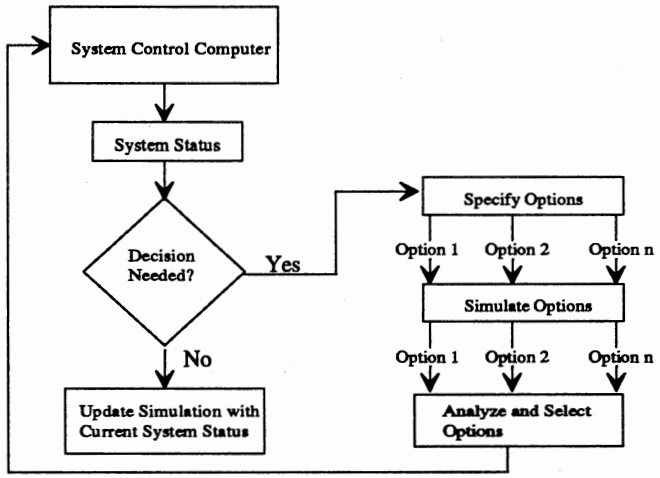

Figure 3. Interfacing Simulation with Physical System for Real-time Control

In the proposed methodology, a rule base is used to select the dispatching rule. If there is a high degree of certainty about the decision rule selected by the rule base, a simulation look ahead may not be necessary. But if a simulation of alternative priority rules is called for, it is done for a period corresponding to the release cycle. The best rule is then used for this cycle.

Wu and Wysk (1989) (Chapter III, page 41) also use deterministic simulation look-ahead to select dispatching rules from a set of dispatching rules selected by an expert system. The difference in the approach in this research from that in Wu and Wysk (1989) is in the criteria followed for the selection of priority rule and in the time horizon for simulation. They suggest using the same criteria for control simulation as for the system modeled. A preliminary study with this approach was not successful in this research. Instead, the criteria of total utilization of the machines was used for the control simulation. Wu and Wysk also use a relatively much shorter time horizon for their simulation. Again, an early test of extending the result of small horizon simulation to longer release cycles was not encouraging and longer, actual time cycles were used in this research.

```
congestedWhenManyJobs
        "Conway study: high congestion indicates use of SPT "

        if: (  (pendingJobs = 'many')  & (lateJobs = 'many'))
                then:
                        [congestion is: 'high' withCertainty: 0.9 ]
                else:
                        [congestion is: 'low' withCertainty: 0.8]

defineNiceAndEasy
        "If the load is light: not too many pending jobs and few or medium released jobs
are late, it is nice and easy"

        if: ((pendingJobs = 'few')  &  (lateJobs = 'few')  )
                then:
                        [niceAndEasy is: true withCertainty: 1.0]
                else:
                        [niceAndEasy is: true withCertainty: -0.6]

dueDateForLateness
        "DueDate is sometimes a good rule for decreasing lateness"

        if: ( measure = 'lateness')
                then:
                        [dispatchingRule is: 'dueDate' withCertainty: 0.6 ]
slackForNoProblem
        "If the load is light: not too many pending jobs and most released jobs are not
late, follow the slack rule"

        if: ((measure = 'lateness')  &  (niceAndEasy) )
                then:
                        [dispatchingRule is: 'slack' withCertainty: 1.0].
```

Figure 4. Example of Dispatcher Rule Base

Summary

This chapter describes the framework that was developed for the realization of objective 1 of this research. The proposed methodology does not bring about a mathematically optimum solution of the problem. It, however, seeks to address the multiple facets of the problem as outlined in the problem statement of this research. (Chapter II, page 7). As an overview, the multiple constraints in a random FMS are handled in the following way:

Material Handling Constraints

When the transportation subsystem is perceived by the Releaser rule base as the bottleneck, the rule base selects heuristic 4 to reduce part transfers. Another significant consideration of the material handling constraint is the priority rule used in the system to reduce idle travel: when a transport device is available, it picks up the work part that is closest to it. Further, in the dispatching rule selection, the dynamic effect of material handling constraints is accounted for in the simulation.

Buffer Capacity

Buffer capacity is taken into account by considering the congestion during release. Jobs are released so that congestion is reduced. Part of the problem is solved by off-line simulation: in determining the range of work part levels in the FMS at which the FMS should operate. Then this range is used by the on-line control system.

Alternate Routing of Parts

Alternate routing of parts is used to allocate the operations to the machines. This is done with a view to improve the machine load balance and/or the transportation requirement, as the situation may warrant.

Tool Transport and Tool Changes

Basically this is a setup question and every setup is allowed setup time in the system. The effect of this constraint is felt in the quantities of work parts allowed in the FMS at any one time.

Due Date and Priority

Each job has a due date based on the total work content. It can have one of two priorities: high and low. Due date is directly considered in release. All the jobs are ordered by their slacks (up to due date) and their priority.

Material Availability

The jobs are not scheduled all in advance. The jobs are only scheduled for a small time window. So material availability is not a problem. A job is not released if its material is not available.

Fixtures and Pallets Availability

A simple rule is used during release to take into account the availability of pallets. Work parts are only released that require up to a factor of the number of pallets available. This factor needs to be determined, again, by off-line simulation.

Tool Availability and Tool Life

Tool availability is directly considered in the release heuristics. An order is released only if the tool is available for the order. The tool life is considered in the simulation and time for tool change is allowed after tool life is expired.

Machine Failures

Machine failures and machine repairs invoke heuristic rules. Upon failure, all the parts needing that machine are routed to the load/unload station. From there they are routed to an alternate machine in the next release if an alternate is available. Upon completion of repair, a rule base is invoked to determine if a rescheduling (re-release, and selection of dispatching rule) is called for.

Dynamic Production Environment

The dynamic changes occurring in the FMS are always updated in the simulation environment. The continual arrival of orders, failure and repair of machines and other dynamic events cause the corresponding control/decision rule to be invoked. Thus, the dynamic production environment is accounted for.

On-Line Scheduling and Control Decisions

The above should be implementable on-line. The reason is that the heuristics are not complex, and are quickly executed. Further, the time window for scheduling is brief. Since the dynamic production environment changes continuously, it is not fruitful to create detailed schedules far into the future. Therefore, projection is made only for a short duration at a time. At that time, another release decision is made and further projection is made again for a short duration. The simulations for selection of dispatching rule are carried out only for a time window until the next release. Further, on-line interactive decision making can be carried out by the user by selecting the orders to be released. The program will determine if the release is feasible. If feasible, a dispatching rule may be selected and a time window established. The program can carry out a simulation with this input and report on the results.

With the development of the methodology as described above, research objective 1 has been achieved successfully.

CHAPTER VI

OBJECT ORIENTED REPRESENTATION OF THE PROPOSED FRAMEWORK

Introduction

It is apparent from the description of the proposed methodology (Chapter V) that a prerequisite of computer implementation of the methodology is a good environment for modeling of complex systems and decisions therein. The rationale for creating an object oriented programming environment for scheduling and controlling FMS is the superior capability of this paradigm to represent complex systems. Since the methodology proposed herein uses discrete event simulation as a monitoring and controlling tool for the flexible manufacturing system, the simulation modeling capabilities of the OOP paradigm are especially attractive. Investigations into object-oriented modeling over a number of years at the Center for Computer Integrated Manufacturing at OSU have demonstrated the attractiveness of OOP for the purposes mentioned above [Beaumariage (1990), Karacal (1990)].

Features of Object Oriented Programming

The basic construct in OOP is the object. An object may represent a physical object, an information object, or a decision making entity in a complex system. The attributes of the object are stored in instance variables. These instance variables are, again, objects representing some aspect of the object. These objects can be created as an instance of some specific class of objects. Various classes of objects are predefined as needed. Operations to be carried out on a class of objects are defined as methods for that

class. These methods are privy to the class for which they are defined. Only the methods defined in this way can alter the state of an object. Thus, these methods are safe to use (do not cause unexpected results) since they have been previously defined with the particular class of objects in mind. Any modeling of a complex system consists of creating instances of the predefined classes. These objects represent, for instance, the FMS controller, the NC machine, the transport device, etc. The interaction between the objects takes place through message passing between the objects which causes the appropriate method defined for the objects to be invoked. This causes alteration of the state of the object and/or carrying out the desired computation.

The main feature of the OOP paradigm is inheritance. It means that once a class of objects is defined, a subclass of it can be defined that hierarchically inherits all the instance variables and methods of the superclass(es). Thus, a class hierarchy can be created with the highest class representing the most general attributes of the family of objects, and the lower classes defining more and more specific types of objects. Inheritance increases the effectiveness of the modeler by providing for the reusability of the predefined classes. Once a class is defined, if a modeler needs a similar class he can make the new class a subclass of the old class, thus using all the code previously written for the old class. Only the specific differences between the new classes need to be redefined.

Another advantage of using OOP, briefly alluded to before, is encapsulation and consequent message passing. The instance variables and the associated methods defined for a class of objects are the sole responsibility of the class: the instance variables and methods are encapsulated. No other object or method can access the state of an object without passing an appropriate message to the concerned object. This makes for a modular environment where the modeler needs to be concerned only with one identifiable entity in the system and the changes in the state of the entity. Still another productivity enhancing feature of OOP is polymorphism, which means that the same message passed

to different objects may cause entirely different effects. Since the methods are encapsulated within the object, the commonality of method names does not cause any ambivalence. The method will make the object behave in a way that is appropriate to the object receiving the message.

Proposed Framework

The proposed framework uses rules and simulation to make control decisions for the operation of a random flexible manufacturing system. For a complete description of the proposed methodology, reference may be made to Chapter V. A brief review is presented here. The system supervisor computer maintains the current state of the FMS in a simulation model. As the host computer and the NC machines send in information, the simulation model is updated. This process is illustrated in Figure 5, in the next page. These external events may merely signal the updating of the simulation model. In those cases no control decisions are needed. When the supervisor decides that some action is needed, it invokes the appropriate rules. These rules may require carrying out some algorithm, or simple prioritizing of some queue. They may require evaluation of control policies through execution of a simulation model. In this latter case, a simulation model is created from the current simulation model, and the simulation is carried out. The appropriate policy is then chosen from the simulation output, and conveyed to the NC machines.

Object Oriented Representation of Proposed Framework

The object classes needed for representation of the framework (and the FMS) have been coded in Smalltalk-80, which is a pure object oriented language [Goldberg and Robson, 1989]. A rather large object library and some elementary classes needed for simulation are provided with the software. A considerable amount of additional development was required for the representation of the system being investigated.

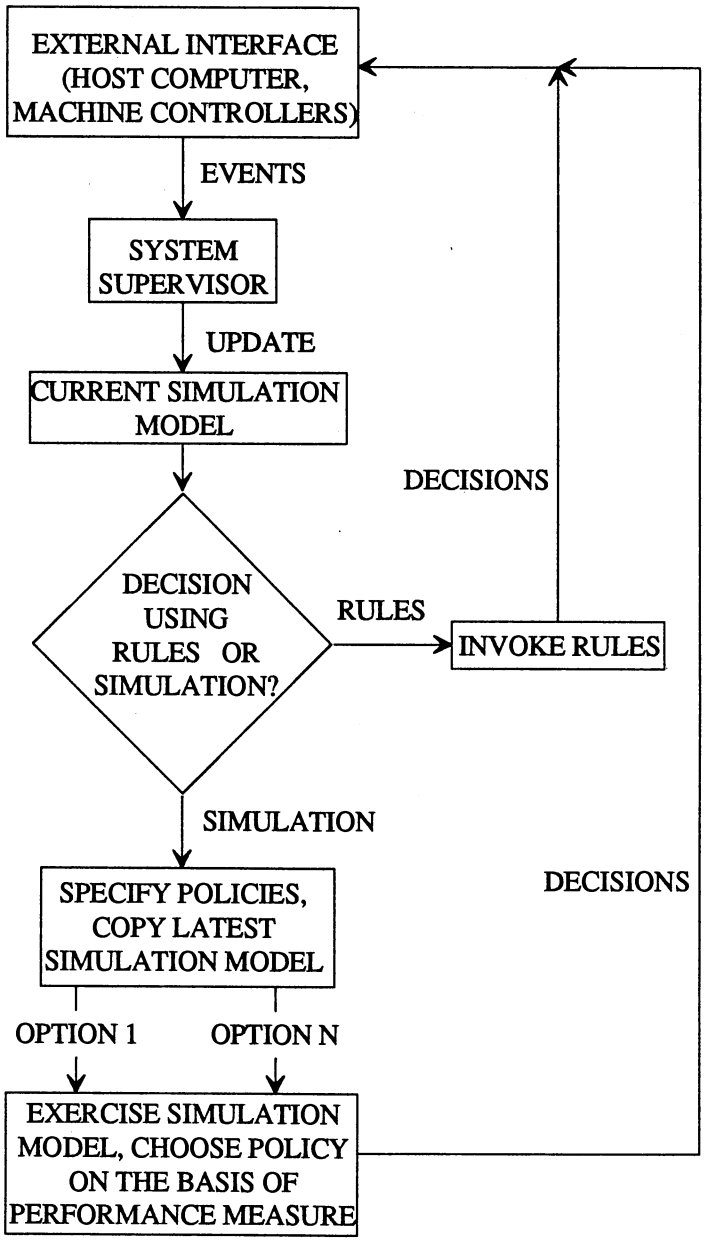

Figure 5. Framework for the Proposed Methodology

The object oriented representation parallels the communication aspects of the real system. The machines and the transport devices "talk" to the controller via message passing. The controller, in turn, decides on the control response and communicates the decision to the machine. In a real world implementation of the methodology, the message passing of OOP could be left intact, and the messages could then be passed through the real communication interfaces.

The main class hierarchies developed are:

1) Resource. These objects represent the physical equipment and fixtures used by the system. By using subclasses, all the different types are represented with the minimum duplication of code.

2) FMSController. These objects represent the decision/control aspects of the system. Again subclasses represent different control policies that may be used. One instance of the controller is used at a time.

3) WorkFlowItem. This object is a representation of the parts flowing inside the FMS. It also represents the computer process for the part.

4) Buffer. Buffers represent the locations of work in process. They may be attached to the machines or may be free standing. Subclassing permits representation of limited capacities.

5) Operation. This is a data structure representing one process of the work part.

6) Routing. This object is a collection of operations needed to complete a work part. A subclass permits representation of alternate routings.

7) Order. This class represents incoming shop orders complete with order size, priority, availability, etc.

8) Tool. Objects of this class represent the tools to be mounted on the FMS machines.

9) ToolCrib. This object is a repository of the tools.

10) TimeMatrix. This is an information object for look up of the travel distances for the transport devices.

11) Simulation. This hierarchy of classes performs the important function of simulation processing - handling of the event queue, pausing and resuming of the computer processes, etc.

In addition to the hierarchies mentioned above, rule bases were implemented, also in Smalltalk. HUMBLE (Piersol, 1987), a commercial expert system shell for Smalltalk-80, was used for this purpose. The rule bases are:

1) Releaser. This rule base decides on the heuristic to be used for the next release cycle. It looks at the congestion in the system, severity of tool constraints, and transport requirements to select from heuristic 2, 3 and 4 (Chapter V). Currently 10 rules are implemented.

2) Dispatcher. This rule base selects the priority rule to be used in the next release cycle. Currently 8 rules are implemented.

3) OrderArrival. These rules decide if a reschedule is to be initiated at the time of arrival of an order. There are 7 rules in this rule base.

Some examples of these rules were presented in Chapter V. Listings of these rules are given in Appendix B.

The following contains a very brief description of the class hierarchies, their instance variables, and methods. This description should facilitate perusal of the complete code attached in Appendix B.

Resources Hierarchy

The primary objects in any dynamic discrete event system are the resources. OOP provides an excellent medium for representing different types of resources in a hierarchical manner. The hierarchy developed to represent resources in a random FMS is shown in Figure 6, next page.

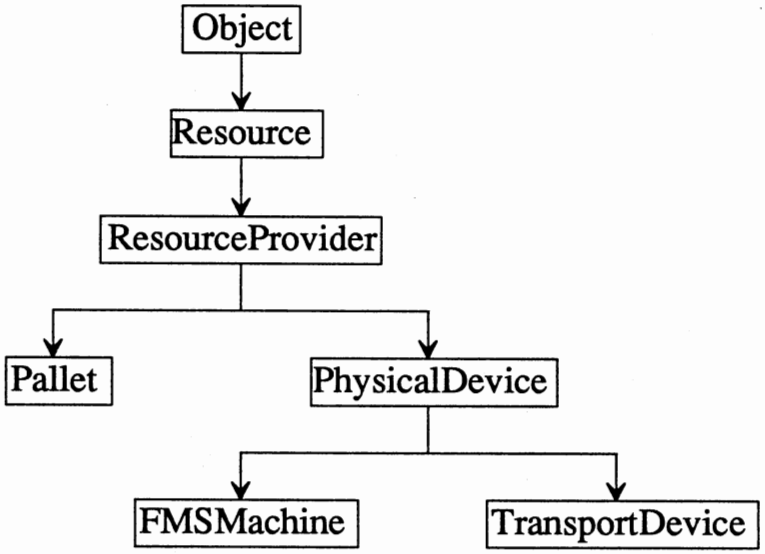

Figure 6. Hierarchy of Resource Classes in Random FMS

Resource and ResourceProvider are basic Smalltalk-80 object classes that provide the function of entities waiting for units of a server, getting the service, and releasing the server (s). Resource has an instance variable of pending, which is the queue of entities waiting for service. It also has an instance variable of resourceName, which separates the identity of one resource from another. ResourceProvider maintains the number of available servers in an instance variable of amountAvailable. Every work part in an FMS queues up for a pallet of the requisite type. There are many different pallet types, and a number of pallets of each type are available. Thus Pallet is not much different from a ResourceProvider . The few differences from ResourceProvider are taken care of by making the Pallet a subclass of ResourceProvider and defining new methods or modifying methods of the superclass.

PhysicalDevice is an abstract class. It does not have any instantiations, but forms an umbrella for defining common methods for the classes FMSMachine and TransportDevice. PhysicalDevice includes the instance variables of timeToFailure and timeToRe-

pair, which are the probability distributions for the time between failures and time taken for a repair. It also has the instance variable of status, which indicates whether it is up or down. These variables are inherited by FMSMachine and TransportDevice. FMSMachine also has the instance variable of blocked, which indicates its status regarding blocking by a work part that has nowhere to go, and tools, which represents the set of tools mounted on the machine. TransportDevice includes the instance variable of availableLocations, which represents the locations where available transport devices are located.

Summary of Functions of Resources Hierarchy

Resource.

- Identify the resource by name
- Maintain a queue of requests for service

ResourceProvider.

- Quantify the number of servers available
- Provide service to the customer on a FIFO basis, maintaining count of available number of servers. This discipline can be overridden in lower level subclasses.

Pallet.

- Use the Queue object to provide statistics on time in queue and queue length
- Use the FMSController object to allocate pallets to work parts when available

PhysicalDevice.

- Provide failure and repair mechanism. When failure occurs, one unit of the resource is withdrawn from service, and after the passage of repair time, it is returned to service.

- Communicate with the FMSController object to decide which work part to process next from its input queue
- Inform the FMSController when the failure or repair completion event occurs.
- Adjust (increase) the remaining processing time of work parts when failure occurs
- Keep track of process times and utilization statistics

FMSMachine.

- Maintain tool magazine as a capacitated set of tools, provide methods for the interchange of tools
- Interact with tool crib
- Use input and output buffers of limited capacity if initialized for that purpose
- Provide mechanism for blocking of the FMS when there is no place for a finished job to go, i.e., the output buffer is full
- Make work parts wait not only for the servers, but also for the input buffer space when the input buffer is limited in capacity
- To avoid deadlock, use a reservation mechanism when a work part proceeds to the FMSMachine after waiting for a space in the input buffer
- Provide for failure conditions when work part may be reassigned to another machine

TransportDevice.

- Track all the transport devices in the FMS as to their locations
- Provide service to transport requests as per the priority rule
- Provide for aborting of transport services when work parts are reassigned to a different machine

FMSController Hierarchy

The scheduling and controlling functions within the FMS are represented by subclasses of FMSController. This is an abstract class without any instantiation, and serves to introduce common methods for all controllers. Its instance variables are fms, representing the FMS controlled; controlSim, the simulation process for choosing the priority rules; and numberOfActiveWFI, the number of work parts released into the FMS. The hierarchy of its subclasses is shown in Figure 7, next page.

FixedPeriodController class has the bulk of the code for this hierarchy. It releases work parts into the FMS based on minimizing an objective of underload/overload in a given scheduling period. A list of route allocations of work parts is maintained as allocatedWFI. The list of allocation of tools to the machines in the FMS is stored in the instance variable toolAllocations. The objective function is stored by objective, and resourceUtilization is the list of machine loadings. The list of active machines in the FMS is updated in resources, while toolCopiesUsed is a dictionary of the number of copies of tools used indexed by tool types. The instance variable dispatchPolicy stores the current priority rule used in controlling the FMS.

VariablePeriodController releases work parts into the FMS based on rules, and a simulation for selecting the priority rule. Its instance variables are palletUtilization: a dictionary of pallet loadings, timeInTransport: a list of time spent by work parts in transport activities, timeInSystem: sojourn time of work parts inside the FMS, pendingOrders: a prioritized list of unreleased orders, and congestionFactor: a factor used to determine the number of work parts released into the FMS to avoid congestion. PriorityRuleController releases and dispatches work parts on the basis of one assigned priority rule: the dispatchPolicy. SimController is a dummy controller used for the look ahead simulation of priority rules.

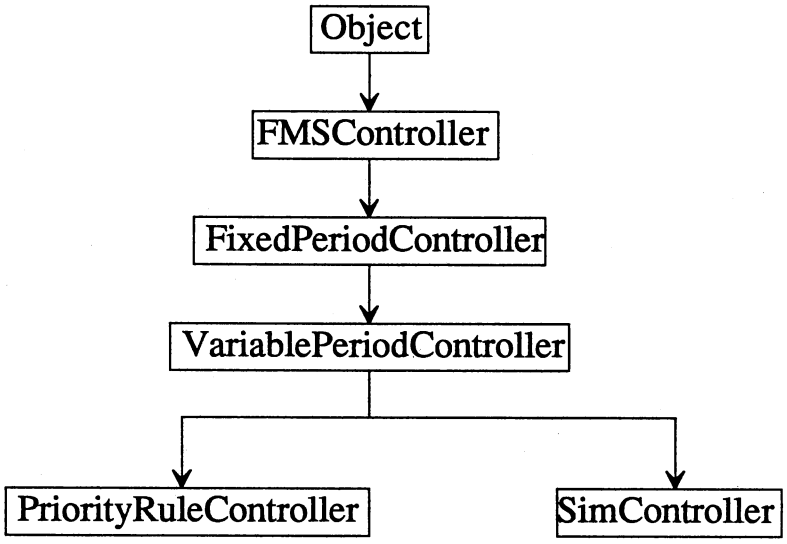

Figure 7. Hierarchy of the FMSController Subclasses

Summary of Functions of the FMSController Hierarchy

FMSController.

- Provide linkage to the FMS that it controls
- Act as an umbrella for its subclasses

FixedPeriodController.

- Implement Heuristic 1 (Chapter VI) attempting to minimize the over-load/underload for all machines
- Check for feasibility for a particular machine assignment, or a routing assignment: test if the machine is up and whether the appropriate tools are available and there is place in the tool magazine
- Schedule tool changes for the machines after a release cycle is completed
- Keep track of tool allocations and tool availability

- Handle the event requests of order arrivals, machine failure or repair, and order completions
- Choose job from an input queue following the selected priority rule
- Find, if possible, an alternate machine for a work flow item when the assigned machine has failed

VariablePeriodController.

- Implement Heuristic 2, 3 and 4 for release of work parts into the system
- Create a simulation model when a control simulation is needed, and run it. Then select the appropriate rule for use
- For event handling, when it is necessary to interface with rule bases, initialize HUMBLE knowledge bases, and provide interface to them
- Provide its own version of event handling, some of which is different from the FixedPeriodController
- Prioritize pending orders

PriorityRuleController.

- Implement different aspects of event handling
- Implement release of work parts into the system based on specified priority rule

SimController.

- Implement its own event handling
- Provide means of copying tool allocations, and routing allocations for use in the control simulation

WorkFlowItem Hierarchy

All work parts are represented by objects of the class WorkFlowItem. This class has the dual function of representing both the physical part and the process associated

with the part. Some of its instance variables: routing, name, entryTime, location, status, resourceWanted, amountWanted, priority and pallet refer to the physical work part. Other instance variables: myProcess, mySemaphore, and resumptionTime are attributes of the computer process. The class hierarchy of this object is shown in Figure 8.

Figure 8. Hierarchy for the WorkFlowItem

Summary of Functions of WorkFlowItem Hierarchy

SimulationObject (Smalltalk-80).

- Functions as the entity that "flows" through the system. Each instance creates a process, and carries out its "tasks"
- The basic task of a simulation object is to wait for a resource, acquire a resource, and to release a resource

WorkFlowItem.

- Store an identification of its own process so that the process may be suspended, resumed, or terminated as necessary
- Answer to various queries regarding its state, or its attributes
- Carry out its task - repeatedly enquire the controller about its next process, get itself transported there, and carry out the process at the location

- Provide means to stop its ongoing process and to restart another process for itself, which is necessary when there is a change in machine allocation, or in the tasks for the work part
- Wait for resources like pallets, machines, and transport devices, and release them when done
- Re-create its own process for control simulation

Buffer Hierarchy

The representation of buffers where parts wait in an FMS is shown hierarchically in Figure 9. The basic buffering function is represented by an object of the class Queue. It has the instance variable queue, which represents a collection of the work parts waiting at the buffer. The queue discipline is, however, determined by the controller object.

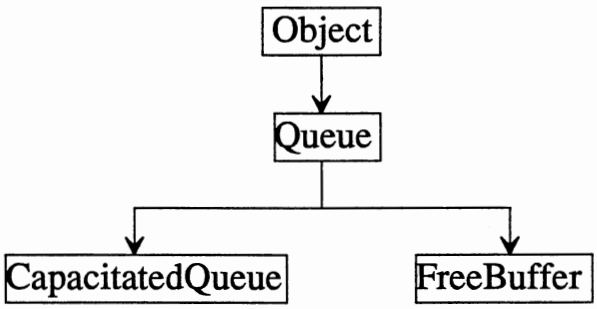

Figure 9. Classes to Represent Buffers

Addition of an instance variable of capacity changes the class Queue to CapacitatedQueue and permits it to represent finite queues. These two objects represent the input or output queues of machines. InputQueue and OutputQueue instance variables in

FMSMachine are instances of CapacitatedQueue. A free standing buffer is represented by a FreeBuffer, which has a name attached to it. A load/unload station is represented by a FreeBuffer.

Summary of Functions for the Buffer Hierarchy

Queue.

- Provide a list of work parts waiting at the queue
- Collect time in queue and queue length statistics
- Provide methods for enumerating over the queue's items
- Provide answers to queries about items in the queue

CapacitatedQueue.

- Impose a limit on the number of parts that the queue can hold
- Keep trace of the parts for which space has been allocated at this buffer
- Answer questions relating to remaining queue capacity

FreeBuffer.

- Identify individual buffers by name

Operation class

The class Operation represents an individual process on a work part. It has the instance variable of machine, indicating the machine required for the operation; tool, the tool to be mounted for the operation; processTime, the time required for the operation; and setupTime. Since alternate operations are represented by a linked list, this class is a subclass of Link. Its class hierarchy is depicted in Figure 10, in the next page.

Figure 10. Hierarchy of Class Operation

Summary of Functions for Operation Class.

- Maintain a list of the machine, tool, process time, and setup time for any (alternate) operation, and provide access when needed
- Answer queries about any of its instance variables

Routing Hierarchy

The routing to be followed by a work part is represented by a class hierarchy as shown in Figure 11.

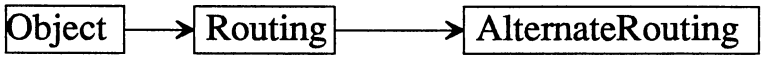

Figure 11. Class Hierarchy of the AlternateRouting Object

An object of class Routing has instance variables of length, denoting the number of serial operations to be carried out; routing, which is a list of operations to be carried out; and currentStage, pointing to the current stage of the processing of the work part. AlternateRouting is a subclass of Routing, and each element of its routing is a linked list of the alternate operations.

Summary of Functions of Routing Hierarchy

Routing.

- Information object to hold the information on the list of operations needed for a work part
- Track which stage of the operations is currently pending
- Enumerate over all the operations in its different stages
- Alteration and creation of its structure
- Check if any failed resources are included in any of its stages
- Answer the number of work part transfers necessary to accomplish the routing

AlternateRouting.

- Store alternate operations for each stage in a linked list. Add alternate operation for a stage, as well as add total number of stages of operations
- Provide all possible routings, and permissible current machines for any stage

Order Class

An object of the class Order represents incoming shop orders. It maintains a list of its associated work parts in the instance variable wfiCollection. Other significant instance variables are: dueDate, totalWork, orderSize, priority, routing, and pallet.

Summary of Functions of Order Class.

- Keep track of the availability and release status of the order
- Answer queries regarding different aspects of the order
- Keep the identity of the group of work parts forming an order

Tool Class

Objects of the class Tool represent tools used in the FMS. Its instance variables are: numCopies representing the number of copies of this tool, toolLife representing the useful life of a copy of the tool, location pointing to the machine on which a particular copy of the tool is mounted, name to keep track of its identity, and toolCrib representing the location of the tool inventory.

Summary of Functions of Tool Class.

- Keep track of movement and location of all copies of itself
- Keep track of life of each copy of a tool, and schedule a tool replacement when tool life is expired

ToolCrib Class

Class ToolCrib represents the tool crib in the FMS with instance variables: changeOverTime, which is the setup time required for tool reallocations; toolCollection, representing the list of tools in the FMS; and totalCopies representing the master list of copies of all tool types.

Summary of Functions of ToolCrib Class.

- Work as a repository of all the tools
- Provide information to the controller about availability of tools

TimeMatrix Class

An object of the class TimeMatrix stores the time required by the transport device to travel from one station to another. Its instance variables are: resourceArray, the list of the station names; size, the size of the matrix; and the timeArray, which is the matrix of travel times.

Summary of Functions of TimeMatrix Class.

- Work as an information object holding travel times from/to all the stations in the FMS

Simulation Processing Hierarchy

The actual processing of the simulation is carried out by the CimSimulation object. The hierarchy of its superclasses is shown in Figure 12.

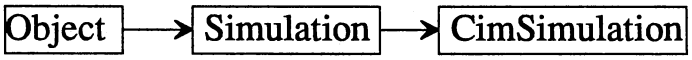

Figure 12. Simulation Processing Classes

The Simulation class is provided by Smalltalk-80. It provides its basic functions of simulation processing through the instance variables resources, currentTime, and eventQueue. A list of the machines in the FMS is maintained by the resources instance variable. All simulation in Smalltalk-80 is done through multiple processes. When a process cannot proceed, it is suspended and its resumption event joins the eventQueue. The class CimSimulation maintains additional lists of pallets, and transportDevices. The orderList is a list of active orders. Other lists are lists of unreleased work part entities: arrivedWFIs, list of released entities: releasedWFIs, list of entities waiting for failed machines: failedMachineWaiters.

Summary of Functions of Simulation Processing Hierarchy

Simulation (Smalltalk-80).

- Provide the event queue necessary for the conduct of discrete event simulation
- Keep track of active and inactive processes; and suspend and resume them as called for by the event triggering
- Creation of new processes and scheduling them in the event queue
- Maintain relevant information about the resources involved in the simulation

CimSimulation.

- Scheduling mechanism for ending the simulation arbitrarily by any process
- Keep track of active and released work parts, arrived orders, pallets, and transport devices
- Direct manipulation of the event queue such as deletion of a scheduled event, or alteration of time for a particular event

Interactions Between the Objects

This section describes the interactions (message passing) between the objects in fulfillment of research goal 3. Since the physical objects are represented by software objects in the implementation, the messages sent by the software objects to each other are very similar to the messages that would be sent between the actual objects. For the purpose of easy access, the methods for each object are categorized into groups, each group addressing one behavior of the object in question. These groups are called protocols. The detailed implementation is presented in the form of actual code in Appendix B. Here, only the more important protocols and their messages are discussed. Figure 13 on the next page shows a summary of the class hierarchies developed to implement the methodology.

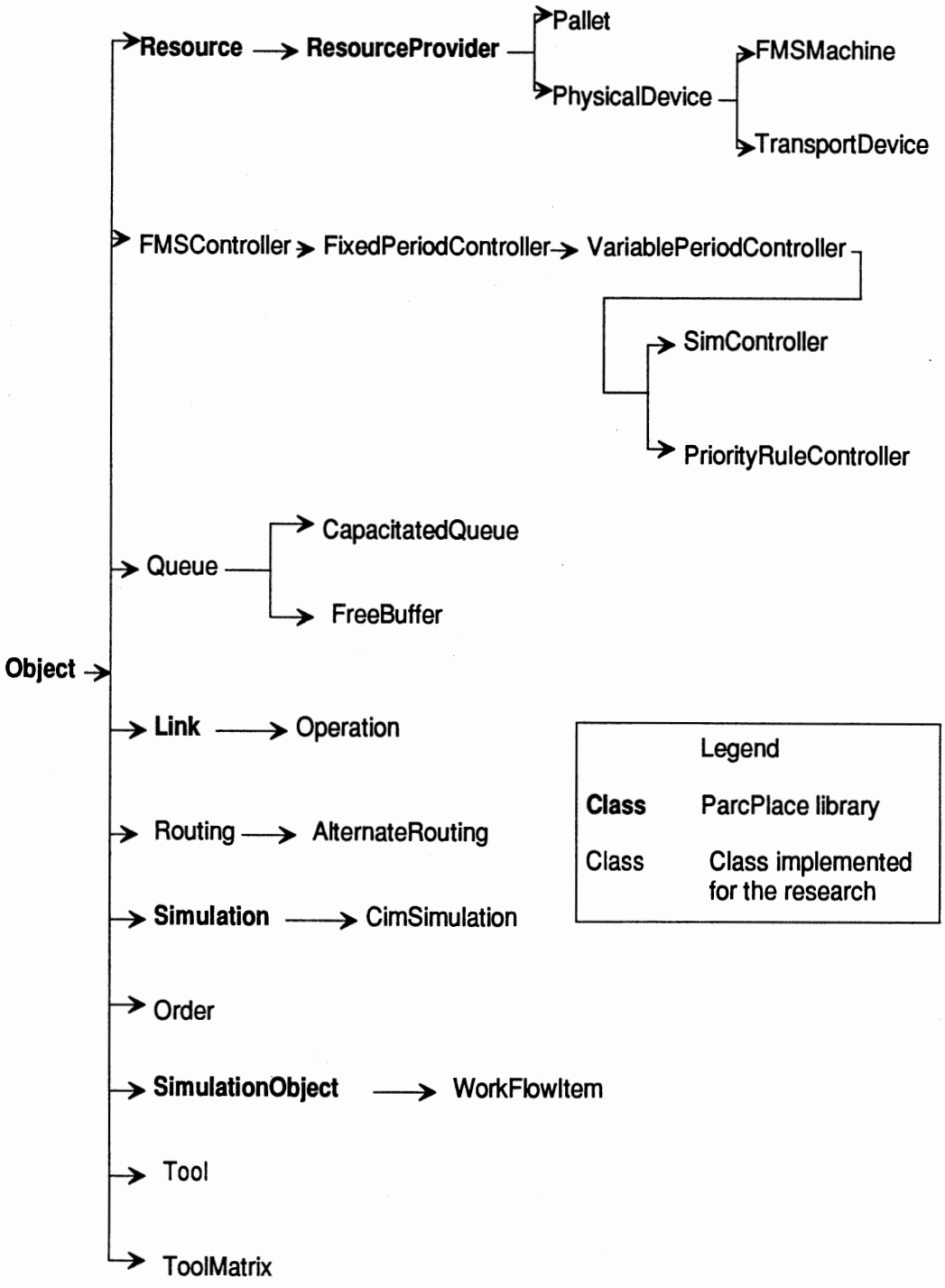

Figure 13. Summary of the Class Hierarchies

The major interactions between the objects are the message passings that occur between the resources and the FMSController. These mainly consist of event updates to the controller, and the controller's response to the events. The protocol in the FMSController hierarchies to handle events is called event requests. The methods in this protocol differ between the controllers to reflect the control policies of the particular controllers. Where possible, advantage is taken of the inheritance feature of object oriented programming.

Controller Hierarchy

The major methods in the event requests protocol are:

1) doneOrder: anOrder. An order has been completed. The FixedPeriodController ignores the message. VariablePeriodController schedules a new release of parts into the FMS, if the number of work parts inside the FMS has dropped below the minimum point. PriorityRuleController attempts to release more work parts into the system if feasible.

2) iAmDone: aWFI. A work part has completed its journey through the FMS. FixedPeriodController schedules a new release if there are no more work parts in the FMS. VariablePeriodController collects statistics about the departing part for use in its decision making process. PriorityRuleController attempts to release more work parts into the system if feasible.

3) orderArrival: anOrder. An order has arrived at the FMS. VariablePeriodController invokes the OrderArrival knowledge base to decide if a release cycle of the FMS is called for. Other controllers ignore this message except in the case when there is no work part inside the FMS, in which case, they initiate a release cycle.

4) failedDevice: aResource. A resource has gone down. All the controllers inherit the same behavior for this event, from the FixedPeriodController class. The controllers send the work parts being processed, if any, and parts whose next process-

ing was scheduled at this resource back to the load/unload station. An attempt is made to reallocate these parts to another machine if the machine is appropriately tooled.

5) upDevice: aResource. A downed machine has been rendered serviceable. All the controllers exhibit the same behavior for this event. A release cycle of the FMS is initiated in which all the parts waiting for this failed machine are routed to this machine.

The main computations in the controller hierarchy are carried out by code in the task language protocol. The major methods in this protocol are:

1) reallocate. This method initiates the release cycle. It uses various methods to carry out its functions. On the whole, the tool allocations necessary for the work parts currently inside the FMS are calculated first. Then, the appropriate algorithms are carried out to determine new work parts to be released into the environment. Any necessary simulation runs and/or rule base consultations are carried out.

2) releaseWaitingJobs. This method is used by the FixedPeriodController to implement Algorithm 1 (Chapter V). VariablePeriodController consults the Releaser rule base to determine the algorithm to be used. It then uses the appropriate method: balancedRelease implements Algorithm 2, feasibilityRelease carries out Algorithm 3, and minTransportRelease incorporates Algorithm 4. This controller also determines the appropriate priority rule using Dispatcher rule base. If there are more than one potential candidates for the priority rule, simulation models are created and run to select the rule for the current simulation cycle. PriorityRuleController has its own version of this method. It implements the priority rule under which it is operating.

3) scheduleToolChange. This method is carried out after the initiation of the release cycle. It finds out the earliest time when a tool change can be carried out:

when all the current operations are finished. It then passes the message to all the machines to carry out the tool change necessitated by the latest order allocations.

Resource Hierarchy

Objects of the Resource class hierarchy provide service to all the work parts. They get instructions from an object of the controller hierarchy. They send event requests to the controller hierarchy and requests for decisions including which work to process next. The major methods in this hierarchy have to do with handling of the objects in the queue and processing of the work parts. These methods are defined in the task language protocol.

1) provideServices. This is the work horse method defined for all the subclasses as appropriate. This method is used whenever the status of resource availability or service demand changes. If there is enough resource at hand, it sends a message to the controller to find which work part to process first.
2) reserveSpace. This method is sent to the FMSMachine when a work part needs to be sent there. The work part waits at its previous location until there is a space available at the machine. This provision is necessitated by the limited input queue capacity of the machines: to prevent deadlock in the event when the transporter is waiting for a place in the input buffer of the machine and the machine is waiting for the transporter to offload a work part from its buffer.
3) hasSpace. This message sent to a FMSMachine returns the availability of space in the machine input buffer.
4) isBlocked. This message sent to a FMSMachine is answered with the status of the machine with regard to blocking.
5) scheduleToolChange: toolSet at: aTime. This message tells a FMSMachine to change the tools in its tool magazine with the new tool set at the requested time.

This will allow for the time required to swap tools with the tool crib and with other machines.

6) continuePending. This message is sent to a FMSMachine after it has recovered from a failure. It causes the machine to start processing any parts waiting in the input queue or parts with partially finished operation.

7) amountAvailable. A PhysicalDevice answers with the number of available servers when this message is sent to it.

8) currentWork. This message is implemented at the PhysicalDevice level and answers the work parts currently processed by all the servers.

9) transport: aWFI to: aResource. This message sent to the transport device causes the transport to pick up the work part and transport it to the resource. In so doing, the actual server is determined by the controller in response to a message.

10) isUp. This is another inquiring message, and any subclass of PhysicalDevice answers with its status, true when it is up and false when it is down.

Tool Class

Objects of the Tool class represent the tools in the implementation. Each tool is an instance of this class. Its important messages are in the tools transaction protocol.

1) supplyToolFor: aMachine. This message to a tool causes it to send a copy to the machine, and to do the necessary book keeping about the location of the copy.

2) usedTime: aTime location: aLocation. This message informs the Tool that the copy at the location has been used for a given time. This goes towards book keeping of the used life of the tool copy.

3) youAreReturnedFrom: aMachine. The Tool is being told that the copy previously at a machine is now returned to the tool crib.

Routing Hierarchy

The routings are represented by the Routing and AlternateRouting classes. Methods are defined for the construction of routing and for the accessing of the information about the routing. Important methods are located in the accessing protocol.

1) numberOfMovements. This message, answered by Routing , informs about the number of movements needed if this routing is followed.
2) numRemainingOps. This method answers the number of remaining operations in the current routing including the current stage of operation.
3) allPossibleRoutes. An AlternateRouting answers this message with all the possible combinations of routings that can be followed by a work part.
4) currentProcesses. This message is implemented at the AlternateRouting level, and returns a list of the machines to which a work part could go for the current operation.

Simulation Hierarchy

Objects of the Simulation hierarchy exhibit all the behaviors necessary for carrying out and updating a simulation model. The implementation uses the subclass CimSimulation for these functions. Most important simulation processing methods are placed in the simulation control protocol.

1) activate. This method tells a simulation model to activate itself and to carry it out. Thus the control simulation model, once created, is sent this message to determine the performance.
2) changeTimeForWFI: aWFI to: aNewTime. This message informs CimSimulation to reschedule the event for a work part to a new time and to re-sort the event queue.

3) holdObject: aWFI for: aTime. A CimSimulation is told by this message to suspend the process for a work part for the given time. This is the basic operation for placing an event in the event queue.

4) scheduleEndAfter: aTime. This message causes CimSimulation to schedule the end of simulation after the passage of a given time.

5) removeFromEventQueueIfPresent: aWFI. This is another simulation processing message that causes CimSimulation to remove a work part from its event queue.

Many methods are implemented for CimSimulation class to keep track of essential book keeping. These methods are mostly in the accessing protocol.

1) activeList. This message when sent to CimSimulation causes it to return a list of work parts that have arrived at the FMS but have not completed the operation.

2) activeResources. CimSimulation responds with a list of resources that are up when this method is invoked.

3) failedMachineWaiters. Again, this method results in a list of work parts that have been waiting for some failed machine.

Queue Hierarchy

The objects of this hierarchy perform the purpose of storing work parts during processing. Usually they are attached to a machine as input or output buffers. But they could be free standing, for example the load/unload station. They could also be a logical queue, as for the transport device. As in other classes, numerous methods permit interaction with other objects. The essential functions of adding and removing work parts are carried out by methods in adding and removing protocols.

1) add: aJob. This method causes a work part to be added to the buffer. The class CapacitatedQueue implements a variation, taking care of the reservation of spaces.

2) remove: aJob. This method removes a job from the queue. Queue statistics are updated.

Methods to enquire about the buffers are provided in the testing protocol. Some examples are:

1) includes: aWFI. This message asks the buffer if it has a given work part. Answer true if it has the work part, false otherwise.

2) isEmpty. This method tests the buffer to see if it is empty.

3) hasSpace. An enquiry is made to see if there is space in the buffer. CapacitatedQueue has to check its actual contents, as well as allocated space. Other buffers, of course, always have space available.

WorkFlowItem Class

Objects of this class represent the work parts flowing in the FMS. For each such object, there is an ongoing process in the computer. As such, there are methods to handle the physical object as well as the computer process. The methods to handle the process aspect are placed in the simulation control protocol.

1) holdFor: aTime. This message causes the WorkFlowItem to stop its process and to wait in the event queue for a given time before proceeding further.

2) pause. This method tells the process to go to sleep temporarily.

3) resume. The WorkFlowItem is told to resume its process after having gone to sleep.

4) terminate. The associated process is permanently terminated. This happens when a work part has finished all its processing, or if a process needs to be restarted after a certain point, as when a machine fails. Then, a new process is started from this point.

The most important methods for this class are contained in the task language protocol. This is where the processes to be followed by a work part are to be found.

1) tasks. This method is a statement of the tasks to be followed by the work part. When a work part is told to carry out its tasks, it follows its routing, and gets processed and transported until it is done.

2) completeProcessesAtLocation. A WorkFlowItem has arrived at a resource. It has grabbed the resource. It is then told to complete all the processes that need to be carried out at this resource.

3) requestOutputQueue. A WorkFlowItem needs to obtain space at the output buffer of a machine after its processes at the machine are completed. If there is no space there, the machine is blocked. To prevent deadlock where parts in a group of machines are waiting in a circular fashion for the input queue space and thus are causing permanent blocking, a work part is sent to the load/unload station if this situation is suspected.

4) rerouteToStation: aResource. The current machine has failed, and this WorkFlowItem is told to re-route to another machine. This method will cause the WorkFlowItem to terminate its current process and start another process for itself.

5) restartFromLoadStation. This is another method to handle a machine failure. This time the WorkFlowItem is told to go to the load/unload station. The current process is terminated, and another process is initiated.

Summary

In accordance with the object oriented programming concepts, software objects that are 'natural' representations of the physical, decision/control, and information aspects of the random FMS have been created. Numerous messages have been written that are sent back and forth between the objects as their interaction proceeds in the system.

In this chapter, the object oriented representation of the framework and the interactions between them were described. These fulfill objectives 2 and 3 of the research proposal. Only the highlights of the development have been set forth. Further details can be obtained from the Smalltalk code presented in Appendix B.

CHAPTER VII

EVALUATION OF THE METHODOLOGY

Introduction

This chapter presents an evaluation of the proposed methodology. This evaluation has been done in the context of the measures of merit outlined below. For the purpose of the evaluation, the alternative methodologies as outlined in Chapter V have been implemented in the class hierarchy of FMSController as detailed in Chapter VI. Besides the controller classes, a FMS simulator has been built. An order generator was also constructed to load the FMS simulator. The statistical results of the simulation were then used for the quantitative evaluation.

Measures of Merit

This section identifies performance measures to be used in the evaluation of methodologies for scheduling and controlling random flexible manufacturing systems. A FMS is usually a part of a hierarchical organizational structure. Any performance goal for the control of the FMS is thus subservient to the higher level goals of the organization. A methodology to carry out the lower level tasks should thus be evaluated in terms of its ability to achieve the major goal. However, due to the complexity of the organizations, it is not possible to devise a complete mathematical formulation of the objective (supremal) of the manufacturing system. Thus, there is no objective of the infimal that could be set forth from the supremal.

Therefore, one is led to the use of surrogate objectives for the scheduling and controlling function of manufacturing systems. It should also be said that, a single

objective is not enough to capture the requirements placed on scheduling and controlling. Thus the performance measures are multiple criteria to be used for the evaluation. Many such performance measures have been suggested in the literature.

Average Weighted Tardiness

The main requirement placed by the upper planning and control hierarchy on the shop floor control is the target time when the work is to be completed. This is especially true of the job shop environment considered in this research. A job shop environment is, by definition, a make to order (MTO) situation. The orders as they arrive at the organization have due dates assigned to them: by the customer, or promise dates assigned by the management. This due date is translated into due dates for work parts that flow through the system. From the view point of the upper planning levels, it is necessary to meet the set due dates with due consideration to the priority assigned to a product. Even if the FMS is operated very efficiently, if high priority work is delayed or work needed early is finished late while work needed late is finished early, the FMS operation has not performed well. Thus, due date performance is the primary criteria to be used for the evaluation of scheduling and controlling algorithms. The measure of average weighted tardiness is suggested for the evaluation of control frameworks. The tardiness of a high priority work order is assigned a weight of 2 and that of low priority work is assigned a weight of 1. The steady state average value of this measure is then used for evaluation.

tardiness = max (0, finish time - due date)

weighted tardiness = 2 * tardiness for high priority orders

= 1 * tardiness for low priority orders

Sojourn Time

In some literature dealing with the planning of FMS, the authors have used the utilization of the machines as the objective. This is considered necessary especially in view of the expense incurred in the building of the system. The use of this criterion can be justified only for the static loading of the system often considered in the literature. When only current work in hand is considered, it is justifiable to use the machines as much as one could use them so that they are available for future use. But when the dynamic situation of continuously arriving work is considered, maximization of the average utilization of the machines can not be used as a measure of merit. Average utilization is directly determined by the arrival rate of the work. In the long term, the average utilization will assume the same value as the traffic density (if traffic density < 1). Thus, measure of efficient operation of the FMS is not the average utilization, but the queueing delay: how long the incoming work must wait before the work is finished. Thus, a second measure of merit used in this research is the average time a work part spends in the system after arrival at the FMS load/unload station.

C.P.U. Time

The current research aims at developing a methodology for scheduling and controlling a random FMS on-line. While sophisticated search techniques could be used off-line to find an optimal schedule for this problem, it is contended that they are not suitable for dynamic operation, where decisions are needed on-line. The requirement of real-time control, however, varies with the application. What is considered real-time in some applications may be too slow to handle some other applications. For the macro level operation envisioned in this research, an algorithm is considered fast enough if it can find a decision within a time which is of the order of the time between successive macro events in the FMS - arrival of parts, completion of processing, completion of transport,

breakdown, repair etc. It is apparent that the requirement is not as severe as that of real time process control of the FMS. Nevertheless the time taken for decision making by the control software is a useful benchmark. Thus, another performance measure suggested in this research is the milliseconds of actual CPU time needed for control decisions per hour of operation of the (simulated or real) system. This measure is necessarily hardware dependent. But if all the compared algorithms use the same hardware, meaningful comparisons can still be made.

Target System Flexibility

Another useful measure for comparison of control methodologies is the latitude allowed for differences between target systems. How flexible is the modeling capability of the framework? Flexibility here refers to the capability to address different FMS configurations. For example, if a new component such as a robot is introduced in the target system will a complete rewriting of the model be necessitated? While the three earlier performance measures are quantitative, this last measure is a qualitative one.

This section presented four performance measures suitable for the evaluation of scheduling and controlling methodologies for random FMS. The following two sections describe the experimental setup developed for the quantitative evaluation.

FMS Simulator

A highly detailed FMS simulator has been implemented in the Smalltalk-80 software environment using the components described in Chapter VI. Figure 1 on page 59 shows a schematic of the physical configuration. The simulator permits the modeling of the machines, transport devices, fixtures, and tools. These physical components have different types and exist in limited quantities. It is possible to represent the limited queue capacity of the machines. The machines can have individual distributions of time between failure and time to repair. The events of arrival of work part, mounting on the

needed pallet, its movement within the FMS, retrieval of pallet, and eventual exit from the FMS are modeled in complete detail. The machines, in turn, are allowed to fail, to get repaired, to get blocked by work pieces which can not move, and to get the tools in their tool magazines changed.

A complete listing of the Smalltalk-80 code implementing this simulator and the order generator described in the next section may be found in Appendix B.

Order Generator

An order generator to load the FMS simulator was implemented in Smalltalk-80. It consists of the class LoadGenerator. Its main instance variables are numberOfMachinesVisited, which is the distribution of number of machines in the routing of an order; numberOfAlternates, the distribution of number of alternate machines for any operation; arrivalDistribution, the distribution of time between arrival of the orders; visitTime, the distribution of process time for the operations; and orderSize, the distribution of size of an order. Once initialized with these objects, the order generator schedules itself with the active simulation processor to generate arrival of the orders. The main methods defined for this class are:

1) seed: aSmallInteger. This message permits reseeding the random number stream for all the distributions used by the LoadGenerator.
2) scheduleArrival. This method causes the LoadGenerator to activate. It accesses the active simulation processor and schedules the arrival of orders as specified in its instance variables.
3) createOrder. This method makes the LoadGenerator take samples from its various distributions and to create an order to be sent to the FMS.

The order generated by the order generator consists of a randomly specified number of work parts with one routing, with alternate machines specified. The required tools are also specified. These orders have one of high or low priorities. However, they can

not be executed until the part arrives to the FMS (there is another specified distribution for this delay).

Experimental Evaluation

The order generator, the FMS simulator and the FMS controller were used to evaluate the proposed methodology vis-a-vis alternative methodologies experimentally, via discrete event simulation. Figure 14, on next page, depicts the interactions of simulation objects constructed for the purpose of the evaluation. The experimental setup is described in the following sub-sections.

Fixed Component of the Experiment

1. A hypothetical FMS cell was modeled consisting of 8 NC machines, a load/unload station, a tool crib, pallets and 4 AGV's.
2. The number of transportation devices (AGV's) was fixed at 4.
3. The input and output queue capacities at each machine was fixed at 2.
4. The processing time for each operation was distributed uniformly from 5 to 25 minutes. The processing time sampling was done at the time of creation of the order by the order generator. This time was assigned to the order and was used throughout the life of the order.
5. Transportation time was uniformly distributed from 1.5 to 4 minutes.
6. Each machine was equipped with a tool magazine of capacity 10 slots.
7. The tool for a process was randomly selected from a total of 100 tool types in the tool crib. It was assumed that two copies of each tool are available. A setup time of 30 minutes was incurred in mounting/dismounting of tools for the 8 machines and in transportation from/to the tool crib. One tool occupies one slot, and one operation requires only one tool.

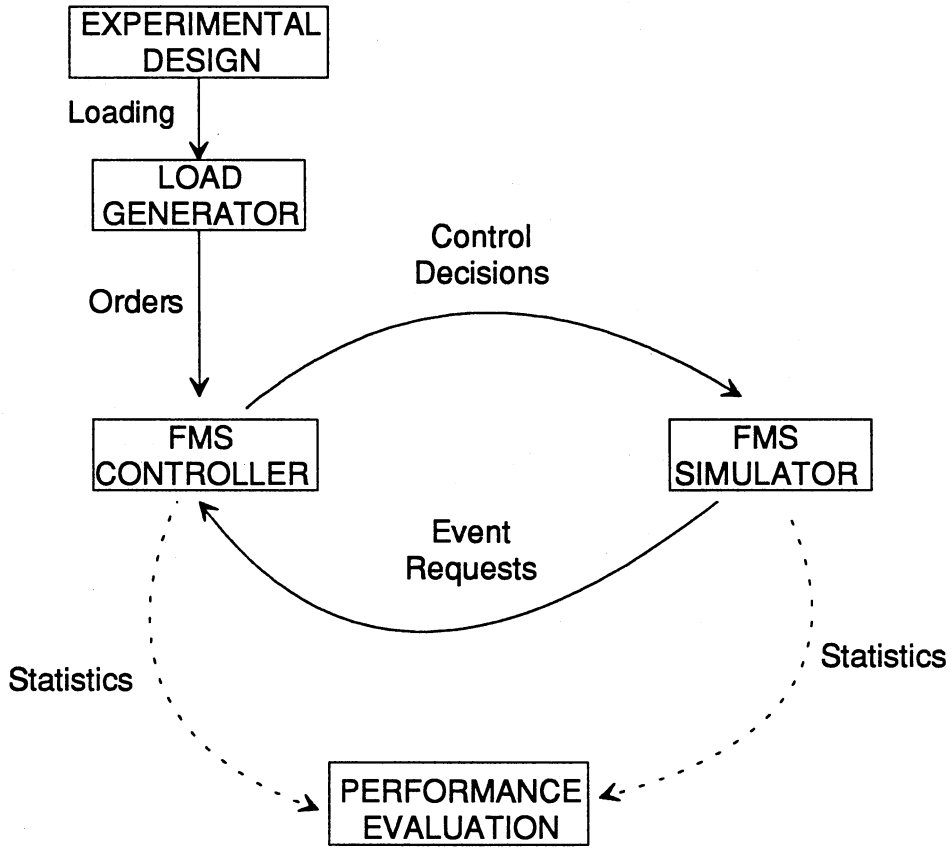

Figure 14. Flexible Manufacturing System Simulation

8. It is assumed that there is no need to adjust the position of work part in a pallet, or to change the pallet through the entire time a work part visits the flexible manufacturing system.

9. Orders for a random (3 to 8) number of parts of a specific part type were generated with the inter-arrival time following an exponential distribution to maintain the desired traffic density.

10. Each part type requires one pallet type. There are 50 pallet types, with 5 copies of each type being available.

11. Due dates are assigned to each order on the basis of the total work content. A factor of 5 is used. That is, if an order is generated at current time,

$$\text{Due date} = \text{current time} + \sum_{\text{All operations}} \text{process time} * \text{order size} * 5$$

12. Machine failures. The machines fail with an exponential distribution of time between failures (500 hours) and uniform distribution of time between repairs (5 hours to 10 hours). When the failure of a machine occurs, the work part under process is not scrapped. Instead, it is assumed to need only the remaining part of its processing time.

Variable Part of the Experiment

A complete factorial experiment has been carried out with the following factors and levels:

1. Control approach: The scheduling and controlling methodology is a factor in the experiment. Five different approaches (levels) were evaluated:

a) Fixed period release. This assumes a scheduling period of 480 minutes. Details about this heuristic are presented in Chapter V. The heuristic is used for release of parts and aims at a target machine utilization of 480 minutes at each release. Then one of the following priority rules (each is a level) is used for dispatch of the parts.

i) SPT. The work part with the shortest imminent processing time is picked first from the queue. However, if there is a part which is already late, then this part is preferred. If more than one part is late then the part with the least slack is chosen, where slack is given by

Slack = Due date - current time - remaining work

This approach is called FSPT in the following discussion.

ii) SLACK. The work part with the least slack time is picked first. Hereafter, this approach is called FSLACK.

b) Priority release. All the order releases are based on the priority rule which is also followed for dispatching the work part through the FMS. Every time an order is finished, as many work parts (with the priority decided by priority rule) are released as feasible. Two rules were investigated:

i) SPT. This is the same rule as described above.This approach will henceforth be called PSPT.

ii) SLACK. This rule picks the order or the part with the lowest remaining slack time. This control policy is called PSLACK in the following.

c) Variable period release. This is the comprehensive scheduling and controlling approach developed in this research. The rules and algorithms followed for this methodology are set forth in Chapter V. This approach is called SIMRULE in the following discussion.

Thus, the five levels of the factor "control approach" have been labeled FSPT, FSLACK, PSPT, PSLACK, and SIMRULE.

2. Traffic density. FMS loading was another factor in the experiments. This was to determine the effectiveness of the methodologies for different loads on the FMS. Two levels of traffic density (60% and 70%) were used.

3. Routing Flexibility. The number of alternates provided for each operation was also used as a factor. This was to determine the impact of flexibility and control approach on the performance measures. Two levels of alternate routings were employed. The first permitted one to two (uniformly distributed) alternate operations for each stage of processing. The second permitted two to three (uniformly distributed) alternate operations.

In summary, the factorial design can be represented by the diagram presented in Figure 15, on next page.

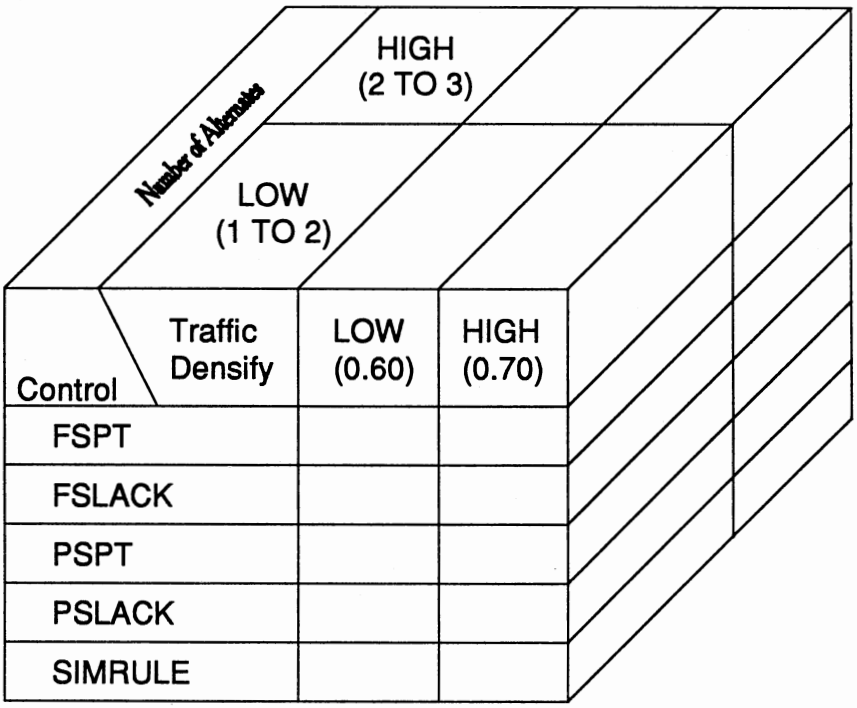

Figure 15. Factorial Design of Simulation Experiments

Validation of the Model

The load generator, FMS simulator and FMS controller depicted in Figure 14, above, were validated using the following procedures.

Random Number Generator. The success of any simulation experiment depends to a large measure on the availability of a number of good random number streams. For this purpose, a random number generator (Payne, Rabung, and Bogyo [1969]) recommended in Law and Kelton (1991) was implemented in Smalltalk-80. Reportedly, this generator is well-tested and works correctly. Up to 100 streams 100,000 numbers apart can be obtained.

Trace Check. The orders and the resulting work parts were followed through the system from their creation to the completion for a number of simulation runs. All events were found to occur in the simulation as described in this Chapter, above, and in Chapters 5 and 6. The work parts followed the routing (alternate) specified, and the correct tools and process times and transport times were used. Specifically, the following events were checked for correct occurrence:

Creation of order

Generation of work parts from the order

Movement of transport vehicles

Failure and repair of machines

Tooling setup

Flow of work part: creation, waiting for pallets, pallet loading, waiting for transport, movement to machine, waiting for machine, processing, further movements and processings, final movement to load/unload station, retrieval of pallet (including integrity of pallet numbers), completion of an order, data collection on weighted tardiness of the order and sojourn time of the work part

Machine allocation algorithms

Triggering of a release cycle

Use of alternate routing

Priority rule (queue discipline) operation

Load Generator Check. Tests were conducted on the load generator to see if the orders and the work parts were generated as specified. Data from a run of 100,000 simulated minutes are given in Appendix A, where it can be seen that the load generator performs as specified.

Normal Flow of Parts. Tests were conducted to assure that the parts flowed through the system evenly: no parts should get stuck in the system, and all parts should

be tracked until they complete their journey through the FMS simulator. For this purpose, periodic snapshots of the system were taken every 1000 simulated minutes to check on the parts and orders in the system. All parts were tracked, with none being lost or stuck in the system.

Calculation of Statistics. Calculation of sojourn time and lateness were hand checked for some simulations of short durations and found correct.

Utilization Check. A number of simulation runs were conducted in which the load generator was set to generate a 60% traffic intensity. After the model had attained steady state, utilization statistics were collected for a further 60,000 simulated minutes. The average utilization was found to be very close to 60% (See Appendix A). Since the load generator was pretested, this independent test shows that indeed the parts do visit the resources as specified and the model runs as envisaged.

Replications

Five replications were made under each of twenty combinations. For each experiment, two random number streams were used: one for the order generator, the other for the failure and repair cycles. In order to curtail variation of the estimates, common random numbers were used: one set for each replication. That is, for each experimental condition, the same five sets of common random number streams were used for the five replications. The experiment was analyzed with each replication as a block.

Simulation Termination

Preliminary simulation runs were carried out to determine when the simulation can be assumed to have warmed up. The graphical procedure of Welch (Law and Kelton, 1991) was used to determine the time period. For each cell in the experimental design, simulations were carried out until the number of work parts in the system could

be assumed to have reached steady state. Statistics were cleared at that point in time, and observations were taken for the next 20,000 (simulated) minutes.

Analysis of Experimental Results

A total of 100 experiments were carried out, 5 replications for each of 20 combinations of the factor levels. Each experiment resulted in observations of 3 responses:

1. Average weighted tardiness, in minutes.
2. Average sojourn time, in minutes.
3. C.P.U time (milliseconds real time per hour of simulated time)

For all three responses, the linear model is

$$X_{ijkm} = \mu + C_i + T_j + F_k + R_m + CT_{ij} + CF_{ik} + TF_{jk} + CTF_{ijk} + \varepsilon_{m(ijk)}$$

where

X_{ijkm} ε {Average weighted tardiness, Average sojourn time, CPU time}

μ = Average response over all the populations

C_i = Effect of the control policy (i = 1 , 5)

T_j = Effect of the traffic density (j = 1, 2)

F_k = Effect of routing flexibility (k = 1, 2)

R_m = Effect of the replication block (m = 1, 5)

CT_{ij} = Interaction between control policy and traffic density

CF_{ik} = Interaction between control policy and routing flexibility

TF_{jk} = Interaction between traffic density and routing flexibility

CTF_{ijk} = Interaction between control policy, traffic density and routing flexibility

$\varepsilon_{m(ijk)}$ = Random error for replication m within the cell i, j, k.

The results of the experiment were analyzed using a SAS program. The SAS program with the embedded data is presented in Appendix A. The complete output of

the SAS computations is also included in Appendix A. The conclusions drawn from the output are presented in the following sub-sections.

Average Weighted Tardiness

With respect to the dependent variable of average weighted tardiness, the analysis of variance is presented below in Table II.

TABLE II

ANOVA WITH AVERAGE WEIGHTED TARDINESS AS DEPENDENT VARIABLE

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Control	4	66419280.2	16604820.1	25.04	0.0001
Flexibility	1	79663720.5	79663720.5	120.12	0.0001
Traffic	1	252219549.5	252219549.5	380.29	0.0001
Replication	4	1205315.2	301328.8	0.45	0.7689
Control * Flexibility	4	16856384.7	4214096.2	6.35	0.0002
Control * Traffic	4	65307009.5	16326752.4	24.62	0.0001
Flexibility * Traffic	1	73240484.8	73240484.8	110.43	0.0001
Control * Flexibility * Traffic	4	17722746.2	4430686.6	6.68	0.0001
Model	23	572634490.7	24897151.8	37.54	0.0001
Error	76	50405230.4	663226.7		
TOTAL	99	623039721.1			

The significance of the overall F - test is very high, indicating that the model accounts for a significant portion of the variability in the average weighted tardiness. All the main factors of control policy, routing flexibility, and traffic density are significant, as are the interactions. This shows that not only the different levels of control policies have different mean values of average weighted tardiness, but also the means are different for different combinations of control policies with routing flexibilities and traffic densities.

The mean values of the average weighted tardiness at different levels of traffic density and routing flexibility are shown in Table III, next page. From Table III, it can be concluded that although there is high variation in the ordering of the rules (there is interaction between the experimental conditions and the control policy), SIMRULE has the least average weighted tardiness. The coefficients of variations (in percent) vary widely although SIMRULE seems to have the lower value in the tables. For low traffic density situations, there is hardly anything to choose between the control policies, but SIMRULE proves itself in high traffic density situations.

To find the significant differences among the control policies, Duncan's multiple range test was carried out. A significance level of 0.05 was used. The result for the average weighted tardiness measure is shown below in Figure 16. Mean values of performance measure are shown in parenthesis. A line is drawn under any set of means for which the differences are not statistically significant.

FSLACK	FSPT	PSPT	PSLACK	SIMRULE
(2461.9)	(2400.1)	(2026.6)	(1973.0)	(236.9)

Figure 16. Multiple Comparison of Control Policies for Average Weighted Tardiness

TABLE III

AVERAGE WEIGHTED TARDINESS MEASURE AT COMBINATIONS OF FACTORS

	FSPT	FSLACK	PSPT	PSLACK	SIMRULE
High Traffic Density					
Mean	4640.96	4731.34	3691.46	3684.65	290.72160
Coeff of Var	54.834911	52.027082	72.384493	70.882520	54.475321
Low Traffic Density					
Mean	159.23664	192.52940	361.67210	261.25260	183.01144
Coeff of Var	45.094461	35.433600	32.985442	29.914974	64.385205
High Routing Flexibility					
Mean	1340.19	1462.77	860.24310	816.47880	156.00474
Coeff of Var	126.98730	116.47315	107.37896	93.356579	36.495273
Low Routing Flexibility					
Mean	3460.00	3461.09	3192.89	3129.42	317.72830
Coeff of Var	101.14546	101.69072	96.215925	99.959258	52.324947

From the multiple comparison, it can be concluded that the SIMRULE control policy has significantly lower average weighted tardiness. Priority rule based release seems to perform better than fixed period release, although the difference is not significant.

Sojourn Time

The results of analysis with the sojourn time as the dependent variable are now presented. The table for the analysis of variance is given in Table IV.

TABLE IV

ANOVA WITH SOJOURN TIME AS DEPENDENT VARIABLE

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Control	4	75651069.1	18912767.3	28.58	0.0001
Flexibility	1	76282230.2	76282230.2	115.26	0.0001
Traffic	1	259675689.0	259675689.0	392.35	0.0001
Replication	4	1027218.8	256804.7	0.39	0.8166
Control * Flexibility	4	13916729.8	3479182.5	5.26	0.0009
Control * Traffic	4	68620665.1	17155166.3	25.92	0.0001
Flexibility * Traffic	1	67988776.5	67988776.5	102.73	0.0001
Control * Flexibility * Traffic	4	15208756.1	3802189.0	5.74	0.0004
Model	23	578371134.6	25146571.1	37.99	0.0001
Error	76	50300095.7	661843.4		
TOTAL	99	628671230.3			

The overall degree of significance for the model is very high, thus indicating that the model is able to represent substantial amounts of variation in the average sojourn time values. All the main effects of the major factors are significant. All the interac-

tions are also significant. This suggests that the effect of all the factors is different at different levels of the rest of the factors.

The mean values of average sojourn time at combinations of routing flexibility and traffic density with the control policies is given in Table V.

TABLE V

AVERAGE SOJOURN TIME MEASURE AT COMBINATIONS OF FACTORS

	FSPT	FSLACK	PSPT	PSLACK	SIMRULE
High Traffic Density					
Mean	5554.28	5647.92	4298.81	4303.94	980.24290
Coeff of Var	46.802449	44.013399	57.416027	55.750829	25.180545
Low Traffic Density					
Mean	785.06730	903.55450	1143.77	1078.72	759.60880
Coeff of Var	16.707651	13.592149	16.337199	11.682774	25.915209
High Routing Flexibility					
Mean	2071.69	2256.35	1649.83	1642.26	740.83760
Coeff of Var	88.429605	79.717050	54.572341	45.255044	19.214315
Low Routing Flexibility					
Mean	4267.66	4295.12	3792.76	3740.40	999.01
Coeff of Var	85.698285	84.444074	75.938354	78.194089	26.375149

From the above tabulation of means of average sojourn times, it is apparent that the least sojourn time for all the levels of routing flexibility and traffic densities is given by SIMRULE. However, there is interaction between the control policies and the loading conditions: different orderings of control policies are obtained for different experimental conditions. As in the case of average weighted tardiness, there is not much to choose for low traffic conditions, but SIMRULE performs better for high traffic conditions. The coefficient of variation also appears to be lower for SIMRULE.

Multiple comparison of means for different control policies was carried out using Duncan's multiple range test. The result is presented in Figure 17, where the means under each control policy is presented in parenthesis. Policies which do not differ significantly are grouped together by an underline. The SIMRULE control policy is significantly better than the other two approaches, and FSLACK is significantly worse than PSPT and PSLACK, which are not significantly different.

FSLACK	FSPT	PSPT	PSLACK	SIMRULE
(3275.7)	(3169.7)	(2721.3)	(2691.3)	(869.9)

Figure 17. Multiple Comparison of Control Policies for Average Sojourn Time

C.P.U. Time

The third response variable considered in the simulation experiment was the C.P.U. time (in milliseconds per simulated time of FMS operation) required for control

decisions under each of the approaches. The result of this analysis is presented below. The table of analysis of variance is given in Table VI.

TABLE VI

ANOVA WITH C.P.U. TIME AS DEPENDENT VARIABLE

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Control	4	280273.2860	70068.3215	577.16	0.0001
Flexibility	1	3506.0894	3506.0894	28.88	0.0001
Traffic	1	27779.7456	27779.7456	228.82	0.0001
Replication	4	790.9630	197.7407	1.63	0.1757
Control * Flexibility	4	657.2528	164.3132	1.35	0.2581
Control * Traffic	4	2845.0869	711.2717	5.86	0.0004
Flexibility * Traffic	1	0.0005	0.0005	0.00	0.9984
Control * Flexibility * Traffic	4	1892.7905	473.1976	3.90	0.0062
Model	23	317745.2147	13815.0093	113.80	0.0001
Error	76	9226.5584	121.4021		
TOTAL	99	326971.7731			

The significance of the overall model is very high indicating that the model does explain a significant portion of the variation in C.P.U. Time. The analysis of variance shows that the main effects of control policy, routing flexibility, and traffic density are significant. The three way interaction between control policy, routing flexibility and

traffic density is also significant indicating that the combinations of these are different from each other for their effect on C.P.U. Time.

The mean values of C.P.U. Time for combinations of control policies with levels of routing flexibility and traffic densities are presented in Table VII.

TABLE VII

C.P.U. TIME MEASURE AT COMBINATIONS OF FACTORS

	FSPT	FSLACK	PSPT	PSLACK	SIMRULE
High Traffic Density					
Mean	32.763655	40.168395	65.966390	66.717515	192.36323
Coeff of Var	46.093319	61.478724	13.0533	8.9010865	9.3341116
Low Traffic Density					
Mean	8.3934850	7.9706100	37.470565	39.186300	138.28566
Coeff of Var	40.071391	44.162017	27.880448	27.330566	11.853789
High Routing Flexibility					
Mean	26.058965	34.698815	56.825620	58.538890	168.12673
Coeff of Var	75.96911	84.364525	23.799459	19.949570	19.325333
Low Routing Flexibility					
Mean	15.098175	13.440190	46.611335	47.364925	162.52216
Coeff of Var	69.677186	70.651451	42.559952	40.353025	20.798946

Perusal of the above results shows that SIMRULE uses the highest amount of C.P.U. Time, followed by the priority rule based releases and the fixed period releases. There is no appreciable difference between the SPT and the SLACK rules. There does not seem to be any reason to prefer one dispatching rule over another on account of their effect of the C.P.U. Time. The coefficient of variation of C.P.U. Time is also the lowest for SIMRULE.

Duncan's multiple range test was carried out to find which control policies were significantly different in terms of their effect on C.P.U. Time. The result of the analysis is presented in Figure 18. The mean C.P.U. Time (millisecond/hour of simulated time) is given in parenthesis. The control policies which do not differ significantly are grouped together by an underline.

SIMRULE	PSLACK	PSPT	FSLACK	FSPT
(165.324)	(52.952)	(51.718)	(24.070)	(20.579)

Figure 18. Multiple Comparison of Control Policies for C.P.U. Time

As can be seen from the figure, SIMRULE has the highest mean C.P.U. Time which is significantly different from the others. This is followed by the priority rule based releases which are grouped together. Fixed period release takes significantly lower C.P.U. Time. It is apparent that SLACK and SPT rules do not differ significantly in terms of C.P.U. Time.

Summary of Statistical Analyses

The proposed methodology (SIMRULE) provides significantly lower average weighted tardiness and lower average sojourn time for the hypothetical situation explored in this simulation experiment. The coefficients of variation of these measures also appear to be lower. Routing flexibility and traffic density also have significant impact on the two measures. This impact exists when the factors are taken alone or in combination with the control policies investigated. Table VIII, below, presents a summary of the mean value of performance measures.

TABLE VIII

SUMMARY OF PERFORMANCE MEASURES

	SIMRULE	FSPT	FSLACK	PSPT	PSLACK
Average Weighted Tardiness	236.9	2400.1	2461.9	2026.6	1973.0
Sojourn Time	869.9	3169.7	3275.7	2721.3	2691.3
C.P.U. Time	165.32	20.58	24.07	51.72	52.95

As was expected, SIMRULE requires significantly higher C.P.U. Time than the other methods evaluated. It is far more comprehensive than any control policy previously

reported in the literature. It is gratifying, however, that the mean value of C.P.U. Time, 165 millisecond per simulated hour of operation is still a very small value when considering the dynamics of decision making in a real world scheduling environment. .

It may be pointed out, however, that there exists a significant interaction between all the three factors. This interaction exists for all the three response variables tested. For this reason, the comparison of the control policies should be taken with caution, particularly for lateness. For lateness, SIMRULE performs worse than FSPT when the traffic density is low. But it performs significantly better when the traffic density is high. It may very well be the case that, for this particular configuration, performance of SIMRULE is better only in a narrow region around the higher traffic density tested. Further investigations are needed with 1) other levels of traffic density, both higher and lower, 2) various system sizes and 3) more control policies to comprehensively test the methodology developed in this research.

Target System Flexibility

Object oriented programming provides a high degree of modeling flexibility on account of the modularity of the objects. The objects represent some particular aspect of the system that is being modeled. The objects isolate those specific behaviors independent of the other aspects. When an appropriate message is passed to them, they exhibit the specified behaviors. Thus they are very amenable to changes in other objects in the model. The new objects just need to use the old methods defined for the old objects to obtain the desired behaviors.

For example, if a new material handling device such as a robot needs to be introduced into the current framework, virtually no change is needed in other objects such as the resources hierarchy, or the work flow items. Only the methods needed for the robot need to be defined. The robot object can still access the resources, work flow items, and the controller as before.

Furthermore, use of inheritance in object oriented programming permits use of some existing object when the new behavior is close to that of another previously defined object. Only the differences need to be coded, all the similarities can be inherited. Thus, in the above example, the robot can inherit many of the behavior of transportDevice, already defined in the framework.

Thus an object oriented framework such as developed in this research provides much more flexibility in modeling the control aspects of complex systems than a procedural programming framework could provide. It is possible to obtain modularity in traditional programming languages but not to the same degree as in OOP. Further, OOP highly enhances reusability of code in comparison to procedural programming paradigm.

The above discussion applies generally to all aspects of modeling of scheduling and control. It has even more applicability when discrete event simulation is part of the control strategy. Using control simulation for an evaluation of control options is an area of active research. This strategy requires on-line construction and exercise of simulation models. The flexibility of OOP in construction of simulation models has been widely investigated and publicized. Based on these results, it seems fair to say that the object oriented framework developed in this research has higher flexibility than other methodologies involving control simulation.

Another aspect of target system flexibility provided by the framework is the decomposition of the problem into rules and heuristics. Decomposition may lead to some sub-optimization but when a change in the target system is required, it may still be possible to retain certain components of the decomposition, making changes only in the other components. This can not be said of a monolithic mathematical programming formulation, where even a small change is likely to require a complete reformulation and even a change in the mathematical approach used.

Chapter Summary

In accordance with objective 4 of the research, three quantitative and one qualitative performance measures have been identified in this chapter for evaluation of scheduling and controlling methodologies for random FMS. These measures were then used to evaluate the methodology developed in this research, SIMRULE, against alternate methodologies in fulfillment of objective 5.

The SIMRULE methodology is superior in terms of the main measure of performance - average weighted tardiness. It is also superior in terms of the second measure of performance - average sojourn time. But it is inferior when it is evaluated against the third measure - C.P.U. time. However, this disadvantage does not appear to be a serious handicap, and is far outweighed by its superior decision results.

Using an object oriented framework for controlling and scheduling random FMS seems advantageous from the viewpoint of the flexibility offered, especially when a control simulation is part of the decision evaluation process.

CHAPTER VIII

SUMMARY, CONCLUSIONS AND RECOMMENDATIONS

Introduction

This chapter presents concluding thoughts about the research. First a summary of the research is presented. Then, the contributions of the research are pointed out. Finally, recommendations for future research are cited.

Research Summary

The goal of this research has been the development of a comprehensive methodology for scheduling and controlling random FMS. The requirement of comprehensiveness implies the consideration of all the specified constraints. Another requirement is on-line processing. Keeping this goal in mind, six objectives were identified for this research. These objectives and the status of their attainment are now discussed.

Methodology

The first objective was the development of a comprehensive methodology for scheduling and controlling random FMS that is capable of generating consistently good solutions. Although there exist many algorithms for static scheduling for job shops, the only viable approach to dynamic job shop scheduling is the use of priority rules. Similarly, it is contended that effective control of random FMS will require development of rules to handle the different situations encountered therein. At the same time, discrete

event simulation is the only modeling approach that is capable of capturing all the complexity of random FMS.

The methodology developed in this research consists of a decomposition of the problem, where heuristics, rules, and simulation are used to make the control decisions needed for random FMS. Heuristics have been developed to address the problem of release of work parts. Rules are used for the selection of the heuristics. Rules and simulation are used for selection of dispatching rules. Rules also handle events such as machine failure and machine repair.

Using these solution elements, the goal of developing a comprehensive methodology for scheduling and controlling random FMS was achieved.

Object-Oriented Representation

The second objective was to determine the classes and subclasses of objects required for the representation and expression of the random FMS scheduling problem in the object oriented paradigm.

Smalltalk-80, an object oriented language, was used as the vehicle to develop this representation. Figure 13 on page 108 depicts the class hierarchy created in this research. The OOP principles of modularity and inheritance were adhered to in this development. In a nutshell, objects of the class Resource and its subclasses represent physical equipments and pallets; objects of the class FMSController and its subclasses represent decision/control elements; CimSimulation carries out simulation processing; and WorkFlowItem represents work parts and their processes.

The above classes and other classes developed in this research are capable of expressing the problem of controlling a random FMS within all the constraints specified in this research.

Development of Framework

The next objective was the development of an object oriented framework for the interactions of the components within the environment being developed that would be capable of implementing the methodology while operating in a dynamic, on-line environment.

The research was successful in developing a framework of messages between the objects for implementation of the methodology outlined in Chapter V. In this framework, like in the real system, the physical objects send event requests to the decision/control object which then sends its decision to the physical objects. The controller object may create its own control simulation process in order to arrive at a decision, as discussed in Chapter V.

Measures of Merit

The fourth objective was to develop and validate measures of merit for evaluating scheduling and controlling methodologies for random FMS. Three quantitative and one qualitative measures of merit were developed. The most important measure is the average weighted tardiness. The second yardstick is the sojourn time taken by work parts inside the FMS. The third comparative figure is the C.P.U. time taken for control decisions. A qualitative criterion for selection of control methodology is the flexibility of the methodology to the variations in the target system.

Evaluation

This objective consists of evaluation of the methodology developed in this research. A full factorial simulation experiment was carried out with two levels of route flexibility, two levels of traffic density, and five levels of control policies. The SIMRULE methodology developed in this research effort was found to be superior to the

other control policies compared in this research on the criteria of average weighted tardiness, and sojourn time. The C.P.U. time taken for control decisions was highest, but even this time was not unconscionably high. It was also argued that the object oriented methodology is particularly attractive when target system flexibility is considered.

Further Research

The final objective was the identification of further research in this area. Certain avenues of future research were identified, and they are discussed in the final section of this chapter.

Contributions of the Research

The literature of scheduling is vast. Much research work has been carried out in different areas of scheduling. Scheduling of random FMS is just one such area and many researchers have applied their efforts in this field. However, a disconcerting aspect of this research are the various idealizations that permit a succinct expression and resolution of the problem, but leave many realities of the problem out of consideration. The current research was an attempt to fill this void. The completion of the objectives as set forth in the preceding section makes the following contribution to this area of Industrial Engineering and Operations Research.

Representation Issues

This research developed an object oriented representation of the complete random FMS problem. This included the decision/control elements, the information elements, and the physical elements and their interactions. This representation provides the OOP features of modularity, reusability, and separation of decision/control, information, and physical elements.

The most significant aspect of this representation is its intuitive appeal. In every case, an object is defined separately from others, and exhibits behaviors that one would intuitively expect from it. For example, a material handler is represented by an instance of TransportDevice and exhibits some behaviors of Resource. The behavior of Resource is coded separately from that of other objects, and some of this behavior is inherited by TransportDevice. Other behaviors of material handler are independently coded.

Another significant contribution of the representation is the separate modeling of decision/control elements. This permits convenient change of control policies. FMSController represents the control policies to be followed, and when a change in control policy is required, it just entails the use of a different subclass of FMSController. In this way, a high degree of reusability of code is achieved.

Methodology Development and Evaluation

A comprehensive methodology was developed for scheduling and controlling random FMS. This included all the multiple constraints as set forth in the problem statement in page 7. One or more of these dimensions of the problem are usually ignored, and, to the knowledge of the author, all of them are never considered together as they were in this research. Thus the development of this methodology is a significant contribution of this research.

Dynamic Environment Considered

Whereas most of the existing approaches to scheduling view the shop floor as static, the current research attempts to account for the constantly changing dynamic nature of a real world random FMS. This contribution is especially significant in view of the comprehensive realistic manner in which the methodology in this research considers the dynamic environment..

Test Bed for FMS Control Policies

The implementation of the object oriented framework, as carried out in this research, provides a test bed for comprehensive evaluation of control policies designed for automated manufacturing systems. All the features of a FMS are implemented, so the control policies have to make provisions for these features. Furthermore, the OOP paradigm permits many variations of the basic model to be easily developed by using subclasses, and thus provides a very high degree of reusability. Therefore, the test bed developed in this research will be fruitful for future work to be undertaken in this field.

Integrated Environment for Control Simulation and Expert Systems

The methodology developed in this research integrates heuristics, expert systems, and simulation. Heuristics are usually implemented in a procedure oriented language, expert systems in a list processing or logic processing language, and manufacturing simulations in a special purpose simulation language. To the knowledge of the author, there is no environment where all this can be done within a single environment in a seamless way. The framework provided by this research provides such a single environment. This is useful both for future research and for practical implementation where heuristics, simulation and expert systems need to be integrated.

Recommendations

As a result of this research effort, some recommendations about future research can be made. These are described below.

Release Levels

As the number of work parts introduced to the FMS at one time increases, initially the performance of the FMS improves. But after a point, the FMS becomes congested and the effectiveness of the FMS starts to degrade. In this research, this point was identified by preliminary simulation. A useful research would be to obtain analytical results that could be used on-line. Of course it can not be expected that an analytical solution to such a complex problem could be set forth in its complete form. But even a solution for a simple FMS could be very useful as an approximation.

Control Simulation Criteria

In choosing the priority rule to be used, total utilization of the machines were used as a criterion for the control simulation even though weighted tardiness has been the major performance criterion for the actual system. Preliminary research showed that better results were obtained with total utilization than the actual criterion of weighted tardiness. A useful line of enquiry would be to identify performance criteria for the control simulation. This would enhance the use of on-line simulation as a control tool.

Control Simulation Input

The only modeling approach that can capture the complexity of a random FMS in its entirety is a discrete event simulation model. As the processing speed of microprocessors goes up, the attractiveness of on-line simulation as a control tool increases, especially when applied to a short time horizon. In this research the only control factor investigated by control simulations was the priority rule. A topic for further research would be identification of other factors to be investigated at the same time.

Distributed Control Simulation

The control simulations for the evaluation of alternatives were carried out in this research using a single processor consecutively. However, these simulations are particularly amenable to distributed processing since there is no interaction between them. One useful line of research would be to develop a framework for distributed on-line object oriented simulation. On-line simulation is currently slow on account of serial processing. Progress in distributed simulation would substantially enhance the use of on-line simulation.

BIBLIOGRAPHY

Adelsberger, H.H. and J.J. Kanet (1989), "The Leitstand - a New Tool for Computer Integrated Manufacturing", In *Proceedings of the Third ORSA/TIMS Conference on Flexible Manufacturing Systems,* 253-258.

Akella, R., Y. Choong, and S.B. Gershwin (1984), "Performance of Hierarchical Production Scheduling Policy", *IEEE Transactions on Components, Hybrids, and Manufacturing Technology,* CHMT-7 (3), 215-217.

Anonymous (1990), *Total Control of Scheduling Dynamics,* Sales brochure, Pritsker Corporation, Indianapolis, Indiana.

Barr, A.B. and E.A. Fiegenbaum (1981), *The Handbook of Artificial Intelligence*, Vol. 1, 343-348, William and Kaufmann Inc., Los Altos, California.

Beaumariage, T.G. (1990), "Investigation of an Object Oriented Modeling Environment for Manufacturing Systems", Unpublished Ph.D. Thesis, School of Industrial Engineering and Management, Oklahoma State University, Stillwater, OK.

Bensana, E., G. Bel, and D. Dubois (1988), "OPAL: A Multi-Knowledge-Based System for Industrial Job-Shop Scheduling", *International Journal of Production Research,*26 (5), 795-819.

Bourne, D.A. and M.S. Fox (1984), "Autonomous Manufacturing: Automating the Job-Shop", *IEEE Computer,* 17 (9), 76-86.

Bruno, G., A. Elia, and P. Laface (1986), "A Rule-Based System to Schedule Production", *IEEE Computer,* 19 (7), 32-40.

Bullers, W.I., S.Y. Nof, and A.B. Whinston (1980), "Artificial Intelligence in Manufacturing Planning and Control", *AIIE Transactions*, 12 (4), 351-363.

Buzacott, J.A. and J.G. Shantikumar (1980), "Models for Understanding Flexible Manufacturing Systems", *AIIE Transactions*, 12 (4), 339-349.

Buzacott, J.A. (1982), "Optimal Operating Rules for Automated Manufacturing Systems", *IEEE Transactions on Automatic Control*, AC-27 (1), 80-86.

Buzacott, J.A. and D.D. Yao (1986), "Flexible Manufacturing Systems: A Review of Analytical Models", *Management Science,* 32 (7), 890-905.

Chang, Y., H. Matsuo, and R.S. Sullivan (1989), "A Bottleneck-Based Beam Search for Job Scheduling in a Flexible Manufacturing System", *International Journal of Production Research,* 27 (11), 1949-1961.

Chiodini, V. (1986), "A Knowledge Based system for Dynamic Manufacturing Replanning", In *Symposium on Real-Time Optimization in Automated Manufacturing Facilities*, National Bureau of Standards, Gaithesburg, Maryland.

Chiodini, V. (1989), "SCORE: An Integrated System for Dynamic Scheduling and Control of High-Volume Manufacturing", Unknown publication.

Conway, R.W., W.L. Maxwell, and L.W. Miller (1967), *Theory of Scheduling*, Addison-Wesley Publishing Company, Reading, Mass.

Conway, R.W. (1965), "Priority Dispatching and Work in Process Inventory in a Job Shop", *Journal of Industrial Engineering*, 16 (2), 123-130.

Conway, R.W. (1965b), "Priority Dispatching and Job Lateness in a Job Shop", *Journal of Industrial Engineering*, 16 (4), 228-237.

Conway, R.W. and W.L. Maxwell (1986), "Low-Level Interactive Scheduling", In *Symposium on Real-Time Optimization in Automated Manufacturing Facilities*, National Bureau of Standards, Gaithesburg, Maryland.

Davis, W.J. and A.T. Jones (1989), "On-Line Concurrent Simulation in Production Scheduling", In *Proceedings of the Third ORSA/TIMS Conference on Flexible Manufacturing Systems,* 253-258.

Donath, M., R.J. Graves, and D.A. Carlson (1989), "Flexible Assembly Systems: The Scheduling Problem for Multiple Products", *Journal of Manufacturing Systems*, 8 (1), 27-33.

Donath, M.W. (1988), "A Scheduling Methodology for Flexible Manufacturing Systems", Unpublished Ph.D. Thesis, University of Massachusetts.

Dupont-Gatelmand C. (1982), "A Survey of Flexible Manufacturing Systems", *Journal of Manufacturing Systems,* 1 (1), 1-16.

French, S. (1983), *Sequencing and Scheduling, An Introduction to the Mathematics of the Job-Shop*, E, Horwood, West Sussex, U.K.

Fox, M.S., B. Allen, and G. Strohm (1982), "Job-Shop Scheduling: An Investigation in Constraint-Directed Reasoning", In *Proceedings of the National Conference on Artificial Intelligence*, 155-158.

Gere, W.S. (1966), "Heuristics in Job Shop Scheduling", *Management Science,* 13 (3), 167-190.

Gershwin, S.B., R.K. Hildebrant, R. Suri, and S.K. Mitter (1986), "A Control Perspective on Recent Trends in Manufacturing Systems", *IEEE Control Systems Magazine*, 6 (2), 3-15, 1986.

Gershwin, S.B. (1989), "Hierarchical Flow Control: A Framework for Scheduling and Planning Discrete Events in Manufacturing Systems", In *Proceedings of the IEEE*, 77 (1), 195-209.

Gevarter, W.B. (1984), *Artificial Intelligence Expert Systems Computer Vision and Natural Language Processing*, Noyes Publications, Park Ridge, New Jersey.

Grant, F.H., S.Y. Nof, and D.G. MacFarland (1989), "Adaptive/Predictive Scheduling in Real Time", In *Proceedings, Advances in Manufacturing Systems Integration and Processes,* Society of Manufacturing Engineers, Dearborn, Michigan

Grant, F.H. (1989), "Scheduling Manufacturing Systems with FACTOR", In *Proceedings of the 1989 Winter Simulation Conference*, 277-280.

Gupta, Y.P., M.C. Gupta, and C.R. Bector (1989), "A Review of Scheduling Rules in Flexible Manufacturing Systems", *International Journal of Computer Integrated Manufacturing*, 2 (6), 356-377.

Han, M., Y.K. Na, and G.L. Hogg (1989), "Real-time Tool Control and Job Dispatching in Flexible Manufacturing Systems", *International Journal of Production Research,* 27 (8), 1257-1267.

Harmonosky C.M. (1990), "Implementation Issues Using Simulation for Real-time Scheduling, Control, and Monitoring", In *Proceedings of the 1990 Winter Simulation Conference,* 595-598.

Hutchison, J., K. Leong, D. Snyder, and F. Ward (1989), "Scheduling for Random Job Shop Flexible Manufacturing Systems", In *Proceedings of the Third ORSA/TIMS Conference on Flexible Manufacturing Systems,* 161-166.

Hwan, S.S. and A.W. Shogun (1989), "Modeling and Solving an FMS Part Selection Problem", *International Journal of Production Research,* 27 (8), 1349-1366.

Jackson, S. and J. Browne (1989), "An Interactive Scheduler for Production Activity Control", *International Journal of Computer Integrated Manufacturing,* 2 (1), 2-14.

Jaikumar, R. (1986), "Postindustrial Manufacturing", *Harvard Business Review,* 64 (6), 69-76.

Jaikumar, R. and L.N.Van Wassenhove (1989), "A Production Planning Framework for Flexible Manufacturing Systems", *Journal of Manufacturing Operations Management*, 2, 52-79.

Jain, S., K. Barber and D. Osterfeld (1990), "Expert Simulation for On-Line Scheduling", In *Proceedings of the 1989 Winter Simulation Conference*, 930-935.

Karacal, S.C. (1990), "The Development of an Integrative Structure for Discrete Event Simulation, Object Oriented Programming, and Imbedded Decision Processes", Unpublished Ph.D. Thesis, School of Industrial Engineering and Management, Oklahoma State University, Stillwater, OK.

Kimemia, J.G. and S.B. Gershwin (1983), "An Algorithm for the Computer control of Production in Flexible Manufacturing Systems", *IIE Transactions,* 15 (4), 353-362.

Kusiak, A. and M. Chen (1988), "Expert Systems for Planning and Scheduling Manufacturing Systems", *European Journal of Operational Research*, 34, 113-130.

Lashkari, R.S., S.P. Dutta, and A.M. Padhye (1987), "A New Formulation of Operation Allocation Problem in Flexible Manufacturing Systems: Mathematical Modeling and Computational Experience", *International Journal of Production Research,* 25 (9), 1267-1283.

Law, A.M. and W.D. Kelton (1991), *Simulation Modeling and Analysis*, 2nd Edition, McGraw-Hill Inc., New York.

Lee, S.M. and H. Jung (1989), "A Multi-Objective Production Planning Model in a Flexible Manufacturing Environment", *International Journal of Production Research,* 27 (11), 1981-1992.

Manivannan, S. and J. Banks (1989), "Design of a Knowledge-Based On-Line Simulation System to Control a Manufacturing Shop Floor", Unknown publication.

Moreno, A.A. and F. Ding (1989), "Goal Oriented Heuristics for the FMS Loading (And Part Type Selection) Problems", In *Proceedings of the Third ORSA/TIMS Conference on Flexible Manufacturing Systems*, 105-110.

Nilsson, N. (1980), *Principles of Artificial Intelligence*, Tiago Publishing Co., Palo Alto, California.

Nof, S.Y., M.M. Barash, and J.J. Solberg (1979), "Operational Control of Item Flow in Versatile Manufacturing Systems", *International Journal of Production Research,*17 (5), 479-489.

Panwalker, S.S. and W. Iskander (1977), "A Survey of Scheduling Rules", *Operations Research,* 25 (1).

Rachamadugu, R. and K.E. Stecke (1989), *Classification and Review of FMS Scheduling Procedures,* Working Paper #481 C, The University of Michigan,Ann Arbor, Michigan.

Robbins, J.H. (1986), "Simulation Helps Schedule Shop", *American Machinist and Automated Manufacturing*, 130 (10), 106-108.

Shanker, K. and Y.J. Tzen (1985), "A Loading an Dispatching Problem in a Random Flexible Manufacturing Systems", *International Journal of Production Research*,23, 579-595.

Shanker, K. and S. Rajamarthandan (1989), "Loading Problem in FMS: Part Movement Minimization", *Proceedings of the Third ORSA/TIMS Conference on Flexible Manufacturing Systems*. 99-104.

Shaw, M.J. (1988), "Knowledge-Based Scheduling in Flexible Manufacturing Systems: an Integration of Pattern-Directed Interference and Heuristic Search", *International Journal of Production Research*,26 (5), 821-844.

Slomp, J., G.J.C. Gaalman, and W.M. Nawijin (1988), "Quasi On-line Scheduling Procedures for flexible Manufacturing Systems", *International Journal of Production Research*,26 (4), 585-598.

Smith, M.L., R. Ramesh, R.A. Dudek, and E.L. Blair (1986), "Characteristics of U.S. Flexible Manufacturing Systems - A Survey", In *Proceedings of the Second ORSA/TIMS Conference on Flexible Manufacturing Systems*. 477-486.

Stecke, K.E. and J.J. Solberg (1981), "Loading and Control Policies for a Flexible Manufacturing System", *International Journal of Production Research*,19 (5), 481-490.

Stecke, K.E. and J.J. Solberg (1982), *The Optimality of Unbalanced Workloads and Machine Group Sizes for Flexible Manufacturing System*, Working Paper No. 290, Graduate School of Business Administration, The University of Michigan, Ann Arbor, Michigan.

Stecke, K. (1983), "Formulation and Solution of Nonlinear Integer Production Planning Problems for Flexible Manufacturing Systems",*Management Science*, 29 (3), 273-288.

Steffen, M.S. (1986), "A Survey of Artificial Intelligence-Based Scheduling Systems", In *Proceedings, Fall Industrial Engineering Conference*, Institute of Industrial Engineers, Norcross, Georgia.

Vollmann, T.E., W.L. Berry, and D.C. Whybark (1988), *Manufacturing Planning and Control Systems*, Third Edition, 169-173, Richard D. Irwin Inc., Homewood, Illinois.

Wilson, J.M. (1989), "An Alternative Formulation of the Operation-Allocation Problem in Flexible Manufacturing Systems", *International Journal of Production Research*, 27 (8), 1405-1412.

Wu, S.D. and R.A. Wysk (1988), "Multi-pass Expert Control System - a Control/Scheduling Structure for Flexible Manufacturing Cells", *Journal of Manufacturing Systems,* 7 (2), 107-120.

Wu, S.D. and R.A. Wysk (1989), "An Application of Discrete-Event Simulation to On-Line Control and Scheduling in Flexible Manufacturing", *International Journal of Production Research,* 27 (9), 1603-1623.

Yancey, D.P. and S. Peterson (1989), "Implementation of Rule-Based Technology in a Shop Scheduling System", In *Proceedings of the 1989 Winter Simulation Conference*, 865-873.

Young, R.E. and M.A. Rossi (1988), "Toward Knowledge-Based Control of Flexible Manufacturing Systems", *IIE Transactions*, 20 (1), 36-43.

VITA

Chuda B. Basnet

Candidate for the Degree of

Doctor of Philosophy

Thesis: ON-LINE SCHEDULING AND CONTROL OF RANDOM FLEXIBLE MANUFACTURING SYSTEMS WITHIN AN OBJECT-ORIENTED FRAMEWORK

Major Field: Industrial Engineering and Management

Biographical:

Personal Data: Born in Dingla, Nepal, July 9, 1948, the son of Dambar B. and Rambha Basnet.

Education: Received Bachelor of Engineering (Mechanical) from University of Poona, India in May, 1973; received the Master of Science degree in Industrial and Management Engineering from Montana State University, Bozeman, Montana in August 1987; completed the requirements for the Doctor of Philosophy at Oklahoma State University in July, 1991.

Professional Experience: Production Engineer, Structo Nepal, Kathmandu, Nepal, 1973 to 1978; Aircraft Maintenance Engineer, Royal Nepal Airlines Corporation, Kathmandu, Nepal, 1978 to 1985; Teaching Assistant, Montana State University, Bozeman, Montana, 1985 to 1987; Teaching and Research Assistant, Oklahoma State University, Stillwater, Oklahoma, 1987 to present.