DEVELOPMENT OF A GENERIC SIMULATION BASED CRITICAL RESOURCE SCHEDULER FOR BATCH MANUFACTURING ENVIRONMENTS

Thesis submitted in accordance with the requirements
of the University of Liverpool for the degree of
Doctor of Philosophy

by

SIMON F. HURLEY, M.A., B.A.

The Department of Industrial Studies
Faculty of Engineering

June 1992 - Liverpool, England.

To Birthdays and Christmases missed.

Acknowledgements

I would like to acknowledge Dr. John Driscoll for his supervision of this project.

A very special thank you to Mike without whose help this thesis could never have been finished. For providing money, equipment, time, and supervision, and for being my friend.

The author wishes to thank the staff and students of the Industrial Studies Department, Liverpool University and the Industrial and Systems Engineering Department, Virginia Tech, USA. Pete and Kev for there help and support at the very end. Jin for his philosophical discussions late into the night about this research and tolerancing. Janis for her friendship and support.

Laurence whose friendship is invaluable.

Angie B. for her encouragement. Bee for her sanity checks and Annie for proof reading.

Lastly I would like to thank M&D who never doubted that I would and could complete this Ph.D.

Abstract

Operational Research-based methods for scheduling of manufacturing systems have proved unsatisfactory when applied to actual manufacturing systems. This is because such methods become too cumbersome to use when they try to model all the variability inherent in a real factory.

A different, extensively-researched approach is to use dispatching rules to sequence jobs through a facility. The majority of research into dispatching rules tends not to account for any capacity constraints. A capacity constraint or "bottleneck" will be present in any manufacturing facility that is not in perfect balance. The motivation for this research is the premise that the bottleneck exerts a significant influence on the performance of the facility and therefore should be explicitly considered when making sequencing decisions. This research has first developed and implemented methods to locate the critical resource, and then developed a dispatching rule approach to job sequencing that uses the identified critical resource as the focus of sequencing decisions.

As a consequence of the combinational nature of job shop sequencing, simulation is normally employed to test the effectiveness of sequencing heuristics. To investigate the critical resource sequencing methods it is advantageous to create a simulator, thus avoiding being restricted by the functionality of particular simulation packages. The end user of this research is a scheduling manager responsible for determining the best sequencing method to employ given a known facility and potential product mix. To achieve this, extra functionality is required within a simulator (i.e., menu driven interface, flexibility in the modeling approach, and quick model creation and

editing). The second strand of this research is thus to develop the conceptual model of the simulator and then implement it incorporating the critical resource sequencing heuristics mentioned above.

The complete methodology is termed Manufacturing Simulation and Analysis (MSA) and is reported in this thesis.

List Of Contents

List of Figures

Figure Number

List of Tables

CHAPTER ONE

INTRODUCTION

1.1 Introduction

Globalisation of world markets, the development of advanced manufacturing techniques and technologies, higher labour and energy costs, the need for economies of scale and the pursuit of flexibility all led to a new phase of integrated automation beginning in the 1970's [Cleland and Bidanda (1990)].

In 1984 Harrington (1984) introduced the now commonly accepted umbrella term for automation "Computer Integrated Manufacturing" (CIM). The ideal of a CIM plant is one which is highly integrated, completely optimised and a strong competitive weapon. CIM however, represents a high investment manufacturing scenario that will remain restricted to a narrow band of manufacturers. For many years to come small batch manufacturing facilities will coexist along with CIM "Factories of the Future". A 1990 survey by NIST (National Institute of Standards and Technology) reported there were over 150,000 job shops in America each with less than fifty work centres.

Both the "Factory of The Future" and batch manufacturing face a similar problem of how to use the resources available to effectively and efficiently produce the products required. Baker (1974) defined this problem of scheduling as:

> "Scheduling is the allocation of resources over time to perform a collection of tasks"
> [Baker, pg2 (1974)].

Operations Research based investigations into solving the scheduling problem are extensive with many small variations to the models used. Examples include: variable batch sizes, set-up times, inclusion of processing costs, secondary routes, bi-criteria optimisation. Although the models used are restricted due to limits imposed by necessary assumptions, such models

are useful in predicting how certain rules and modes of operation may affect performance statistics when applied in the actual production facility. The most complex of the OR based models, the Generalised Job Shop, has been shown to be NP-complete, that is, it is highly unlikely that a solution method exists for finding an optimal solution. For the general job shop model one method exists for generating optimal solutions, however, the constraints on the model restrict its applicability.

Extensive research has been carried out into alternative means of sequencing job shops, these include branch and bound techniques, integer programming and dispatching rules.

A measure of success of research into job shop scheduling is the level of application in industry, the end users of such research. At the shop floor level, dispatching rules have had the widest application due to their relatively simple implementation.

Dispatching rules are structured techniques for selecting jobs from those in a queue waiting to be processed. Simple rules exist such as First-Come-First-Served (FCFS) where jobs are processed in the order they enter the queue. More complex rules reorder the queue as a new job arrives, for example, the Shortest Processing Time (SPT) rule. Rules may be merged together in an attempt to capture the advantage of both. As an example consider SPTx. Here, the SPT rule is applied until a job has been resident in a queue for more than a preset time then the Earliest Due Date (EDD) rule is applied.

Panwalker and Iskander (1978) reviewed research into dispatching rules listing over 100, from static/local rules to dynamic/global versions. The conclusion of the study was that researchers reported results which did not

have a common performance evaluation criteria or application methodology. When these two did coincide, the results reported where occasionally conflicting.

When investigating the performance of dispatching rules under various shop configurations, the methodology for launching the jobs to the facility tends to be "push" down to the shop floor. When on the shop floor, the jobs are "pushed" through the facility with no regard for the capacity limitations inherent in any unbalanced manufacturing facility.

The problem of scheduling and sequencing of jobs through a facility is complex and combinational in nature. When sequencing jobs through a facility, queuing times are indeterminate, hence the use of simulation to analyse the effectiveness of rules. The primary use of commercial simulation packages in manufacturing is to investigate the layout of existing or new facilities. A newer role for simulation is to investigate the effects of various sequencing decisions. Simulation packages to investigate sequencing methods use similar information to those used to investigate layouts, but have different functional requirements. They require powerful user interfaces focused on scheduling, quick model building capabilities and the ability to quickly change important data (eg. routing, processing times) along with the implementation of a variety of dispatching rules. Job shops come in many configurations and the modeler inside the simulation package should be generic, that is, it should have the ability to model the complex combinational nature of these facilities.

A problem with some research into sequencing methods has been that the resultant methods lacked applicability. Examples of this problem include requirements for information overhead and the coordination of many factors

required to make a decision (facility state and job launch order). Of this theory-practitioner gap Browne (1988) said:

> "consider scheduling as a combinational problem to be solved using mathematically elegant techniques accounts for the 'theory-practice' gap in scheduling."
> [Browne (1988)].

To further research into critical resource sequencing, a modeler has been developed aimed primarily at investigating sequencing heuristics.

This research proposes a methodology to explicitly recognise the presence of a capacity constraint and use it as the focus for both generating the launch order of jobs to the facility and the subsequent selection of jobs from input queues of idle resources.

This chapter begins with a discussion of two manufacturing planning and control frameworks. These are discussed in order to give the research a time frame context and a functional context within a manufacturing organisation.

1.2 A Structured Framework Approach

In designing the manufacturing model used in this research it is necessary to first develop a conceptual model or framework for a manufacturing facility. This will aid in the formulation of the boundaries of the research as well as giving a context in the wider system of a manufacturing company.

The use of frameworks in systems design effectively began with Anthony (1965) who outlined a framework for "Planning and Control". His was a framework comprised of three main decision categories:

1. Strategic Planning;
2. Management Control;
3. Operational Control;

and two supporting modules

4. Information Handling; and
5. Financial Accounting.

Although the work is dated, the framework is still relevant in todays manufacturing facilities that use hierarchical control. The framework indicates the need for: systems design, integration and aggregation, and disaggregation of information. The primary differences between the decision categories is the degree of data aggregation, the time frame for actualisation and realisation of results, and the degree of uncertainty which increases with the length of the planning horizon. Anthony recognised the limitations of his work admitting that the boundaries between classifications are indistinct and that situations exist that do not fit precisely into the definitions. He believed, however, that the classifications are applicable in the majority of cases and are a valid starting point for the development of more in-depth applied frameworks.

The most notable development of Anthony's work is the Hierarchical Production Planning (HPP) System [Hax (1984)]. Based on the same three level approach, the focus is a more dynamic time based framework and develops the principle of aggregation and disaggregation of data. This principle allows complex information to be formatted in such a way as to clarify the decision making process at each level.

The HPP model reflects the hierarchical nature of managerial decisions. Initially aggregate decisions are made which impose constraints within which

more detailed decisions are made at a lower level. In turn, these detailed decisions produce feedback to evaluate the quality of the aggregate decisions.

An indication of the functions performed at each level is given by considering Holstein's (1968) framework. Although over twenty years old, portions of the framework are still incorporated into MRP-II packages today. The elements of the framework are as follows:

1. **Forecasting** (strategic) - activity of analysing historical and presently available data to predict future product demand.
2. **Long Term Capacity Planning** (strategic) - process of ensuring the organisation is able to produce the forecasted demand, also the allocation of aggregate production to different facilities.
3. **Order Entry** (strategic) - primary input into the production control system.
4. **Inventory Planning and Control** (management) - the mechanism for ensuring the availability of materials that are required by the production process.
5. **Short Term Scheduling** (management) - the development of the detailed plan to meet the delivery commitment represented by the master schedule.
6. **Master Scheduling** (management) - determining the overall production plan for the next several months.
7. **Short Term Capacity Scheduling** (operational) - determine whether the short term schedule is feasible and identify potential bottlenecks.

8. **Dispatching, Releasing and Shop Floor Control** (operational) - the activity of executing the schedules developed earlier. This involves controlling the level of work-in-progress, job sequencing, job release and adjusting the schedule.

The role of the operational control module is at the bottom of the HPP hierarchy and requires the most information. Within the module itself the functions are also hierarchical in nature. For example, the *Master Scheduling* section has a time frame of several months whilst the *Short Term Scheduling* and *Short Term Capacity Planning* modules are making decisions on a minute by minute basis as to which job to start processing on an idle resource. It is with the subject of dispatching, releasing and shop floor control, that this research is focused on.

1.3 Short Term Shop Floor Control

Research in the area of the Strategic and Tactical levels of production control have yielded the now familiar MRP and MRP-II philosophies. Problems with MRP-II are in the lower operational areas, for example:

- capacity requirements are checked using the maxim, "aggregate lo ad should not exceed capacity in any time bucket".
- coarse synchronisation of materials flow. Taking averages over successive time buckets MRP-II assumes that there will be sufficient capacity.
- sufficient WIP to keep the shop floor busy and use expediting for late orders.

Buxey (1989) states that if lead times are not short and predictable then MRP-II may be untenable.

1.3.1 SHOP FLOOR CONTROL: DESIRABLE FEATURES

Within any shop floor control methodology several important features are desirable. These features include:

Low WIP Inventory

Minimising WIP is a major objective of newer production philosophies such as JIT and OPT. If a reduction in WIP inventory is implemented, this will lead to a lower manufacturing lead time, increased flexibility, and reduced risk of part deterioration.

Short Scheduling Scope

In MRP-II the scheduling scope is usually a week. In applications such as JIT it can be as short as a day. Under real time control the effective scheduling scope may be variable due to unforeseen events, for example machine breakdown causing the need to re-schedule.

Key Machine Utilisation

Higher machine utilisation rates will have to be achieved in the future to justify the higher costs involved in purchasing automated machinery, communication hardware/software and computational equipment. Due to unavoidable imbalances in machine capacities there is going to be a key critical (bottleneck) resource. It is important to identify key resources because:

> "an hour lost at a bottleneck is an hour lost for the whole system".
> [Goldratt (1984)]

Identifying key machines helps finite capacity scheduling become more precise and hence lead to a less congested shop as order release to the shop

floor is closely linked to the key machine capacity.

Finite Capacity Scheduling

Infinite loading and post processing the schedule through Rough Cut Capacity Planning (RCCP) software is cumbersome and not particularly precise [Buxey (1989)]. If the capacity requirement exceeds what is available, the load is redistributed by manipulating the lot sizes, re-routing, or moving orders to another time bucket. There are three major drawbacks to this method:

1. The capacity requirements are usually "rough cut", that is, imprecise due to aggregation.
2. To trim the load or move the loadings requires a priority schema.
3. The method requires significant computational time.

1.3.2 RESEARCH AREA

As previously discussed, strategic and management levels of the control hierarchy have been implemented within the MRP-II closed loop framework and software packages available today. MRP-II, however, does not successfully manage a shop floor where production requirements are not stable. It is in this area of shop floor control that this research is based.

1.4 Research Methodology

The focus of this research is in sequencing techniques for the general job shop. The techniques developed will use dispatching rules as part of a complete capacity sensitive heuristic.

To judge effectiveness of the heuristic, a simulator is required which is

focused at performing "what if?" sequencing analysis. Instead of developing a simulator to simply implement the rules, a research need was identified for the investigation of desirable properties of a simulator to be used exclusively for "what if?" sequencing analysis.

To develop a solution to this problem, three areas are of direct concern:

1. Modelling;
2. Simulation; and
3. Finite capacity sequencing heuristics.

The underlying principle was to develop and implement a generic modeling methodology capable of representing a variety of manufacturing facilities. The simulation driver uses this generic model to simulate the movement of jobs through the facility using a critical resource scheduling technique.

The next chapter reviews literature on the subject of sequencing and simulation, and chapters three, four and five discuss the "Manufacturing, Simulation and Analysis" (MSA) methodology.

Modeling

A generic approach to modeling, simulating and scheduling a manufacturing system leads to a lowest common denominator theory of manufacturing. This means selecting methods, information and goals that are both relevant and available in the group of facilities one wishes to consider. For the modeler, the basic element is the resource or work centre. For information basic files were chosen which are available in all manufacturing facilities, for example, Bill of Materials and routing sheets. To sequence and schedule the jobs through the facility, bottleneck scheduling was chosen. This is justified since virtually every facility will have a bottleneck unless it is in perfect balance.

Simulation

To make the model dynamic, a simulation model needed to be created. Using the simulation driver on the model, "what if" analysis can be carried out on scheduling alternatives.

Finite Capacity Sequencing Heuristics

The sequencing heuristic must take account of the critical resource and use this when launching and sequencing jobs. An important aspect of the heuristic which is often overlooked in research of this genre is application on the shop floor. The implementation of the heuristic must result in a simple to implement and maintain priority system.

1.4.1 THESIS OUTLINE

Chapter 2 reviews the relevant literature in the area of sequencing and the application of simulation to manufacturing control.

The three elements of the approach: modeling, simulation and sequencing have been incorporated into a consistent methodology called MSA (Manufacturing Simulation and Analysis). An overview of the MSA methodology is provided in Chapter 3 which is expanded upon in Chapters 4, 5 and 6.

To investigate the efficacy of the approach an extensive test suite was developed and over 12,000 weeks were simulated. The results from this analysis are reported in Chapter 7 along with the conclusions and recommendations. Chapter 8, the final chapter, discusses several interesting areas of future research for simulation and finite capacity sequencing.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

A job shop includes work centres (generally machine-tools), work tasks, (generally referred to as jobs). These two main parameters have associated data structures, e.g., process routes and both exert constraints on the job shop situation. Scheduling this job shop scenario has been extensively researched by a previous generation of academics; Panwalker and Iskander (1977), Salvador (1978) or Blackstone et al (1982). The basic OR research into production scheduling dates from the mid 1950's through to the late 1970's. Research after this time concentrates on extensions to the basic results. The first section of this chapter reports the basic results covering the four standard models used in OR production research. The second section is concerned with extending this research and hence the references are considerably more recent.

The volume of published work on scheduling job shops is considerable. The reason for this amount of research generated in this area is the integrated nature of the problem and the inherent difficulties in finding an optimal solution. To date optimal schedules have only been produced for small and specially formulated cases. The general job shop schedules are still generated from heuristics and, consequently, are sub-optimal.

Research into complexity theory [Coffman (1976)] classifies OR problems as P-time and NP-time (NP-complete). P-time problems are solvable by the use of an optimising algorithm. NP-time, the largest group of problems, describes a situation where it is "unlikely" that an optimising algorithm exists.

Consequently research efforts are directed towards finding efficient approximation techniques for generating feasible schedules. How close to

optimal is a schedule, is another question. Garey et al (1978) and Gonzalez and Sahni (1978) presented work on developing bounds of worst case behaviour.

The Sequencing Problem

To appreciate the size of this scheduling problem consider the following illustrating example:

> Five items are to be produced and each job must visit each of the five work centres, once and only once. The processing time of each job on each machine is known with certainty. The performance criteria applied is to minimise the total processing time. This sequencing problem appears straight forward but there are approximately 25 million sequences that could be evaluated.

The problem of discovering preferred sequences in the considerably larger and more complex setting of the typical factory is clearly more difficult than this simple example. The general sequencing problem could be stated as: given *n* jobs, with known processing times, which must visit all of *m* machines, determine the order in which the jobs must be serviced at each machine in order to optimise a given criteria.

In the real world of manufacturing, job arrival is dynamic in nature, according to a probability or purely random process. Processing times may be probabilistic and more than one route through a factory may exist.

2.2 Traditional Techniques For Operation Sequencing

The four basic models used in sequencing research are:

1. Single machine model;
2. Parallel machine model;

3. Flow Shop; and
4. Generalised Job Shop.

The single machine model involves the sequencing of *n* jobs through one machine. The parallel model is a generalisation of the single machine case, it consists of *n* jobs with *m* machines working in parallel. A job can visit any machine and must only visit one before its processing is complete. The flow shop model is a natural development of the parallel model, instead of *m* machines in parallel there are *m* machines in series and each job follows the same set route and visits each machine once and only once. The generalised job shop is the most realistic model were a job can take any route through a collection of available work centres. There is no specific gateway machine which always performs the first operation nor does there have to be one specific terminating operation.

Model Types

The above models can be further categorised into *static* or *dynamic* types. In a static model all the jobs to be scheduled through the facility are known at time T=0. With dynamic models jobs arrive at the facility according to some predefined process. The problems may also be categorised further as either *deterministic* or *probabilistic*. A deterministic problem is when processing times are known with certainty, or in the dynamic models jobs are to arrive in the future at known points in time. A probabilistic model is when arrivals and processing times are derived from a probability distribution.

It can be argued that the dynamic model more closely resembles practical scheduling problems raising the question why study the static case. If the variance in the processing time is insignificant when compared to the magnitude of the total processing time, then a static model, considered an easier model to study, would be sufficient.

OR based research and results can therefore be categorised by the model used. For individual models to be mathematically feasible, combinations of restrictive assumptions are normally considered as acceptable within each case. The most common assumptions are:

1. Each job uses each machine only once.
2. There are no alternative routes.
3. Set-up times may be sequence independent or dependent.
4. Setup times are sequence dependent.
5. All operation times are deterministic and known.
6. Machines never break down.
7. All machines are available, that is, functioning, manned and have the correct tools.
8. There is no batch splitting.
9. There are no duplicate machines.
10. Each job requires an operation at each machine.
11. There are no random arrivals into the system.
12. No subcontracting or overtime.
13. Pre-emption is not allowed (when processing of a part begins it continues until it is complete, without interruption).
14. If due dates exist then they are fixed.
15. Job routing is fixed and no alternative routes are allowed.
16. Processing times are fixed, that is, estimated time equals actual time.
17. Each machine can process only one job at any one time.
18. A given operation can be performed by only one type of machine.

19. No order cancellation.

20. Transportation times between machines is equal to 0.

21. Parts are not recycled due to a processing error.

Assumptions 1, 2, 6, 8 and 9 clearly limit the applicability of the OR based solutions. If assumptions 3 and 4 do not hold, then it would be a futile exercise to construct an optimal sequence. An appreciation of the behaviour of sequencing rules and the limitations of analytical research into facility sequencing can be gained by examining the static version of the basic OR problems.

2.2.1 STATIC SINGLE MACHINE MODEL

The importance of research into the single machine model is apparent when the results are incorporated into a larger problem. In multi-operation environments, identifying a "bottleneck" or "critical resource" and then scheduling this resource as a single machine forms the basis of the schedule of the entire manufacturing facility looks a good approach and is the method examined in the work of this thesis.

A typical static single resource model has *n* independent jobs available for processing at time T=0. The set-up times are assumed independent of the process sequence and incorporated into the processing time.

Two basic proofs of the static single machine model which are often quoted in the literature are:

1. To minimise the mean flow time, the shortest processing time dispatching rule is optimal, (proof: Baker page 18),

2. To minimise both the maximum job tardiness and the maximum job lateness, the earliest due date (EDD) dispatching rule is optimal (proof: Baker page 24).

Baker also proves which dispatching rule is optimal when minimising weighted mean flowtime, mean lateness, minimum job lateness and the number of tardy jobs.

2.2.2 STATIC PARALLEL MACHINE SHOP MODEL

Properties of the model are:

1. *m* machines working in parallel;
2. jobs consist of a unique operation;
3. any job can be loaded onto any machine;
4. no machine is left idle if jobs are waiting; and
5. each job must visit one and only one machine before processing is complete.

Research on this formulation has followed a similar theme to the single machine model. Processing times are assumed to be deterministic and the machines may or may not be identical, however, any job can be processed by any machine. If the machines are not identical the processing time of a job may differ between machines.

Results for this class of problem have limited applicability and have been obtained under restrictive assumptions. Baker (1974 Chapter 5) explains the basic techniques and results pertaining to the parallel machine models. Following is a summary of these results:

McNaughton's Algorithm

The algorithm is to minimise makespan, the time from order release to order completion, on *m* identical machines when preemption is allowed:

1. Set M = to the longest processing time.

2. Select a job to begin processing at time t=0.
3. Select any unscheduled job and schedule it as early as possible on the same machine. Continue this process as long as the machine will not be processing longer than M.
4. Select the next machine, time is t=0 and go to step 3.

The solution produced is optimal but is not unique. If preemption is not permitted then the LPT heuristic is used.

LPT Heuristic

No direct algorithm has been developed to optimise makespan when preemption is prohibited. An "effective" heuristic uses the Longest Processing Time to develop a schedule:

1. Ordering all the jobs with respect to the LPT rule.
2. Schedule the jobs in order, assigning jobs to the machine with the least amount of processing time.

The resultant schedules are not necessarily optimal when considering makespan.

Miscellaneous Heuristics

Other algorithms and heuristics mentioned by Baker are:

1. Minimise mean flowtime with parallel identical machines.
2. Minimise makespan for sequence dependent jobs with equal processing times.
3. Minimise makespan with two parallel machines and preemptible jobs.

Generating schedules for parallel machine models requires both allocation

and sequencing decisions. Analytic results have tended to focus on minimising makespan. First by allocating the jobs to the machines so that the longest time a resource is working is minimised. The problem then becomes a series of single machines which need to be sequenced to optimise a particular goal.

Procedures for scheduling parallel machine models to minimise makespan are generalisations of single machine results. To minimise weighted mean flowtime, however, these generalisations cannot be made. Rothkopt (1966) produced a dynamic programming formulation for an m-dimensional problem. Even for "moderate cases", however, the approach becomes computationally infeasible. Mortan and Dharan (1978), using the criterion mean flowtime, developed an algorithm producing optimal results with dependent jobs.

In the area of tardiness-oriented measures, little progress has been made. The algorithms that exist have particularly restrictive assumptions, see Root (1965).

Finding an optimal solution to sequencing n jobs through a set of parallel machines is especially difficult. For example, non-preemptive scheduling of n individual jobs is NP-complete for $m > 2$ [Lenstra and Rimnog Kan (1978)].

Extensions to the basic model such as pre-emption disciplines, performance criteria and the consideration of precedence constraints, are reported in the literature, (Hax page 288).

2.2.3 STATIC FLOW SHOP MODEL

The flow shop problem is the simplest multistage model to be considered in

the literature. The sequencing of the model, however, "turns out to be disappointingly difficult due to the inherent combinational difficulties", Hax p288.

An *m* machine flow shop is defined as:

1. Each job visits each machine (processing time may be set to zero).
2. Unidirectional flow of jobs (each follows the same route).
3. After the first, each operation has one predecessor and each operation before the last has one successor (linear precedence structure).

In the previous two static models, if a resource became idle and jobs were awaiting processing then they were immediately loaded and processing began. The major difference in flow shop type models is that inserted idle time may be advantageous in reaching an optimal schedule.

In the nx1 (*n* jobs, 1 machine) case there were n! possible sequencing permutations. In the nxm flow shop case there are $(n!)^m$ different sequencing permutations. The first step in finding an optimal solution then must be a reduction in the search area. Two important theorems exist, the proof of which can be found in Baker (1974) Chapter 6:

Theorem 1

When scheduling to minimise any regular measure of performance in the static deterministic flow shop, it is sufficient to consider only schedules in which the same job ordering is imposed on the first two machines.

Theorem 2

When scheduling to minimise makespan in the deterministic flow shop, it is sufficient to consider schedules in which the same job ordering is imposed on

machines 1 and 2, and the same job ordering on machines m-1 and m.

From the above theorems the following results hold:

1. In a two machine flow shop the optimal schedule with regard to a regular measure of performance is a permutation schedule.
2. In a three machine flow shop the makespan is minimised by a permutation schedule. (Johnson (1954)).

The second theorem above does not hold for other measures of performance besides makespan and mean flow time or when $m > 3$.

2.2.4 STATIC GENERAL JOB SHOP MODEL

The General Job Shop Model is defined by the following:

1. Two or more machines.
2. A job may begin processing at any machine (gateway machine) within the facility.
3. A job can complete processing and leave the shop at any machine.
4. A job can visit any machine as many times as it is required.

Assumption 4 is normally omitted and substituted by:

4. Each job must visit each machine once and not more than once.

The basic model does not allow preemptive sequencing of jobs. Attempting to generate optimal schedules for the general job shop sequencing problem is computationally difficult due the combinational nature of the problem. The only exception to this is the Jackson's (1956) algorithm which produces an optimal schedule to minimise makespan when $m = 2$.

If $m > 2$ or if a job has more than two operations, scheduling in the static job shop is NP-complete [Lenstra et al (1977)], indicating that no optimal algorithm can exist to generate a schedule.

An Integer Programming Formulation of job shop scheduling was developed by Manne (1960). There are computational difficulties involved in this formulation, for instance, a 5 job, 5 machine problem has 120 constraints and 50 integer variables. A 10 job, 10 machine problem has 999 constraints and 450 integer variables.

The Exhaustive Enumeration Approach involves the enumeration of all schedules to find the optimal one, as the name suggests. For large problems complete enumeration is obviously computationally prohibitive. Giffler and Thompson (1960) developed an algorithm for reducing the search area, by recognising and discarding schedules which contained idle time. Again, however, for $m > 5$ the problem is computationally large.

The key difficulty therefore with the general job shop is the computational effort needed to evaluate potential optimal solutions.

2.3 Dynamic Models

Dynamic models are characterised by having jobs arrive over time according to a probability distribution or at deterministic points in time in the future.

2.3.1 DYNAMIC SINGLE MACHINE MODEL

The single machine dynamic shop relaxes two assumptions of the static models. Firstly, all jobs are not available at time T = 0, their arrival is either

deterministic or stochastic and secondly, the processing times of the jobs are either deterministic or stochastic.

If job arrivals are deterministic then leaving a machine idle and waiting for a job may lead to a more optimal schedule. If job arrival is stochastic in nature this causes the mean flowtime variable to become a stochastic variable. For example, job i may be waiting in a queue when a higher priority job arrives, causing the selection of i to be delayed. The delay is probabilistic since more jobs may arrive at some unknown point in the future.

To minimise the mean flow time and thus reduce shop congestion in the dynamic probabilistic single machine model the same assumptions hold as in the static model: the SPT discipline is optimal (Conway et al [1967, chap 8]). As a machine finishes processing a job, the SPT discipline is applied to all jobs in the input queue.

The single machine scheduling problem has been extended in many directions. For example in Gavett (1978) the goal was to minimise the downtime due to setting up the machine. This is a relaxation of the "no machine breakdown" assumption. Another extension to the model is by Jacobs (1988) who developed a branch and bound algorithm for optimal lot sizing.

A comprehensive bibliography through to 1976 is given by Elmaghraby (1978). More recent references include Axsater (1984), and Hsu (1983), they extended the dynamic probabilistic model, introducing machine downtime and varying lot sizes, sequent dependent set-ups hence producing more applicable results.

The key point with the dynamic formulation is the impossibility of conceiving optimal solutions in all but the most restrictive cases.

2.3.2 DYNAMIC GENERAL JOB SHOP

This class of problem is considered the most significant of all because of its close resemblance to the realities of a manufacturing facility.

In this model processing times are known and jobs arrive over time according to a random or stochastic process. As in the dynamic static model, job arrival and processing times may be either deterministic or stochastic in nature. The general job shop model may be viewed as a network of queues, a job is selected from the queue, is processed by the work centre and then joins another queue or leaves the system, processing complete.

A consequence of all jobs not being known at time T=0 is that queuing times become probabilistic making it impossible to assign fixed start times to jobs. It is therefore impossible, prior to running a simulation of the model, to determine the effectiveness of the sequencing and scheduling rules employed.

The Analytical Approach (based on multiple and single machines) generally uses the Poisson distribution to generate job arrival times, and an exponential distribution to generate processing times. The "most powerful analytical result" to date is known as "Jackson's Decomposition Principle", Jackson (1957, 1963). It creates a situation where a network of *m* queues behave as m-independent individual machine queues, however, the results are only valid under the following restrictive assumptions:

1. Jobs arrive according to a Poisson process.
2. The routes of the jobs may be random.
3. Processing times are exponentially distributed.
4. The dispatching rule used does not depend on the route or processing time of any job, for example, the First Come First Served rule.

The result of this method is that the queues act independently as a collection of single machines. The methods used dispatching rules such as SPT or EDD to generate the sequence of jobs to be processed by the machines (previously discussed). In some cases the resultant schedule was optimal with respect to the performance criteria used to judge it.

The resulting schedules are not close to being optimal primarily because of the fourth assumption. In the general job shop however, considered as a whole problem, no dispatching rule is available to generate the optimal schedule. Research in this area then has tended to concentrate on developing variants of the above mentioned dispatching rules to select jobs from those available in front of a resource when it becomes idle. It is this research that is reported in the following section.

2.4 Dispatching Rules

A general job shop acts like a network of queues. Until the simulation is run, it is not possible to say with certainty when a job will arrive at a particular work centre. For this reason, it is not possible to build a complete schedule that dictates which work centre will process which job and at what time. In this context the sequencing of jobs is usually carried out by means of dispatching decisions: at the time a machine becomes idle a decision must be made regarding what it should work on next. These decisions are performed by

dispatching rules, and since it has been demonstrated that the results vary with respect to the rule used, it then makes sense to seek out the rules which produce good performance. Simulation has been the primary tool for investigating the efficiency of various dispatching rules and results from such research is reported in this section.

2.4.1 PERFORMANCE CRITERIA

Reported in this section are the performance measures which are used in the majority of simulation studies concerning the effectiveness of dispatching rules when they are applied to the sequencing of general job shop models. The following list is taken from the work of Fry et al (1988), the basic nomenclature used in the discussion of sequencing rules is:

N	the number of jobs completed
L_i	the lateness of job i
C_i	the completion time of job i
S	the set of jobs completed
NT	number of jobs completed after their due date
r_i	the arrival time of a job

Performance criteria are typically split into inventory and due-date based evaluation. These include:

Inventory Criteria

Mean Flowtime: The average time that the jobs spends in the system.

$$\bar{F} = \sum_{i=1}^{N} (C_i - r_i)/N$$

Mean Earliness: The average earliness of all jobs completed.

$$\bar{E} = \sum_{i=1}^{N} \min[0,-L_i]/N$$

Due Date

Mean Tardiness: The average tardiness of all jobs completed.

$$\bar{T} = \sum_{i=1}^{N} \max[0,L_i]/N$$

Percent Tardy: The percent of jobs completed after their due date.

$$\%T = \frac{NT}{N} * \frac{100}{1}$$

Maximum Tardiness: The maximum tardiness for all jobs completed.

$$T_{max} = \max[\ \max\ (0,L_i)\ ] \quad i \in S$$

Root Mean Squared Tardiness: The square root of the average tardiness squared for all jobs completed.

$$T_{rms} = \sum_{i=1}^{N} [\ \max(0,L_i)^2/N\ ]^{\frac{1}{2}}$$

Mean Absolute Lateness: Average deviation of job completion times around their due date.

$$\bar{L} = \sum_{i=1}^{N} |L_i|/N$$

With respect to manufacturing efficiency Baker (1974) indicated that the mean flowtime measure was an indication of the level of inventory if jobs can be shipped early. If this is not the case, then it is only a measure of WIP, not including finished goods inventory. The mean earliness criteria can be used to measure finished goods inventory. The above discussion is taken mainly from Baker (1974), Fry et al (1988) and Panwalker and Iskander (1977).

An increasingly important measure of shop performance in an era of "customer performance" through a total quality approach is how well due dates are met. Selecting the appropriate due-date criteria set can be a problem. For example, the root mean squared of the tardiness penalises systems which have only a few jobs [Dar-El and Wysk (1982)]. Mean absolute lateness is a measure of how well due dates are met regardless of whether jobs are early or late. This measure is of value in situations when it is important both not to produce too early because of storage costs until shipping date, or to produce the product late because of customer satisfaction concerns.

2.4.2 CATEGORISING DISPATCHING RULES

In the literature, dispatching rules are also known as scheduling rules, priority rules or heuristics. Dispatching rules are methods by which a queue of waiting jobs have priorities assigned to them. Gere (1966) attempted to categorise them as either priority rules, heuristics or scheduling rules. Priority rules are simple techniques to assign a rating to jobs in a queue, heuristics are "rules of thumb" methods and scheduling rules are combinations of dispatching rules.

Jackson (1957) categorised the rules as either static or dynamic, static rules, recall the FCFS rule, do not change the priorities of jobs with respect to time as is the case with dynamic rules, EDD for example, changes take place each time a new job arrives. Conway et al (1967) used the categories of local and global. Local rules only use information about the queue residing in front of the machine of interest, SPT for example, whereas global rules use information about other queues or the current state of other machines.

Panwalker and Iskander (1977) present a summary of over 100 such dispatching rules along with reference to corresponding analysis research papers. The classification schema used was as follows:

1(a). Simple Priority Rules.

1(b). Combinations of Simple Priority Rules.

1(c). Weighted Priority Indexes.

2. Heuristic Scheduling Rules.

3. Other Rules.

1(a) are the local rules mentioned above as well as those with limited global knowledge. 1(b) exist when the queues are partitioned or a second rule is used to break a tie. 1(c) are basically the same as 1(b) but they are combined using different weights. Category 2 rules are when the state of the facility is considered, for example, alternate routings. Category 3 is a miscellaneous section for rules designed primarily for specific facilities.

Dispatching rules most often used in research and industry have been listed by Fry et al (1988):

Static Local Rules

1. **FCFS:** The jobs are processed in the order they reached the machine, that is select the job with the minimum value of the ready time.

   $\min r_{ij}$

2. **SPT:** Select the job with the shortest imminent processing time.

   $\min p_{ij}$

3. **EDD:** Select job with the earliest due date.

   $\min d_i$

4. **BNO:** Smallest number of remaining operations in branch.

$\min o_{ib}$

5. **BMNO:** Largest number of operations remaining in branch.

$\max o_{ib}$

Dynamic Local Rules

6. **BS:** Branch slack.

$\min (d_i - t_{now} - Q_{ib})$

7. **BCR:** Branch critical ratio.

$\min (d_i - t_{now}/Q_{ib})$

8. **BS/NO:** Branch slack per number of remaining operation in branch.

$\min (d_i - t_{now}/Q_{ib})$

9. **BMDD:** Branch modified due date slack per number of remaining operations in branch.

$$\text{if } Q_{ib} + t_{now} \begin{cases} < d_i \text{ then } \min d_i \\ \geq d_i \text{ then } \min Q_{ib} + t_{now} \end{cases}$$

10. **NUB:** Number of uncompleted branches (SPT to break the ties).

$\min NUB$

Global Rules

11. **WINQ:** Work in next queue.

$\min (N_j)$

12. **Weighted WINQ:** Weighted WINQ.

$$\min \alpha(N_j) \text{ where } \begin{cases} \alpha=0.5 \text{ if machine i is a bottleneck} \\ \alpha=1 \end{cases}$$

The nomenclature given here will be used throughout the thesis.

2.4.3 RESULTS FROM SIMULATION STUDIES

The dynamic and stochastic nature of the sequencing problem make it difficult

to predict the outcome of sequencing rules. Simulation is the tool most often employed to perform such analysis of dispatching rules. In the review article of early basic research into dispatching rules by Panwalker and Iskander (1977), are listed 36 references dated from 1959 to 1974. The original goal of their research was to compare the results of the many studies and detail some definitive answers, however, this was described as a "formidable task" since the results from various studies tended to conflict. Further difficulties were encountered when researchers used different models, BoM structures, assumptions, operating procedures and performance criteria. Some generalisations could however be made, and these are the subject of the following discussion.

Results for flowtime

The most commonly used measure of congestion is flowtime. As mentioned earlier, this is only a valid measure of WIP if jobs arriving early have to be stored at the facility until the shipping date. Several studies Conway (1962), Dzielinski and Baker (1960) and Rochette et al (1976) found that the SPT rule minimises the mean flow time of jobs. An adaptation of the SPT rule, by combining the WINQ rule produced slightly better results than the SPT rule alone [Conway and Maxwell (1962)]. Practical application of such combination rules is limited because of the considerable work required to determine optimal weighting configurations.

The SPT rule reduces the mean flowtime measure by selecting the shortest operation jobs and speeding these through the facility at the expense of jobs with longer operation times. The disadvantage of such rules is that in a congested shop where all queues contain several jobs, a job with an unusually long operation time may never be processed. Conway and Maxwell (1962) investigated a method to alleviate this problem by using a truncated

SPT rule (TSPT) and relief SPT (RSPT). Under a TSPT regime, a maximum time a job is allowed to stay in a queue is set. After this time has elapsed, the job is given the highest priority. RSPT uses the FCFS rule until the queue size breaks a certain limit then SPT is applied. The results of simulation studies was that TSPT and RSPT rules are more effective than simple SPT when there were a few jobs from among many in a facility, which had relatively long processing times.

A myriad of adaptations to the SPT rule alone exist. Eilon et al (1975) derived SPTx, when if the slack time of any job is less than 0 then the one with the smallest is chosen, otherwise SPT is used. SPT-T reported by Oral and Maloum (1973) was a similar rule using a combination of slack time and processing time. These two rules were reported to be the most promising of the truncated SPT rules developed [Blackstone et al (1982)].

Results for Due Date

The attainment of good tardiness performance appears to be a more complex problem than the minimisation of the mean flowtime.

In the literature due date performance is usually measured as the proportion of tardy jobs or mean tardiness where tardiness is measured as positive lateness. Considering the different dispatching rules and their effectiveness at meeting due dates, a further complication is the method used to assign due dates.

Conway (1965) detailed four methods for setting due dates:

1. CON: job arrival time and constant manufacturing lead time (equivalent to a salesman setting a delivery date).

2. RDM: random (equivalent to due date set by customer).
3. TWK: proportional to total processing time (equivalent to shop management setting due date).
4. NOP: proportional to number of operations to complete the job (equivalent to shop management setting due date).

Along with the four methods for setting due dates the Conway study also detailed four dispatching rules often used in industry.

1. DDATE: highest priority job is the one with the most pressing due date. This rule is more commonly known as the EDD (Earliest Due Date) rule.
2. SLACK: the job selected has the smallest difference between remaining processing time and time remaining until due date (slack time)
3. OPNDD: the highest priority is given to the job with the earliest start date.
4. SLACK/OPN: highest priority is given to job with smallest ratio between slack time and the number remaining operations.

The performance measures used in the simulation study were: mean lateness and proportion of tardy jobs. No dispatching rule consistently out performed the others. When setting TWK due dates the SLACK/OPN rule performed the "best" but when due date was set by the RDM and CON methods, SLACK/OPN was out performed by the DDATE and FCFS rules.

An important point for this research is that the SPT discipline came close to out-performing each dispatching rule under each due date setting procedure. The SPT rule is regarded as a reliable alternative solution to the more complex due date based rules and is therefore applied in the work described later in

this thesis.

> "One can be confident that it (SPT) will reduce the fraction of tardy jobs, the mean lateness and the mean tardiness" (Conway et al [1967 p237]).

The only major problem with SPT is that it is insensitive to jobs with long processing times which causes the lateness variance to be high.

Results for the General Job Shop

It has been demonstrated that the various sequencing rules do not generalise to the multi-stage job shop [Goodwin and Goodwin (1982)]. Research involving the general job shop is limited [Goodwin and Goodwin (1982), Huang (1984), Russell and Taylor (1985)].

Russell and Taylor investigated the performance of various dispatching rules on two different BoM structures, the first set is tall with four levels and the other set of BoM's are flat with three levels. The conclusion of the study was that there existed a direct correlation between the depth of the BoM structure and the due date performance, the taller the BoM the higher the probability that the jobs will be late.

An assumption across the majority of research into the general job shop is that of balanced capacity across all machines. In the typical job shop long queues of work may disguise any capacity imbalances since all work centres will be working all the time. The existence of large buffers in front of machines creates many problems such as long lead times, poor due date performance, huge inventory costs and difficulty in tracking quality problems. Consequently large buffers are considered undesirable and are symptomatic of the desire for "efficient" production as opposed to "effective" production.

With respect to research into the unbalanced facility, Fry el al (1988) developed a model of a multi stage job shop with five BoM's ranging from flat to tall. In this model, the bottleneck resource, a feature of any unbalanced job shop, was fixed by assigning it a lower capacity than the other machines. The results of the study support the results of Goodwin and Weeks (1986): or, that the due date based rules where the "overall better performers". Fry et al also stated that no one rule was an overall best performer, a point examined in the test programme described later.

Research such as Fry's tend to have a bottleneck resource fixed in one location. As the product mix changes and the quantity demanded varies, it is possible that other machines become the bottleneck. A significant new approach therefore is to research into finding the bottleneck for changing production scenarios and effective sequencing rules when the bottleneck has been located. This would complement and extend the work of Fry et al.

In summary, the simulation approach has proved an appropriate means to evaluate the efficiency of various dispatching rules from the simple (FCFS) to the complex (WWINQ). The conclusions drawn are that the simple SPT rule is the most effective if small adaptations are made to it to avoid jobs with long processing times from being held back. The EDD (or DDATE) rule is an effective method of sequencing jobs through a facility when due date performance is of high importance.

With respect to research into the general job shop, where capacities are not balanced, results have shown due date based measures are most effective. The models used in the research described fix the location of the bottleneck by artificial means to study the effects of dispatching rules. This makes the results somewhat artificial because bottlenecks are caused by product mix

and quantities demanded and move as these parameters vary and may change period by period. As just indicated however, the previous research has indicated some significant useful directions.

2.5 Capacity Sensitive Sequencing Heuristics

Panwalker and Iskander's classification of rules for sequencing jobs defined Heuristic Scheduling Rules to be "ones that involve a more complex consideration such as anticipated machine loading, the effect of alternate routes, scheduling alternate operation, etc." Their study is considerably dated, hence did not cover the more modern methods employed today (e.g. capacity sensitive heuristics embedded in Just-In-Time manufacturing).

Wild (1977) considered the management of capacity as the fundamental problem of Production Management. Buxey (1989) noted that:

> "Production schedules ... cannot be divorced from capacity planning since any worthwhile schedule must at least be feasible."

The above references from two noted authors indicate that at some time within the scheduling cycle capacity must be explicitly considered. If the schedule is based only upon predetermined average times as in MRP then it will only be an approximate guide. A schedule should be based upon forward loading to finite capacity accounting for waiting times in queues [Buxey (1988)].

The management of capacity is important to an organisation because:

1. Sufficient capacity is required to provide the output for the current and future customer demand.
2. It directly influences the efficiency (cost) of operations.

3. It represents a sizable investment by the organisation.

Three modern heuristics are discussed in this section, the first, Load-Level Order Release is an academic model. The other two approaches are the well known and often written about, JIT and OPT (Optimised Production Technology). Load-Level and JIT deal with capacity limitations as they are encountered, OPT expressly recognises them and uses them in developing schedules.

2.5.1 LOAD LIMITED ORDER RELEASE

Becthe and Ketter developed the "Load Level" approach to controlling the release of orders to the shop floor at the Institut Fur Fabrikanlagen of the Universitat Hanover, West Germany [Becthe (1982)]. Two leading papers describing the approach split views with Brechte (1988) in favour of the approach and Kanet (1988) is much more critical.

The Objective

"The idea behind load oriented manufacturing is to limit and balance WIP inventory at a level as low as possible in order to accomplish a high work centre utilisation as well as a rapid and in-time flow of orders", Brethe (1988).

Load Limited Order Release controls the release of orders to the shop floor in the following manner. When the inventory level at any work centre exceeds some critical level (its load level) further release of orders which are routed to that machine are blocked from entering the shop.

For example, assume five machines, A, B, C, D and E and two jobs with independent demand functions and follow different routes.

JOB 1 follows route A to B to E,

JOB 2 follows route C to D to E.

Work centres A and C are regarded as the gateway machines and E may be viewed as a final assembly/packing operation. If the WIP buffer at work centre E reaches its predefined "Load Level", then further release of shop orders to work centres A and C (the gateways) will be blocked until the WIP level at E is reduced to less than the allocated load level. Any work packets already being processed by the shop (machines A, B, C or D) are allowed to continue, as well as any jobs already in the queues. If the load level was exceeded at work centre B further input to A (job 1) would be throttled whereas C would continue to accept jobs (job 2) as per normal.

Critical Review

The load level approach is similar to JIT in the way it controls production flow. In JIT, if a machine reaches its Load Level then production from the next upstream work centre is stopped. The Load Level approach may be more efficient, with respect to WIP inventory since work is prevented from entering the shop if any machine downstream breaches its assigned Load Level. If the breach is caused only by an easily resolved processing problem, however, then throttling and restarting input could have logistical problems. It would appear reasonable that to judge the effectiveness of the Load Level approach, it should be tested against the JIT method of loading a manufacturing facility. The author is unaware of any such comparative studies having being done.

This apparent shortcoming is an example of the theory-practitioner gap in production scheduling theory. To operate Load Level scheduling, the WIP content of each work centre needs constant monitoring, a requirement not necessarily easy to achieve in a working job shop. Furthermore,

communication has to be established with the gateway work centres and shop order release functions to throttle the input of certain work packets. As in JIT, load level order release will be most effective in a shop which is close to having balanced capacity.

Due dates would become difficult to predict because of uncertainty whether material input is about to be throttled. A product which has two gateway machines may encounter problems if one machine is blocked.

The major finding of Kanet(1988) is that system flow time, inventory and job tardiness all deteriorate because the Load Level approach introduces idle time into a production schedule.

A proposed method of bypassing these problems may be to focus on the bottleneck resource by applying load levelling techniques there. If the Load Level is exceeded then input to the shop is stopped. A temporary Load Level breach elsewhere in the shop would not interfere with the flow and should not throttle input to the shop.

2.5.2 OPT (Optimised Production Technology)

The developers of the OPT approach detail the philosophy in two books **The Goal** and **The Race** [Goldratt and Fox (1984) (1986) respectively].

"The Goal" narrates the logical development of OPT by a plant manager given three months to turn around a loss-making factory.

The OPT philosophy is viewed with skepticism within the academic community as the mathematical scheduling algorithms are propriety

knowledge of Creative Output, the software vendors of the product. This lack of explicit formulae make it impossible to test the validity of the results and the fictional style of **The Goal** only serves to heighten this skepticism. In the less pragmatic and more applied world of industry, OPT had to prove itself by working, the internal formulae being less important. A number of Fortune 500 companies have bought and implemented the software, (for example, Ford, General Electric (USA), Caterpillar and Perkins Diesels). The effectiveness of OPT may eventually be judged by the number of these companies that continue to use the technique.

The author supports Jacob's opinion (1983, 1984) that the OPT approach is actually an amalgamation of unreferenced earlier work. For example, Goldratt stresses the fundamental importance to the scheduling approach of the bottleneck or critical resource. The importance of bottlenecks had already been pointed out in 1974 by both Bellafatto (1974) and by Baker (1974).

The Nine Rules of OPT

Fundamental to the OPT approach are nine rules which, according to Lundrigan (1987): "encourage us to look at the manufacturing world in a new way". The Nine Rules:

1. Balance the flow, not the capacity.
2. Constraints determine Non-Bottleneck utilisation.
3. Activation is not always equal to utilisation.
4. An hour lost at a bottleneck is an hour lost to the entire system.
5. An hour saved at a Non-Bottleneck is a mirage.
6. Bottlenecks govern throughput and inventory.
7. Transfer batch should not always equal process batch.
8. Process batch should be variable, not fixed.

9. Set the schedule by examining all the constraints simultaneously.

Full explanations of the rules are contained in the article by Lundrigan (1987).

The first of these rules forms the basis of the approach, namely the OPT assumption that imbalanced capacity is a characteristic of most manufacturing facilities and the focus should therefore be on balancing flow not capacity.

The Methodology

The OPT approach is both a philosophy and a software product sold by Creative Output Limited. **The Goal** detailed the successful implementation of the philosophy without the software.

In companies where MRP systems are already implemented the transition to OPT is easier because it uses existing MRP data files. Additionally the organisation as a whole would be used to the idea of computerised control.

The process begins with the creation of a "product network" which is an amalgamation of a BoM file and routing data. The "resource description" file includes, for example, machine identities, hours available, set-up times and complete descriptions of each resource within the manufacturing facility.

The BUILDNET module organises the product and resource information into a network. Writers on OPT are unsure of how exactly the data is formatted in the network, Lundrigan (1987) never mentions the module and Jacob only says:

> "... using BUILDNET to form an engineering network."

BUILDNET appears to build a map for each product and identifies the processing required at each machine.

The network is passed through a module called SERVE, which is an infinite capacity backward scheduler the same as in MRP. SERVE not only generates capacity requirements but also flags the bottlenecks.

At this point OPT differs from MRP. If there was overloading identified under MRP-like logic, the Master Production Schedule (MPS) would be regenerated and passed back through the CRP module. In OPT, the SPLIT module is applied to the "engineering network" to divide it in two: one section contains the identified bottleneck and all upstream resources (OPT network) and the other section of the network contains only non-critical resources (Serve network).

The OPT network is then forward scheduled from the critical resource. The only capacity which needs to be taken into account is the bottleneck, all other resources can be scheduled to infinite capacity. The Serve network is backward scheduled from the bottleneck to generate the release dates.

Discussion

Most production engineers and researchers are familiar with MRP. For this reason this section when discussing the advantages and disadvantages of OPT, will use MRP as a comparator.

Lot Size Rules: OPT dynamically calculates lot sizes when scheduling. It views the set-up cost at the bottleneck resource as more expensive than other non-bottleneck machines. This originates from the approach taken by OPT that when the bottleneck

machine is not working then throughput for the whole factory is being lost.

Queue Time: In some cases, 90% of the lead time of a part is spent queuing, so reducing transfer batch sizes will not have a significant effect.

When generating theoretical lead times MRP uses fixed (average) queuing times.

OPT, on the other hand, looks at all open demand for capacity at work centres before calculating the lead time.

Swann (1986) believes that the MRP approach is most useful in situations where the MPS is frozen for long periods of time and there are no material shortages, no expedite situations, no quality problems, no machine breakdowns, etc.

2.5.3 JIT (Just-In-Time)

The third capacity sensitive production planning methodology reviewed is Just-in-Time (JIT). The Just-In-Time production philosophy was developed at the Toyota Motor Company (Japan) during the 1970's and applies primarily to repetitive manufacturing. It does not necessarily require large volumes to be produced, but it is restricted to a group of parts that contain a high proportion of repeating products.

The objective of JIT production is the eradication of all waste. Producing one too many is just as serious as producing one too little. This is the opposite of

the traditional viewpoint where requirements are exceeded "just-in-case".

The manufacturing process may be viewed as a series of queues, to eliminate waste, JIT focuses on the queues facing machines and tries to minimise them. By minimising the contents of queues, the result is:

1. minimises WIP;
2. shortens lead times;
3. speeds up reaction times; and
4. uncovers quality problems.

Since WIP inventory is pushed down to a bare minimum there has to be a strong emphasis on quality, preventive maintenance and honouring delivery commitments.

In reality JIT is a collection of logical concepts:

1. uniform plant loading;
2. group technology;
3. total quality control (TQC);
4. total preventive maintenance (TPM);
5. multi-function workforce;
6. JIT delivery;
7. set-up time reduction; and
8. kanban production scheduling.

Schonberger (1984, 1986)

Of interest in this review is the kanban scheduling methodology.

Kanban Scheduling

Tersine (1985) described the JIT scheduling system:

> "Kanban type production control systems are simple, self regulatory, systems for scheduling and shop flow control".

In JIT only the final assembly schedule is generated which is analogous to the Master Production Schedule in MRP. Following are the five steps to the operation of the Kanban system:

1. A production schedule is generated for the final assembly station.
2. Final assembly draws parts in small quantities from the work centres which feed final assembly.
3. The work centres feeding final assembly produce parts to replenish those withdrawn.
4. To produce the parts they must withdraw small quantities of parts from the upstream stations which feed them.
5. This continues upstream and results in the entire network being engaged in production synchronised to final assembly.

(Source: Tersine p573)

To formalise and regulate the replacing of stock which is withdrawn, two "kanban" cards are used: a production Kanban and a conveyance Kanban.

The system may be described as "pull" since the final assembly schedule pulls the parts through the factory to meet demand. Parts are only produced when a production kanban is replaced by a conveyance kanban on a container. No "just-in-case" production is allowed, being regarded as waste. The number of kanban cards allowed regulates the level of WIP inventory within the system. When implementing Kanban production scheduling the number of cards allowed may be set relatively high. This would build in a certain amount of slack in order to overcome any "bugs" in the flow of parts. As the system

becomes familiar to the workers and stabilised, the number of cards can be reduced which in turn decreases the level of WIP inventory.

The number of cards is typically determined by the formula:

$$n = D L (1 + I) / C$$

where:

n	number of cards	I	safety factor
D	average demand	C	transfer container size
L	product lead time		

The formula, however, is only valid when set-up times have been reduced to a relatively negligible proportion of the processing time. In Ohno's (1982) discussion of the use of the formula for setting the number of kanbans, he said that it is only used as a rough guide with ultimate responsibility being with the line supervisors who adjust the number of cards to suit changing situations.

Discussion

The Toyota Production System [Schonberger (1986)] is regarded as the bench mark against which to judge all other implementations of JIT. A strong anti-JIT article by Wilson (1985) indicated that the reasons for Toyota's successful implementation of the philosophy were:

- standardised product line;
- suppliers close to home base; and
- unusually cooperative work force.

The article indicated that the standardised product line was "crucial" in order to have the product schedule frozen for at least two months ahead, adding

that in contrast, American automobile manufacturers have moved in the opposite direction designing computerised planning systems in order to supply a vast number of options to the customer.

Wilson's comments are not backed up by present practice, JIT is being implemented in selected plants by Ford and General Motors, the two largest automobile manufacturers in America. JIT however requires a certain amount of standardisation of products since changing the product mix means that suppliers will have to manufacture and deliver to meet the change. Ohno (1982) pointed to two features which distinguished JIT from other production control philosophies. Firstly, JIT emphasises shop floor control whilst MRP concentrates on inventory control, and secondly, systems such as MRP only have a scheduling/planning function. JIT on the other hand has both a planning and a controlling mechanism. MRP produces a schedule and the parts are "pushed out" onto the shop floor control is then the responsibility of the foreman. JIT on the other hand has a formalised control system (Kanban), to implement the generated schedules.

In all but dispatching and operational control, JIT is similar to other approaches. It has an annual production schedule and a monthly master production schedule.

To operate successfully, a JIT production system product control must exploit the intelligence and decision-making capabilities of supervisors and line operators [Schonberger (1983)]. To implement JIT successfully Cleland and Bidanda (1990) noted that there are "very stringent prerequisites":

1. stable demand;
2. smooth production;
3. highly reliable equipment;
4. 100 percent quality; and
5. small set-up times.

Potential users of JIT were advised in the paper:

> "If it is not possible for one reason or another to meet all of these requirements, then JIT should not be attempted"

This point is not in general backed up by other authors on JIT. Hurley (1987) in a case study on the partial implementation of JIT at a paint factory with fixed batch sizes of 10,000 litres. In this application JIT scheduling meant having a tighter control on the production line so that lead times where known with more certainty and safety stocks could be reduced to a minimum level.

2.5.4 BOTTLENECK ANALYSIS

For the majority of manufacturing facilities, unless in perfect balance, some capacity constraining resources or resource set is going to be present. Goldratt (1984) defined a bottleneck as:

> "any machine whose capacity is equal to or less than the demand placed upon it".

This definition is limited in scope in two ways, firstly, it is only relevant to work centres within a manufacturing facility and secondly, only resources which are more than fully loaded are technically regarded as a bottleneck. Burbidge (1987) sought to widen the definition and suggested the following:

> "the work centre or management function which limits the manufacturing output from a factory".

The definition is wider allowing for example, a Purchasing Department to be labeled as the bottleneck because it restricts manufacturing throughput by not ensuring timely delivery of raw materials.

This research views a bottleneck as the resource which is going to restrict the flow of parts through a facility. Even in an underload situation there may be advantages in finding the most heavily loaded resource and use it as a focus for generating efficient sequencing algorithms.

Capacity analysis under MRP assumes that capacity is adequate if the average utilisation is under 100%. A criticism of this is that if the flow is not balanced, large queues will result, which tend to create higher lead times and WIP inventory levels.

In a notable article on the subject of capacity analysis Karamarkar et al (1988) demonstrate that product throughput is limited by excessive queuing delays. The authors make the statements:

> "An alternative view of a bottleneck is any work centre or machine with a substantial queue".
>
> "This view of bottlenecks makes it clear that in addition to there being more than one, the bottleneck can be affected by batching and shift policies as well as by machine capacity, loading and product mix".

This research investigates shop loading policies in the short-term environment. Shift patterns and machine capacity are assumed to be fixed and can be removed from influencing the results.

Batching had a significant effect on both lead times and WIP inventory. Hence, if one is using queuing statistics to identify bottlenecks, that is, looking at the trade-off between capacity utilisation and WIP, the effect of

batching policies must be included in the analysis.

In Chapter 4 methods of identifying the bottleneck(s) are described and they include capacity analysis (Goldratt's definition) and analysis of simulated queues.

There are researchers, however, who believe finite capacity scheduling is not a feasible solution:

> "finite loading is an interactive approach that requires a lot of computing time," [Hendry (1989)].

Hendry's statement does not account for the speed and amount of computing power available and that reducing the search area by proactively identifying the bottlenecks would reduce the amount of required computations.

2.5.5 CONCLUDING REMARKS ON APPROACHES TO PRODUCTION SEQUENCING

Several important points need to be drawn out of the preceding discussion:

- For only the most simplistic models do techniques exist to generate optimal schedules.
- For the generalised job shop, dispatching rules are an effective method job sequencing.
- In a dynamic job shop environment, queuing times are stochastic hence simulation is an ideal tool for investigating the application of dispatching to such models.
- Of the many forms of dispatching rules investigated SPT and EDD are two of the "best". They are effective not only at producing good results with respect to performance measures, but also in the simplicity of the application; only knowledge of the one particular work centre of interest is required.
- Of the capacity sensitive techniques for sequencing jobs: the Load Level approach is viewed as difficult to implement because of the information requirements; OPT uses back-scheduling to locate the

bottleneck which may move when jobs are forward loaded through the facility; and JIT is most effective in specific environments which have stable Master Production Schedules as well as taking a considerable amount of time and reorganisation of the facility to achieve the full benefits.

- Views of what is a bottleneck differ between researchers in the area. The majority regard loading as the primary measure but some researchers argue that queuing statistics may be used to better judge the location of the bottleneck or critical resource.

Identified is a need to investigate both methods for locating the critical resource and the techniques for using this resource as a focus for making sequencing decisions.

Dispatching rules will be used in the ordering of queues at the critical resource. Further investigation must be performed as to the effect and effectiveness of dispatching rules when they are part of a critical resource sequencing heuristic.

To investigate the critical resource sequencing heuristic, a simulation modeler will be built which will be focused on accessing the performance of various sequencing methodologies.

A final point to note is that however sophisticated the critical resource sequencing heuristic (which incorporates the use of dispatching rules) is inside a simulation modeler the application of the results onto the shop floor must be in the form of a simple to use priority schema.

2.6 Simulation

This review covers the application of simulation in manufacturing research.

> "In its broadest sense computer simulation is the process of designing a mathematical-logical model of a real system and experimenting with this model on a computer," [Pritsker (1979)].

This definition implies that simulation is comprised of two parts: model construction and model experimentation determining the properties of the real system.

> "Simulation is one of the most widely used techniques in operations research and management science, and by all indications its popularity is on the increase." [Law and Kelton p2 (1981)].

The popularity of simulation derives largely from the possible 1-1 correlation that can exist between the model and some aspects of the system being modeled. In pure mathematical models, such a close identification is not normally possible.

2.6.1 APPLYING DISCRETE EVENT SIMULATION TO MANUFACTURING

Discrete event simulation is a computer modeling technique used to model the event orientated behaviour of systems over a simulated time period. When an event occurs it changes the state of the model and may add, change or remove a future event to the event list. The simulation timer jumps from one event to the next on the assumption that the state of the model does not change between times.

At a high level of abstraction, manufacturing systems may be viewed as a queuing network. Discrete event simulation is used extensively to analyse the

queuing properties of manufacturing systems [Pritsker (1988)]. At this level, resources (work centres for example) are modeled as servers or transformation processes. Customers (parts to be processed) enter an entity at some predetermined time (arrival event) and leave at some predetermined point in time (departure event).

Lowering the level of abstraction complicates the model by involving more servers and, hence, more attributes or variables. At a lower level, a work cell could contain a number of robots, milling or guiding machines. The lowest level of abstraction in modeling manufacturing systems would have to model entities such as cutters or sensors along with their related set of attributes.

The original role of simulation in manufacturing was to support the justification of new manufacturing facilities and to evaluate resource requirements or equipment needs. The new role for simulation is shorter term, aiding decisions such as equipment scheduling, shop order release, and work order scheduling, i.e. to fulfill a role as a part of the actual manufacturing system.

Simulation permits sequencing methods to be tested before being applied in production. Management can test the relative benefits of different priority rules when it is infeasible to experiment with the real system. Simulation can test in a few minutes what could take years to test on the real system. A large number of methods can be tested, fairly, and inexpensively without disturbing actual operations. Analytical solutions to scheduling problems in job shops have been of limited use with the most applicable results coming from simulation research, [Tersine (1985)].

The database of information for the design stage of a production facility in most respects is the same as that required for analysing production

schedules via simulation.

Simulation in production planning requires less detailed modeling of the entities then in layout experimentation but require more detail in the representation of the decision processes that occur on the factory floor.

In a typical design oriented simulation model, the interface is normally language-based, the creator of the model being also the end user. Production schedulers tend to be experts in their own field and not in simulation technology. A user friendly interface must be provided to first, assist the scheduler to build a model of the facility and secondly, to provide simple processes to input data and carry out significant "what if" analysis.

The reports generated should be aimed at schedule performance of specific jobs and work centres as well as overall performance of the facility. The software should contain an implemented set of algorithms from which the scheduler can select to test various scheduling scenarios.

2.7 Conclusions

Extensive research has been performed in the area of generating optimal schedules for limited sized problems. These restricted models are useful in indicating the methods which may be successful for application in general job shops. These results do not necessarily generalise to multi-resource job shop type models.

Research has investigated sequencing rules for job shops and indications are that due date based rules perform well against due date criteria but also perform well against flowtime criteria.

Research into use of finite capacity sequencing research has been carried out using artificially created bottlenecks. Critical resources however, move as the product mix, batch sizes and demand quantities change. Research needs are indicated in the area of first locating the Critical Resource and then using this as the focus for generating sequencing rules.

In developing a new scheduling and modeling approach, however, the end user of the research must be kept in mind. Techniques developed to identify the Critical Resource can be sophisticated, but the actual priority setting sequencing rule needs to be relatively simple to apply so as not to cause more confusion on the shop floor than already exists.

CHAPTER THREE

OVERVIEW OF A NEW CRITICAL RESOURCE MODEL

3.1 Introduction to a New Scheduling Methodology

Within this chapter an expanded view of the Manufacturing Simulation and Analysis (MSA) approach is presented. The methodology implements the following features in sequencing jobs through a general job shop:

- techniques for identifying a single critical resource; and
- the use of the identified critical resource as the key to assigning work;

An integral part of the MSA modeler is the simulator. The primary objectives of the simulator are:

- provide a simulation modeler focused at supporting investigation into critical resource sequencing heuristics;
- implement a modeling technique designed to assist sequencing rule investigation; and
- create user orientated interface that removes the need for programming.

The conceptual model of the MSA methodology is illustrated in Figure 3.1. The model is comprised of seven distinct modules (msa 1-7), six of which (msa 1-6) are used in each planning and sequencing cycle. The level of use of each module is dependent on the needs of the scheduling manager.

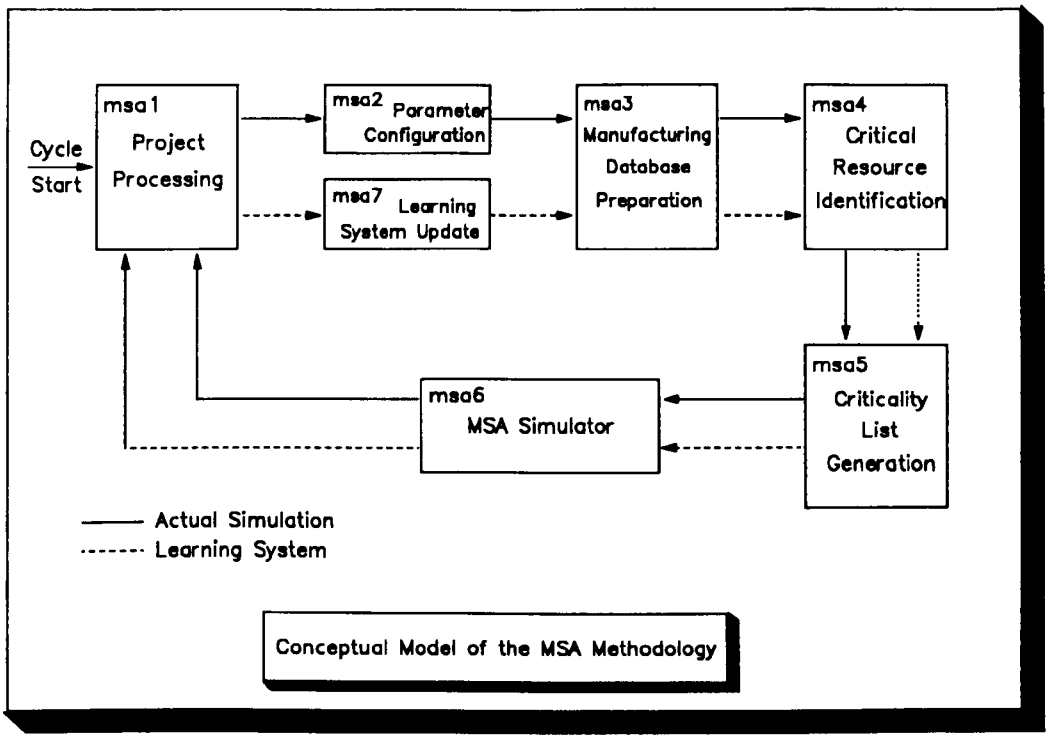


Figure 3.1: The Conceptual Model of the MSA Philosophy

Module msa1 creates a *project*: a set of data files that define the facility of interest. Module msa2 allows the user to configure the simulation parameters and msa3 maintains the databases on which the planning system operates. Module msa4 is a decision making module covering the identification of the Critical Resource. The current version of the methodology includes five alternative Critical Resource Identification models each of which is examined in the test programme chapter later in this work.

Module msa5 generates the Criticality List, an ordered list of the jobs that are to pass through the Critical Resource. The list, used as part of the scheduling heuristic, will determine which available job to load onto an idle resource.

To investigate the effectiveness of the sequencing methodology a

manufacturing system simulator was developed, module msa6. This simulator is invoked periodically to support decision making or to update the databases. Its use will be frequent if realistic control of the methodology is to be maintained over a period of time.

The function of module msa7 is to update the Learning System Knowledge Base for all the periods selected by the user. The Learning System is included for future work and is not investigated in the test case analysis reported in Chapter 7. The reason for this was that sufficient CPU time was not available to run a representative set of tests. It was developed concurrently with the main program due to the similarities in methodologies. Discussion of the Learning System is included both for completeness and to assist any researcher who may wish to use this work in the future.

The objective of this chapter is to:

- demonstrate the integrated nature of the methodology; and
- illustrate that the conceptual model is supported by a consistent methodology.

3.1.1 THE PHILOSOPHY OF THE MSA METHODOLOGY

As mentioned in the conclusion of the previous chapter the goal of this research is to develop and implement a generically applicable manufacturing modeler/simulator, to investigate new critical resource sequencing techniques. The objective of generic applicability pervades the entire process of development and implementation. This is illustrated below in the overview discussion of the essential elements of the methodology.

The Database

In developing the data requirements an adaptation of the 80:20 rule from

inventory control has been applied wherein eighty percent of the benefit is derived from twenty percent of the data. This simplifies the modeling process while still generating realistic and useful results. Consistent with the goal of generic applicability, the selected information must be present in all the facilities to be modeled. The data is stored in four files:

1. Integrated Bill of Materials;
2. Jobs Pending File;
3. Resource Definitions; and
4. Calendar.

When the simulation is executed the Jobs Pending and the Integrated Bill of Materials files are integrated into the Sequencing Database. The contents and structure of this database is explained later in this Chapter 5.

The **Integrated Bill Of Materials** (BoM) database follows closely the I.Prod.E. [Corke (1988)] specification of a Bill of Material file. Included within the file is processing time, routing and set-up time information about components, sub-assemblies, and end items. When the user defines a BoM for a product records are automatically created in the "Sequencing Database". These records are essentially the BoM database reformatted to allow the simulation driver software to quickly and easily access and extract data.

The **Jobs Pending** database contains the complete order book for the facility being modeled; from this a period-by-period order list is generated. Data held in the jobs pending database includes the due date, job number and the status of the job (scheduled, not scheduled, or blocked).

The **Resource Definitions** file holds all the data on the work centres within the facility being investigated.

The **Calendar** file contains information on the number of hours available per day for a period of 200 weeks. Throughout the simulation the availability of standard hours is determined by this calendar.

Applications of the files are illustrated in the next chapter and specifications of the files are included in Appendix B.

Restricting the data set to the above four files achieved the following:

1. Adherence to the generic philosophy;
2. Allows an easier understanding of the MSA methodology and the software;
3. The avoidance of tedious data input; and
4. Simplification of the modeling process.

Sequencing

No algorithm has been developed for generating optimal sequences of jobs for the general job shop model (see Chapter 2). There is also a large number of possible schedules available in even the smallest scenarios. To address these problems an attempt is made to reduce the search area to a reasonable size, one which contains a "good" feasible schedule. In this research the concentration will be on the bottleneck, or "critical resource", as the focus for generating the schedule. It is intuitively obvious that if a facility is not in perfect balance, it will have a resource which has a critical influence on the flow of parts. Even if a facility is not fully loaded, a critical resource will exist wherein if an hour of production time on that machine is lost an hour will be added to the total throughput time.

The critical resource thus influences the schedule to such an extent that it

becomes the focus of a new sequencing heuristic. The sequencing methodology proposed locates the bottleneck and then generates the sequence of jobs through it as in a single resource model. This sequence of jobs is then used to dictate the order in which jobs are launched to the facility and the order in which they are to be processed by the other work centres.

The research content of this work investigates five methods for locating the critical resource and employs four dispatching rules. These rules order the work at the critical resource and occasionally assist in locating the critical resource. Having methods to locate the critical resource, as each week's work is introduced, results in the sequencing heuristic being responsive to changes in the product mix. This in turn may influence the location of the critical resource.

Modeling

The goal of generic applicability has influenced the development of the modeling methodology. To be generic the approach should be able to model a variety of job shop scenarios to a degree of complexity sufficient to generate useful results. To perform extensive "what if?" analysis, models should be able to be updated quickly and easily without tedious data refinement and verification. As highlighted in Chapter 2, the complexity of a manufacturing simulator is inversely related to the level of abstraction adopted. The modeling approach in the MSA methodology is at a high degree of abstraction. Work centres are viewed as transformation systems through which jobs pass taking a deterministic period of time.

Simulation

The simulator is at the heart of the MSA methodology. Its objective is to use the critical resource sequencing approaches, the databases and the model,

to simulate the movement of jobs through the facility.

To simplify the simulation methodology, a job "self carries" its own processing data and each defined work centre contains the logic variables that dictate how the job will move through it. No formal network of resources is created, thus freeing the user from the responsibility of creating the links and attempting to verify them. More importantly these links do not have to be rebuilt every time logic changes are introduced.

The remainder of this chapter reports the MSA methodology and how the objectives of generic applicability have been achieved. Chapters 4 and 5, respectively, expand the scheduling and simulation methodology, and Chapter 6 details the software implementation.

3.2 The Conceptual Model

The first stage in the implementation of the MSA philosophy is the development of a conceptual model to assist in making the methodology consistent. The conceptual model is comprised of seven modules (msa1 to msa7) illustrated in Figure 3.1. The remaining sections of this chapter provide an overview of each module and illustrate how the modules integrate to form a consistent approach to the modeling, sequencing, and simulation of manufacturing facilities.

3.2.1 PROJECT PROCESSING (msa1)

A project in the MSA methodology is a set of data files that define the manufacturing facility of interest: four user files and twelve software/project control files. The role of this module is to control the creation, loading, and

editing of individual projects.

One goal behind the software implementation of this module is to limit data input when creating two similar projects. A feature of msa1 and other modules is consequently the ability to create new project files at any point using existing project files, or as will be described later, using simulation progress files.

Facility models should also be easily manipulated, updated, or expanded, to speed up the process of "what if?" simulation. This is an important aspect when simulation is employed to assist in the investigation of the effectiveness of sequencing methods.

From this module there are two routes through the scheduling methodology displayed in Figure 3.1: first is the *actual simulation* route (shown by the solid black line) and second the *Learning System* database update route (the broken line). The actual simulation must be run first before the Learning System knowledge base can be updated. Running the actual simulation creates the necessary files for the periods of interest. These may then be used to run the Learning System. The results from the Learning System may be passed back through the model and the actual simulation run again. The Learning System is discussed in detail later in this chapter.

To launch the actual simulation run, control is passed to module msa2; for a Learning System update, control is passed to module msa7.

3.2.2 SIMULATION PARAMETER CONFIGURATION (msa2)

The goal of module msa2 is to allow the end user to quickly and easily

configure the sequencing and simulation control parameters, as opposed to the basic manufacturing facility files, without having to edit individual data files. The configuration parameters (listed in Table 3.1) are split into three subgroups: Master, Logic, and Output.

The Master Control parameters influence how a job is processed; for example, the size of the transfer batch of a job or the set up time of the work centres visited. The Logic parameters influence how jobs move through each work centre (Universal Transfer) definition. For instance, when a job has finished processing, the logic parameter *unload rule* is accessed to decide whether to move the job to the output queue or to move the job directly to the next work centre. The Output parameters control software operation, examples include decisions on when to save results or what to display on the screen while the simulation is running.

Master

1. batch size method
2. set-up method
3. critical resource identification.
4. simulate x periods
5. due date offset
6. tardiness tolerance

Logic

7. loading rule
8. selection rule
9. unload rule

Output

10. screen display
11. monitoring selection
12. save actions
13. save results after
14. knowledge base update after

Table 3.1: Simulation Control Parameters

3.2.3 MANUFACTURING DATABASE PREPARATION (msa3)

Module msa3 creates the environment for the user to interactively configure the simulation parameters. Msa3 works transparent to the user to configure the databases in view of the parameter values set in module msa2.

The primary functions of msa3 are:

1. Create the "Work" file;
2. Configure the Sequencing Database; and
3. Configure simulation parameters.

A distinction is made between the Jobs Pending file and the Work file. The Jobs Pending file holds a complete list of all jobs that require processing by the facility. The Work file contains a list of jobs required to begin processing in the period about to be simulated. The first function of this module (msa3) is therefore to generate the Work file by selecting and moving jobs from the Job pending file.

The Sequencing Database is used by the simulator when information is required on those jobs being simulated. Two versions of the Sequencing Database are maintained: master and run time. The master, or static, version is created by module msa1 following a BoM input by the user. The second function of this module is to generate the run-time version from the master. Records are created in the run-time Sequencing Database for every item in the Work file and certain blank fields are initialised. These fields include the set-up time, the process and initial transfer batch sizes, and the job number.

In every project and for every period simulated, a control file holds the data to regulate the simulation run. Examples of data include: the number of minutes

to simulate, the number of periods to be simulated, and what results to archive. The last function of this module is the configuration of the Control File.

3.2.4 CRITICAL RESOURCE IDENTIFICATION (msa4)

Five methods for locating the critical resource in the manufacturing sequence are included in the MSA methodology. Differing opinions have been discussed (see Chapter 2) as to what constitutes a critical resource. It can be defined either as the resource most severely overloaded, or as the work centre with the largest queue of jobs waiting. If a resource constantly has a queue of work awaiting production, then barring any unforeseen circumstances, it can be expected to work to capacity. If queues reside in front of several work centres within a facility it may be difficult to ascertain the true location of the critical resource.

When a quick schedule is required and a consistent model is to be employed, then the manual, random, and loading model 1 (LM1) are the preferred options. If sufficient time is available then either one of the manual, random , LM1 methods, or of the two calculation-intensive methods, loading model 2 (LM2) and queuing model 1 (QM1), may be employed.

Manual, random and LM1 methods proactively identify a Critical Resource, that is, identify it before any simulation process is executed. LM2 and QM1 models simulate through the period of interest and, use loading and queuing statistics, respectively, to identify the location of the critical resource.

3.2.5 CRITICALITY LIST GENERATION (msa5)

This module generates the Criticality List, an ordered list of all components

and sub assemblies to be processed by the Critical Resource. The rule applied in ordering the list is the dispatching rule, selected by the user when configuring the parameters in module msa2.

When a work centre becomes idle, a list of jobs available for loading is generated. The generation of the available jobs list is accomplished by the simulator driver (msa7). This generated list is compared against the Criticality List. The job appearing highest on this list is selected. If none of the available jobs appear on the list, then the active dispatching rule of the Universal Transfer (UT) is employed to make a choice. By this approach jobs related to the critical resource are always given preference. A more in-depth description of the use of the Criticality List as part of the Sequencing Methodology is included in the next chapter.

3.2.6 THE MSA SIMULATOR (msa6)

An important contribution of this research is the MSA simulator. The role of the simulator is to move the project forward, period by period. under conditions set up by the previous five modules.

If models LM2 and QM1 are employed to identify the Critical Resource, the simulator may also be used by msa4 to assist decision-making .

Using a form of the discrete-event scheduling approach, the present version of the software simulates in periods of a one-week duration, on a minute-by-minute basis.

The simulator is also responsible for collecting results, both during the simulation of a period and at the end. Two types of results are evaluated:

"specific" and "global". Specific results are concerned with individual jobs and the performance and usage of individual work centres. Global results are macro statistics, such as the number of jobs completed and average utilisation of the whole facility.

3.2.7 LEARNING SYSTEM UPDATE (msa7)

A distinction must be made between an *actual* simulation run and a *Learning System Update* simulation run. The actual simulation run advances a project forward period by period, implementing the user setting of control parameters. At the beginning of a period, the state of the project is held in the project database. When the actual simulation is executed and completed, the project database is appended with the new state of the project.

The Learning System Knowledge Base is a set of five percentile measures of performance, held for every combination of Critical Resource Identification Method and Dispatching Rule. Clearly, when the actual simulation is run, results for only one combination are evaluated. Subsequent execution of this module updates the Knowledge Base for all combinations of the critical resource identification methods and dispatching rules. Since there are four critical resource selection methods (not including manual) and four dispatching rules, each period of interest has to be fully simulated sixteen times. The Knowledge Base may be used to judge the "best" combination of the critical resource identification method and dispatching rule, given the control parameter settings of module msa2. The "best" combination may then be used to rerun the actual simulation from the end of period one to the end of period two. At this time the project state would be overwritten with the improved performance state. The control variable settings, configured by the user in module msa2, are of primary importance. To achieve consistent

results the settings configured by the user are used in the Knowledge Base update. The user is not given the opportunity to change these values.

The following illustrative example will demonstrate the Learning System update facility and the applications of the results. Assume period two is the period of interest. At this point in simulated time (one period completed) the project status is held in the sixteen project data files. The Scheduling Manager wishes to update the Learning System Knowledge Base; consequently, module msa7 is entered. The initial state (period one) of the project is loaded into software arrays in conjunction with the control parameter settings. The period of interest is fully simulated 16 times. At the end of each run the Learning System is the only project file updated. Other project files are left unchanged maintaining the state of the project at the end of period one.

3.3 Summary Of The MSA Approach

Figure 3.1 illustrated the seven modules of the MSA methodology's conceptual model. This chapter has provided a brief overview of the function of each module. The important aspects of the MSA methodology requiring further explanation are:

1. Generation of the jobs file from the current order book;
2. Four Critical Resource selection models;
3. Dispatching rules;
4. Performance criteria;
5. Learning system methodology and database;
6. Simulator and assumptions; and
7. Software implementation.

Chapter 4 details critical elements of the sequencing methodology; Chapter 5 describes the simulator; and Chapter 6 details the software implementation.

CHAPTER FOUR

CRITICAL ELEMENTS OF THE SEQUENCING METHODOLOGY

4.1 Introduction

The philosophy behind the MSA methodology, generic application to model a relatively wide range of manufacturing facilities, was introduced in Chapter 3. Furthermore, a conceptual model of the methodology, consisting of seven modules, was outlined. The objective of this chapter is to provide a detailed explanation of the methodology's critical elements.

4.2 Creation of the Work File

The Jobs Pending file holds a complete order book for the facility, with no specific allocation of jobs to time periods. A Work file, created from the Job Pending list, contains all the jobs to be launched at the beginning of the upcoming period. This section describes one of the functions of the module msa 2 (reference Figure 3.1), that of creating a Work file for a specific time period from the Jobs Pending file.

Each job in the Pending File has an associated due date, composed of the week, and day the job is required to have completed processing by. In an actual manufacturing facility, due dates may not be as specific (i.e., may not use hours and minutes). However, this feature is included here to enable detailed analysis of results, particularly when the Earliest Due Date (EDD) dispatching rule is employed.

Working backwards from the due date the jobs whose latest start date which falls within the period about to be simulated, is moved from the pending file to the Work file. A "lead time offset" may be used to build in some amount of slack time. This offset may be necessary because until the simulation is run, the amount of time a job may spend queuing is probabilistic and unknown.

The following example will clarify the generation of the Work file and the operation of the lead time offset:

Let:

t_k = total minutes available in period k

s_k = start time of period k

The start time of a period is necessarily the end time of the previous time period, hence:

$$t_k = s_{k+1} - s_k \qquad (4.1)$$

Assume each time period to be five working days long, each of 8 hours duration, totalling 40 hours per period. Then:

s_1 = minute 0

s_2 = minute 2400

s_3 = minute 4800 etc.

and

t_k = 2400 for all k.

Assume the current value of the simulation clock is 2400 minutes, implying that period 1 has been simulated and that period 2 is the period of interest, running from the beginning of minute 2400 to the end of minute 4799.

Associated with each job is a theoretical flow time (tf) described as the critical path through the bill of materials. For the purposes of this example, the theoretical flow time is the amount of time required to complete the process batch of the product, given a fully operational facility, empty of other

jobs. A more in depth description is given later in this chapter. The theoretical flow time measure accounts for batch splitting and is calculated as if the SPT rule was used to sequence the jobs through the facility.

Let tf_i be the theoretical flow time for job i and d_i the due date in minutes of job i. Assume for a job i:

$$tf_i = 3000 \text{ minutes, and}$$

$$d_i = \text{due date job i} = 6210$$

Subtraction of the theoretical lead time from the due date for job i determines its latest start time, r_i :

$$r_i = d_i - tf_i \qquad (4.2)$$

$$= 6210 - 3000 = 3210$$

The ready time of job i is minute 3210 which falls within time period 2. Thus job i will be part of the Work file for that period. In more general terms a job will be included in the Work file for period k if:

$$s_k \leq r_i < s_{k+1} \qquad (4.3)$$

$$\text{job i, period k, for i = 1 to n}$$

In a heavily loaded shop the theoretical flow time could be significantly less than the actual flow time experienced due to a greater proportion of throughput time being taken up by queuing. For this reason a user-configured lead time offset has been implemented and is applied to all jobs in the pending list. The following example illustrates how the offset, a user set percentile value, works in practice:

Let "off" be the lead time offset, then the value of the offset for job i is:

$$off_i = \frac{tf_i}{1} \times \frac{off}{100} \qquad (4.4)$$

The offset is added to the value of the theoretical flow time. The calculation for a jobs ready time, equation (4.2) becomes:

$$r_i = d_i - (tf_i + off_i) \qquad (4.5)$$

The effect of adding the offset is to bring the order ready date forward.

A special case to consider is that of a job entered into the order book with a ready time before the start of the current time period:

$$s_k > r_i \qquad (4.6)$$

If the actual flow time is as long or longer than the theoretical flow time, the job will be unavoidably late, hence:

$$s_k + tf_i > d_i \qquad (4.7)$$

Job i might represent a rush order that optimally would have been included in a previous time period, but for some reason was unavailable for sequencing. Job i would be included in the current time period but unless the Earliest Due Date (EDD) dispatching rule was used, no special priority would be given to it during sequencing.

The Work file for the period to be simulated is thus created but in the same order as the Job Pending File, i.e., a first-come order. During the scheduling cycle the Work List is reordered twice. (This point is examined further in section 4.9, as a discussion of how the elements of the sequencing methodology fit together to form a consistent approach).

4.3 Dispatching Rules

Within the MSA model four dispatching rules are used, each one chosen for a specific quality it possesses. The four rules chosen have been selected for their strength with differing problem types. As this thesis is concerned with producing the first working MSA application, investigation of the remaining dispatching rules has been deferred to future work. The four dispatching rules chosen are explained below, along with a discussion of their specific qualities.

4.3.1 RANDOM (RND)

The PowerBASIC *randomize* function is used to randomly order jobs in a list or to choose a job from those available. The resulting sequences will then act as a benchmark against which the performance of other rules will be judged. In the majority of cases the other dispatching rules should out perform random order.

4.3.2 FIRST-COME-FIRST-SERVED (FCFS)

This rule dictates that a queue of jobs be processed in its present order, beginning at the top of the list. Because the queue is not re-ordered whenever a new job arrives, the rule is described as static. Hax (1984, p284) noted that this rule produced good results for heavily loaded single machine situations, a point made in Chapter 2. As critical resources are used to reduce the search area for feasible schedules, a quasi single machine problem is at the heart of MSA. For this reason the FCFS rule was selected for use in this research. Also, because a further objective of this research is the investigation and development of easy to implement but effective

sequencing heuristics, the FCFS dispatching rule was selected for its ease of implementation.

4.3.3 SHORTEST PROCESSING TIME (SPT)

The SPT rule can be described as a dynamic dispatching rule: each time a new job enters it reorders the queue, in increasing processing time order. The Literature survey (Chapter 2) reviewed several studies relating to the SPT rule. Extensions that make the rule more sensitive to jobs with long processing times were also discussed.

A particular quality of the rule is that it reduces the quantity of WIP inventory within the single machine model. This point becomes intuitively obvious when one considers how the rule works: jobs with the shortest processing time are selected, and move quickly through the work centre, thereby reducing the level of WIP inventory. The SPT rule in the general job shop model is believed to exhibit similar qualities.

The SPT rule, however, is not flawless. It is not sensitive to due date tightness or to jobs with long processing times. Adaptations to the rule can be developed to handle these cases, but the qualities mentioned above no longer strictly apply. Because the criticality list is partitioned into the periods in which the jobs are launched, the SPT rule is applied in this research with no adaptations.

4.3.4 EARLIEST DUE DATE (EDD)

The EDD rule dynamically orders a queue of jobs into non-decreasing due date order and is therefore another dynamic rule. A classic result regarding this rule is that for minimising the maximum job tardiness and maximum job

lateness the EDD sequencing is optimal [Baker (1974)]. This attribute of the EDD rule implies that if some jobs will be unavoidably late, the EDD rule guarantees the jobs to be closer to their due date than they would be with the application of a different rule. EDD is included to determine if, in certain circumstances, use of the due date to sequence jobs proves superior to the use of processing times.

4.4 Identifying the Critical Resource

The function of module msa 3 (ref: Figure 3.1) is to specify which one work centre is to be regarded as the Critical Resource for the period to be simulated. Five Critical Resource Identification methods are defined: manual, random, two loading based methods and one queuing based method. The test case analysis of the methodology will give an indication of the effectiveness of each of the methods.

4.4.1 MANUAL METHOD (CR-MAN)

The manual approach allows the user to choose which resource is to be used as the critical resource. This choice of critical resource is maintained until the software user wishes to change it.

The use of this method becomes apparent when, in Chapter 7, investigation is performed into the effect of using the most often chosen resource by one of the other methods as the critical resource for all periods simulated.

4.4.2 RANDOM METHOD (CR-RND)

For each simulated period the critical resource is selected randomly from those defined to the facility model. This method is used as a benchmark,

against which the effectiveness of the other, more structured approaches to identification of the critical resource can be tested.

4.4.3 LOADING METHOD 1 (CR-LM1)

This loading method uses the potential loading of jobs in the Work file to identify the critical resource. Jobs already in processing on the shop floor are not accounted for, nor whether the jobs in the Work file will complete processing within the period to be simulated. Accounting for the processing that would occur in the period of interest as well as the jobs currently being processed by the facility would have required the simulation model to be run. Further to determine the order in which the jobs are processed and time spent queuing can only be determined by running the simulation.

This method is a quick-to-use, structured method that will identify a critical resource before any simulations are run. If the product mix does not change significantly from period to period, the loading effect of the Work file may be representative of the loading effect created during the jobs simulation through the facility.

The potential loading is calculated as follows: let n be the number of jobs in the generated Work file for the period about to be simulated. Let the loading or required time of job i on resource j be:

$$= l_{ij}. \qquad (4.8)$$

Total loading (tl_j) of resource j is expressed as:

$$tl_j = \sum_{i=1}^{n} l_{ij} \qquad (4.9)$$

for jobs i = 1 to n
for resources j = 1 to m.

Not all jobs necessarily visit each resource. Thus l_{ij} may equal 0 for some values of i. Loading is usually expressed as a percentage of machine time available (mt):

$$tl_{jk} = \frac{\sum_{i=1}^{n} l_{ij}}{mt_{jk}} \times \frac{100}{1.} \qquad (4.10)$$

There is no theoretical upper limit to the potential loading figure and the resource with the highest values for the tl_{jk} variable is selected as the critical resource.

Suggested approaches to dealing with the overload situation are included in the future work chapter but are beyond the scope of this research project.

Because a job's total processing time is included in the loading calculation even if it cannot complete processing, the loading approach is not particularly accurate. When a static approach is used (as in this model), the indeterminate queuing times make this inaccuracy difficult to resolve. For this reason, model CR-LM2 was developed

4.4.4 LOADING METHOD 2 (CR-LM2)

CR-LM2 is a dynamic Critical Resource Identification method that employs utilisation (actual loading) statistics to locate the Critical Resource. This method operates by taking the user configuration (msa 2) and the generated databases (msa 3) and simulating the period of interest using the selected dispatching rule. This simulation is performed without a defined critical resource hence no criticality list. At this point in the scheduling cycle, the

Work file has been ordered according to the active dispatching rule. Launching of jobs from the Work file strictly follows the order of the Work file; subsequent movement of jobs between queues and resources uses the active dispatching rule. Thus, in effect, the jobs are pushed through the facility. Note that when the SPT rule orders the Work file, total processing time to complete a job is used.

After simulation of the period is completed, the resource with the highest utilisation is labelled as the critical resource. The formula for utilisation expresses the processing time (pt_{jk}) and set up time (su_{jk}) as a percentage of the total machine time available in period k.

$$ul_{jk} = \frac{pt_{jk} + su_{jk}}{mt_{jk}} \times \frac{100}{1} \qquad (4.11)$$

where su_{jk} is the total set up time of resource j in time period k.

It is possible for two resources to have the same utilisation figures, both highest for the period simulated. To break the tie the queuing statistics of method CR-QM1, (discussed below), are used.

4.4.5 QUEUING METHOD 1 (CR-QM1)

CR-QM1 is a dynamic queuing model that employs queuing statistics to determine the identity of the Critical Resource. As in CR-LM2 the simulation parameter configuration and currently active dispatching rule are used to sequence the jobs through the facility (with no reference to a Critical Resource or Criticality List).

After simulation selection of the critical resource is based on "highest average

queue length with respect to time" over the period in question. This indicates a resource that restricts the flow of parts to the rest of the facility, a Critical Resource.

The following is a discussion of the calculation of the performance measure. The MSA simulation modeler uses a next event schema to update the clock. The average queue length statistic is evaluated using the formula:

$$jqjt_n^{ave} = \frac{\sum_{i=1}^{n} jqjt_i (t_i - t_{(i-1)})}{t_n} \qquad (4.12)$$

where $jqjt$ is the number of jobs in the queues facing resource j at time t.

The following example is presented to clarify the above equation. The block graph (Figure 4.1) demonstrates the queue lengths at different discrete event times.

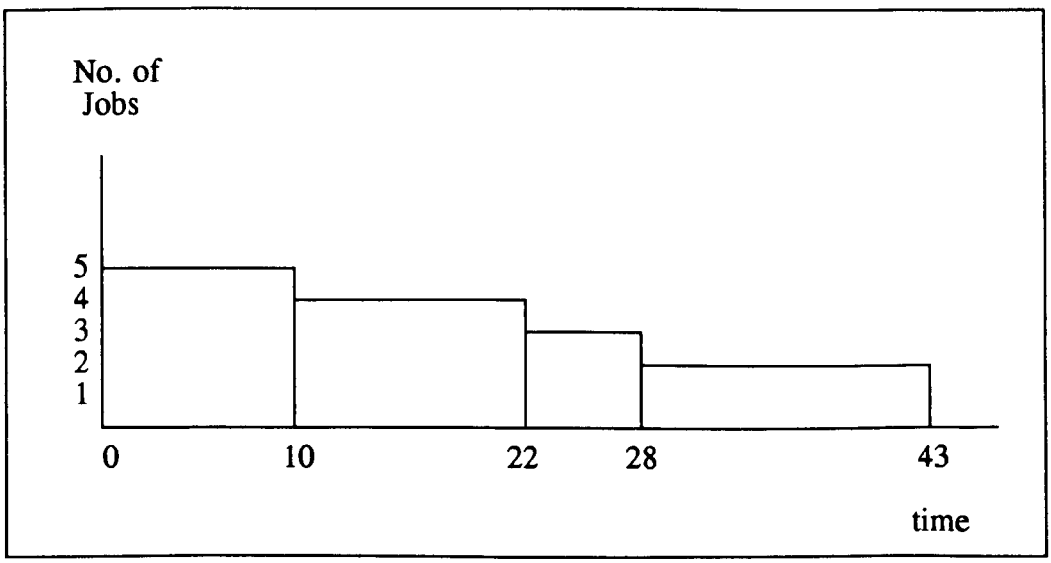


Figure 4.1: Queue Length Example

Times 10, 22, 28 and 43 are event times. Between these times the state of the project does not change; in particular, no jobs move from one queue to another.

After ten minutes the average queue length is:

$$= \frac{(10 - 0)5}{10} = 5 \text{ jobs.}$$

Therefore after ten minutes, one job has completed processing and one more is loaded from the queue onto the resource. The processing time of the job is 12 minutes; thus for the next event time of 22 the average queue length is then:

$$= \frac{50 + (22 - 10)4}{22} = 4.45 \text{ jobs.}$$

The resource with the highest average queue length over the time period will be selected by this model (CR-QM1) as the Critical Resource. In the unlikely event of two resources having the same value for jq_{jt}^{ave} the "average processing time of the queues" will determine the critical resource.

4.5 Batch Sizes

A distinction can be made between a process and a transfer batch. A process batch is the quantity of a product required which was input by the user when the Job Pending file was created. A transfer batch is the quantity processed at a work centre before transferral to the downstream work centre. A process batch does not have to be divisible exactly by the size of the transfer batch:

$$pb = (x * tb) + y \qquad (4.13)$$

where *pb* and *tb* are the process and transfer batch sizes and x and y are integer values such that

$$0 \leq y < tb \qquad (4.14)$$

In module msa 2 the user has a choice of transfer batch size methodologies. There are four available methodologies:

none:

The transfer batch size is set equal to the size of the process batch, that is:

$$pb_i = tb_i \qquad (4.15)$$

for all jobs i = 1 to n.

product:

Each individual product has its own transfer batch size. This size is applied for all components, sub assemblies and the end item. (The value used having been previously input into the model when entries were made into the Jobs Pending file.)

half:

In this option the transfer batch size for each product is half of the process batch size.

$$pb_i = \frac{tb_i}{2.} \qquad (4.16)$$

If the value of the process batch size is odd then the equation (4.16) becomes:

$$pb_i = \frac{tb_i + 1}{2} \qquad (4.17)$$

global:

One transfer batch size is used for all the products being simulated in the particular period of interest.

$$tb_i = tb_g \qquad (4.18)$$

where tb_g is the global transfer batch size.

If the global value of the transfer batch is larger than any process batch, the transfer batch size is set to the value of the process batch.

Thus, if

$$tb_g > pb_i \qquad (4.19)$$

then

$$tb_i = pb_i. \qquad (4.20)$$

The following example illustrates the operation of batch sizes. Assume a product PR1 consists of two components C1 and C2. 2 x C1 and 1 x C2 are required to produce one PR1. Batch sizes are always expressed in relation to the end item.

For PR1

```
tb   =   10
pb   =   20
```

then,

tb of C1 is 20 units, and
tb of C2 is 10.

Calculation of the size of the transfer batch depends on the number of components required to produce one transfer batch quantity of the final product.

If the global batch size methodology is selected and the value of the *tb* was set to 30, such that *pb* < *tb*, then:

```
tb of PR1 is 20
tb of C1 is 40
tb of C2 is 20.
```

This produces the same effect as setting the transfer batch equal to the process batch.

The scheduling Manager is responsible for the choice of *tb* size. The transfer batch size is therefore not directly linked to shop loading, length of the processing time etc.

The batch sizing methods were implemented but not used in the test case analysis. It was developed in parallel to the main code to ease the application and investigation of batch sizing methods at a later date. It is detailed here simply for the purpose of completeness and is not used in the test analysis of

the methodology reported in Chapter 7. The use of batch sizes in sequencing research is discussed in the future work chapter, Chapter 8.

4.6 Set Up Times

Five available options are available to define work centre set up times, from no set up time through to an individual part/centre dependent set up times. The available settings are:

No Set Up:

No set up times are used:

$$su_{ij} = 0 \tag{4.21}$$

for all jobs i on resource j.

Global:

One set up time for all parts on all work centres:

$$su_{ij} = su_g \tag{4.22}$$

where su_g is the global set up time.

Resource:

Each work centre has an individual set up time which is used regardless of what part is to be processed. Thus,

$$su_{ij} = su_j \tag{4.23}$$

for all jobs i = 1 to n on resource j.

Product:

Each sub assembly and component of a product has the same set up time on all work centres. Different products may have different set up times:

$$su_{ij} = su_i \qquad (4.24).$$

where su_{ij} is the set up time of job i on resource j.

Part:

Each component and sub-assembly will have a specific set up time for each work centre visited, a part/work centre dependent set up time.

With the desire to produce a generic modeler, all five options have been included. In practice one method would dominate each test case application.

4.7 Queue Lengths

Once loaded into the model, a job will reside within a defined Universal Transfers in one of the queues (input, component, or output) or at the process. The number of jobs residing in a queue is controlled by the queue length variables quantified when a UT is defined.

This, however, only weakly controls the quantity of inventory residing in a queue at any one time. Assume, for example, the input queue length is two jobs and sub-assemblies SA1 and SA4 are already in the queue. No sub-assemblies or components may then join this queue. Only a transfer batch of SA1 or SA4 would be allowed to join the queue and the totals available would simply be increased.

If the queues lengths are set at too small a value, problems may arise. For instance, assume a resource is required to produce product PR1 from components SA1 and SA2 and the same resource is needed to produce product PR2 using components SA3 and SA4. If, as in the previous example, the queue length is 2 and already contains SA1 and SA4, then neither SA2 or SA3 will be allowed to join the queue. Thus the resource is in effect blocked.

At this point the software, detecting the full queue and incomplete sets of parts, would requests the queue length be increased and the period simulated again.

The purpose of queue length control is to restrict the number of jobs that may reside at any one UT. This restriction makes it possible, for example, to use the software to model and investigate JIT or Load Level approaches to production scheduling.

4.8 Critical Path Through The BoM

In the majority of sequencing research (see Fry et al (1988)), the critical path through the BoM refers either to the time to produce one of an end item, or to the time required to complete the batch on a lot-for-lot basis. In such research the SPT rule is used to sequence the parts, with no split batches through an empty and idle facility which is fully operational.

This research employs a slightly different approach. To obtain a more accurate indication of the facility's performance and effectiveness, the Critical Path should reflect the batch-sizing methodology being used. For example, if the process batch size is 19 and the transfer batch size is 10, then given an empty facility, the Theoretical Flow Time should reflect the time necessary to

produce two transfer batches of size 10 and 9, manufactured in sequence.

The figure is calculated just prior to the start of a simulation run, because only at that time is it known which the set-up and batch size methods are to be used.

4.9 Sequencing Heuristic

The preceding sections of this Chapter have detailed critical elements of the complete sequencing heuristic. This section takes these elements to illustrate their interaction in forming a consistent approach to the sequencing of a general job shop.

Generation of the Work File

By applying the due date offset to the theoretical flow time (the critical path through the BoM), the jobs that are to begin processing in the next period are identified and moved to the Work File.

First Ordering of the Work File

The Work file is ordered according to the selected dispatching rule. This is particularly important if CR-LM2 or CR-QM1 methods are used to identify the critical resource.

The Work file is partitioned into sections, each containing jobs launched in particular periods. In the first section the jobs at the top of the list are those launched in the period further most from the current period. New jobs entering the list are not placed ahead of these. Each section is ordered individually with respect to the dispatching rule being used.

Critical Resource Identification

The selected dispatching rule must be used to simulate the period of interest if models CR-QM1 or CR-LM2 are used to identify a Critical Resource. As stated previously, no Criticality List is referenced when these simulations are run, and jobs are simply pushed through the facility using the active dispatching rule.

Criticality List

The Criticality List is an ordered list of jobs to be processed by the critical resource. It is generated after identification of the critical resource for the current period.

The list is also partitioned as it is in the Work file and each section ordered individually with respect to the dispatching rule being used.

Second Ordering Work File

Once the Critical Resource has been identified and the generated Criticality List has been ordered, the Work file is again ordered. At this time, however it is ordered with respect to the Criticality List. Those jobs that are contained in the Work file but do not pass through the Critical Resource are ordered second, using the dispatching rule.

Launching of Jobs

At the start of a period, all jobs in the Work file are launched. The first components at the bottom of the BoM tree are loaded into the queues of their respective gateway work centres. The order in which the jobs are launched follows the strict order of the Work file. If a queue is full and some component of a product cannot be launched then no component of that product is launched.

Loading Jobs

When a resource becomes idle and all components are in the input queues, the jobs eligible for processing are checked against the Criticality List to determine which to launch. The job highest on the Criticality List is selected for loading. If none of the jobs in the input queues are on the Criticality List then the dispatching rule is employed to select one.

Removing Jobs From the Criticality List

Once a job's component passes through the critical resource it is then removed from the Criticality List. This is done to eliminate competition for processing at an upstream work centre between this processed job and a job which has not yet passed through the critical resource. If competition for production time were allowed, the processed job may be given a higher priority, to the exclusion of one which is critical resource bound.

4.10 Evaluation of Performance

The performance criteria are split into three categories:

1. specific data
2. aggregate data
3. learning system knowledge base

Within each category, four sets of data are collected to measure customer satisfaction, effectiveness, efficiency, and work in progress inventory, respectively.

Effectiveness refers to the number of jobs completing processing in the time period in question. Efficiency describes resource performance and usage. Customer satisfaction reflects due date performance, and work-in-progress (WIP) inventory data describes the number of products within the facility during the period simulated.

The aggregate data is calculated from the support data, and from the aggregate data the learning system database is updated.

4.10.1 SPECIFIC DATA

Fourteen performance criteria are outlined and divided among the four performance evaluation categories in Table 4.2.

Customer Satisfaction

1. job tardiness
2. job delivery error

Effectiveness

3. theoretical flow time
4. actual flow time
5. manufacturing error

Efficiency

6. machine time available
7. set up time
8. processing time
9. idle time
10. no work time
11. utilisation 1
12. utilisation 2

WIP Inventory

13. average processing time of queue
14. average number of jobs in queue

Table 4.2: Performance Evaluation Categories

1. job tardiness

A job that is either late or early is described as being tardy. Tardiness is measured by the difference between the completion time of a job and the due date; it can be either positive or negative.

A first simple equation to calculate tardiness is to take the difference between the completion time and due date of job i:

$$t_i = c_i - d_i \qquad (4.25)$$

It is important to note that the tardiness value is negative when a job is early.

The MSA methodology has the ability to include an estimated amount of tardiness when scheduling the launch of work, known as the tardiness tolerance (*tt*). The tardiness tolerance for job i is a percentage of the theoretical flow time:

$$tt_i = \frac{tf_i}{1} \times \frac{tt}{100} \qquad (4.26)$$

Job i is described as being on time if the absolute value of the tardiness value, from equation 4.26, is less than the tardiness tolerance of job i, that is:

$$|c_i - d_i| \leq tt_i. \qquad (4.27)$$

Job i is early if:

$$c_i < d_i \text{ and } d_i - c_i > tt_i \qquad (4.28)$$

and late if:

$$d_i < c_i \text{ and } c_i - d_i > tt_i. \qquad (4.29)$$

2. job delivery error

For completeness this measure has been included in this discussion. The delivery error of job i is simply the absolute value of the tardiness value of job i:

$$de_i = |c_i - d_i| \qquad (4.30)$$

$$\text{for } i = 1 \text{ to } p$$

This value will be used later in calculating the values for the Learning System.

3. theoretical flow time

The theoretical flow time may be described as the critical path through the Bill of Materials. The process of deriving this figure was described and illustrated earlier in this chapter.

$$\text{theoretical flow time of job } i = tf_i$$

4. actual flow time

The value of the actual flow time represents the amount of time job i spent in the system either being processed or queuing. It is equal to the difference between the completion time (c_i) and the job launch time (l_i):

$$af_i = c_i - l_i \qquad (4.31)$$

$$\text{for } i = 1 \text{ to } p$$

l_i is the time job i was launched to the facility and is not to be confused with the ready time derived in equation 4.2.

The theoretical flow time does not account for batch splitting or sub contracting work hence the situation may exist where:

$$tf_i > af_i \qquad (4.32)$$

that is, theoretical flow time is greater than actual flow time. The theoretical flow time also does not take into consideration queuing time hence a situation may exist where the actual is greater than the theoretical flow time:

$$af_i > tf_i. \qquad (4.33)$$

5. manufacturing error

This is an absolute figure to give an indication of the accuracy of the theoretical flow time prediction of the actual flow time realised. The Manufacturing error for job i is defined as the absolute difference between the actual and theoretical flow times:

$$me_i = |tf_i - af_i| \qquad (4.34)$$

$$\text{for } i = 1 \text{ to } p$$

Note that job i completed processing in the period of interest but could have begun processing in a previous period.

6. machine time available

The time that a machine was available for the processing of jobs in the time period of interest. It is not necessarily equal to the amount of time available in the time period. This difference may be due to planned preventive maintenance or operator absenteeism:

$$\text{Machine time available in period k} = mt_{jk} \quad \text{for } j = 1 \text{ to } m$$

7. set up time

Performance measures 7, 8, 9, 10 and 11 are calculated dynamically as the simulation is running.

st_j measures the amount of time machine j was in set up mode preparing for processing:

$$\text{Total set up time of resource j in k} = st_{jk} \quad \text{for } j = 1 \text{ to } m$$

8. processing time

pt_j records the time spent by a resource j processing jobs. This is the productive time of the resource:

$$\text{Processing time of resource j in k} = pt_{jk} \quad \text{for } j = 1 \text{ to } m$$

9. idle time

The amount of time the resource spent idle waiting for work:

$$\text{Total idle time of resource j in k} = it_{jk} \quad \text{for } j = 1 \text{ to } m$$

10. no work time

This measure is different from idle time. It is when a transfer batch has been completed but a full set of components where not available to start another batch:

$$\text{Total no work time of resource j in k} = nt_{jk} \text{ for } j = 1 \text{ to } m$$

If a resource is not being set up or processing a part then it is necessarily idle, waiting for parts or non functional. Hence:

$$mt_{jk} = st_{jk} + pt_{jk} + it_{jk} + nt_{jk} \quad \text{for } i = 1 \text{ to } m \qquad (4.35)$$

11. utilisation 1

Utilisation 1 treats set-up time as idle time. Utilisation of resource j in period k is then expressed as:

$$u1_{jk} = \frac{pt_{jk}}{mt_{jk}} \times \frac{100}{1} \qquad (4.36)$$

Processing time is expressed as a percentage of the total machine time available. Clearly $0 \leq u1 \leq 100$.

12. utilisation 2

Utilisation 2 regards set up time (since it unavoidable) as part of the processing time. Utilisation of resource j in period k is then expressed as:

$$u2_{jk} = \frac{pt_{jk} + st_{jk}}{mt_{jk}} \times \frac{100}{1} \qquad (4.37)$$

13. average processing time of a queue

At event time t_i the formula for evaluating the average processing time of a queue is:

$$pq_{jt_n}{}^{ave} = \frac{\sum_{i=1}^{n} pq_{jt_i}(t_i - t_{(i-1)})}{t_n} \qquad (4.38)$$

where pq_{jt} is the processing time of queue j at event time t.

When *fathoming* a particular event time at t_i jobs may move between queues. The average processing time of a queue with respect to time is calculated at the beginning of the *fathoming* process.

Blank Page

Customer Satisfaction

1. number of jobs on time
2. number of jobs early
3. number of jobs late
4. total early time
5. total late time
6. average early
7. average late
8. frequency of tardy jobs
9. total delivery error

Effectiveness

10. total theoretical flow time
11. total actual flow time
12. mean flow time
13. maximum flow time
14. minimum flow time

Table 4.3 : Aggregate Performance Evaluation Categories

Efficiency

15. total manufacturing error
16. total machine time available
17. total set up time
18. total processing time
19. total idle time
20. total no work time
21. utilisation 1
22. utilisation 2

WIP Inventory

23. average processing time of all queues
24. average number of jobs in all queues

Table 4.3: (continued)

1. number of jobs on time

The total number of jobs on time is given by:

$$NT_k = \sum_{i=1}^{p} (x_i) \quad \text{where} \quad x_i = \begin{cases} 1 \text{ if } |c_i - d_i| \leq tt_i \\ 0 \text{ if } |c_i - d_i| > tt_i \end{cases} \tag{4.40}$$

for an explanation of tt, c, and d refer to equations E.0 and E.0 and the related discussion.

2. number of jobs early

The total number of jobs early in period k

$$NE_k = \sum_{i=1}^{p} (x_i) \quad \text{where} \quad |c_i - d_i| > tt_i,$$

$$\text{then } x_i = \begin{cases} 1 \text{ if } c_i < d_i \\ 0 \text{ if } c_i > d_i \end{cases} \tag{4.41}$$

3. number of jobs late

The total number of late jobs is given by:

$$NL_k = \sum_{i=1}^{p} (x_i) \quad \text{where} \quad |c_i - d_i| > tt_i,$$

$$\text{then } x_i = \begin{cases} 1 \text{ if } c_i > d_i \\ 0 \text{ if } c_i < d_i \end{cases} \tag{4.42}$$

4. total early time

Total early time for all early jobs in i = 1 to p which graduate from the facility within k is defined as:

$$TE_k = \sum_{i=1}^{p} (x_i) \quad \text{where} \quad |c_i - d_i| > tt_i,$$

$$\text{then } x_i = \begin{cases} d_i - c_i \text{ if } c_i < d_i \\ 0 \text{ if } c_i > d_i \end{cases} \qquad (4.43)$$

5. total late time

Total late time for the period k:

$$TL_k = \sum_{i=1}^{p} (x_i) \quad \text{where} \quad |c_i - d_i| > tt_i,$$

$$\text{then } x_i = \begin{cases} c_i - d_i \text{ if } c_i > d_i \\ 0 \text{ if } c_i < d_i \end{cases} \qquad (4.44)$$

6. average early

The average early calculation is a simple average taking the total early time and dividing by the number of jobs which were early, hence:

$$AE_k = (1/NE_k) \times TE_k \qquad (4.45)$$

7. average late

Again a simple average, taking the total late time (TL) and dividing by the number of jobs graduating late in period k, hence:

$$AL_k = (1/NL_k) \times TL_k \qquad (4.46)$$

8. frequency of tardy jobs

A tardy job is one that is late or early after taking account of the tardiness tolerance. This frequency expresses the number of tardy jobs as a percentage of the total number of jobs.

$$FT_k = \frac{NE_k + NL_k}{NE_k+NL_k+NT_k} \times \frac{100}{1} \qquad (4.47)$$

$$= \frac{NE_k + NL_k}{p} \times \frac{100}{1} \qquad (4.48)$$

9. total delivery error

The total delivery error is simply the summation of delivery error for all the products graduating in k.

$$DE_{ik} = \sum_{i=1}^{p} |c_i - d_i| \qquad (4.49)$$

10. total theoretical flow time

The total theoretical flow time is the summation of all the individual theoretical flow times for the jobs in p, hence:

$$TF_k = \sum_{i=1}^{p} tf_i \qquad (4.50)$$

11. total actual flow time

Actual flow time is described as:

$$af_i = c_i - l_i \qquad (4.51)$$

Total flow time is for all jobs completed in period k and is given by the formula:

$$AF_k = \sum_{i=1}^{p} af_i \qquad (4.52)$$

12. mean flow time

The above two effective measures do not account for significant processing time differences between jobs. To give a feel for this, mean, maximum and minimum flow time criteria are maintained. Mean flow time is defined as:

$$\overline{F}_k = (1/p) \times AF_k \qquad (4.53)$$

13. maximum flow time

Maximum flow time of jobs i = 1 to p in k:

$$F_k^{max} = \max \{AF_i\} \qquad (4.54)$$
$$\text{for } i = 1 \text{ to } p$$

14. minimum flow time

$$F_k^{min} = \min \{AF_i\}$$
$$\text{for } i = 1 \text{ to } p$$

15. total manufacturing error

This value is used in the Learning System when determining effectiveness of a simulated schedule:

$$ME_k = |TF_k - AF_k| \qquad (4.55)$$

16. total machine time available

The time a machine is available for processing in a time period does not necessarily equal the amount of time in a period. The total machine time available is a sum over all m resources:

$$MT_k = \sum_{j=1}^{m} mt_{jk} \qquad (4.56)$$

17. total set up time

Measures 17, 18, 19, 20 and 21 are summations of the Specific measures 7, 8, 9, 10 and 11 and are calculated at the end of a simulation run.

Summing over all resources the total time spent in the facility setting up resources to accept work is expressed as:

$$ST_k = \sum_{j=1}^{m} st_{jk} \qquad (4.57)$$

18. total processing time

Total time in period k spent processing jobs:

$$PT_k = \sum_{j=1}^{m} pt_{jk} \qquad (4.58)$$

19. total idle time

The total time that resources where idle waiting for work in period k:

$$IT_k = \sum_{j=1}^{m} it_{jk} \qquad (4.59)$$

20. total no work time

This measure is different form idle time, it is when a transfer batch was completed but a full set of parts where not available to start another transfer batch of the same product:

$$NT_k = \sum_{j=1}^{m} nt_{jk} \qquad (4.60)$$

If a resource is not being set up or processing a part then it is necessarily idle, waiting for parts or non functional. Clearly,

$$MT_k = ST_k + PT_k + IT_k + NT_k \qquad (4.61)$$

for all resources 1 to m in period k.

21. utilisation 1

Utilisation 1 treats set up time as idle time. This aggregate figure gives a utilisation measure for the whole facility.

$$U1_k = \frac{PT_k}{MT_k} \times \frac{100}{1} \qquad (4.62)$$

22. utilisation 2

Utilisation 2 regards set up time as part of the processing time since it is unavoidable.

$$U2_k = \frac{PT_k + ST_k}{MT_k} \times \frac{100}{1} \qquad (4.63)$$

23. average number of jobs in all queues

The derivation of this formula was discussed in Chapter 4, Section 4.4.5.

$$JQ_k = \sum_{j=1}^{m} jq_{jk} \qquad (4.64)$$

24. average processing time of all queues

This equation is similar to the time average number of jobs in all queues but uses the processing time in place of the queue length.

$$PQ_k = \sum_{j=1}^{m} pq_{jk} \qquad (4.65)$$

4.11 The Learning System

The learning system is a summary of the aggregate performance figures.

Measuring performance as effectiveness, customer performance and efficiency, three linear measures are defined to judge performance of the sequencing heuristic.

Customer Performance

$$CP_k = \left[1 - \left[\frac{\text{total delivery error}}{\text{total theoretical flow time}} \right] \right] \times \frac{100}{1}$$

$$= \left[1 - \left[\frac{DE_k}{TF_k} \right] \right] \times \frac{100}{1} \qquad (4.66)$$

Note that the delivery error is an absolute figure and tends towards zero as customer performance improves. The Customer performance percentile has an upper bound of 100% and no theoretical lower limit.

Effectiveness

$$ES_k = \left[1 - \left[\frac{\text{total manufacturing error}}{\text{total theoretical flow time}} \right] \right] \times \frac{100}{1}$$

$$= \left[1 - \left[\frac{ME_k}{TF_k} \right] \right] \times \frac{100}{1} \qquad (4.67)$$

Note that as the absolute value of the manufacturing error tends towards zero, the facility effectiveness tends to 100%. Because the manufacturing error is an absolute value, the effectiveness measure has an upper bound of

100%. There is no theoretical lower bound.

Efficiency

The aggregate measure Utilisation 1 ($U1_k$) is used as the measure of efficiency because it is already a percentile value directly reflecting the efficiency level of the facility.

Because knowledge of actual output production time is so important utilisation measure U1 is chosen over U2. By using U1 the efficiency figure is directly accessible. If it is unacceptably low other aggregate figures may be accessed to determine why. Selecting U2 may mask problems by producing a high percentage value that reflects set ups instead of production.

Overall Performance

For each combination of the Critical Resource selection model (CR-R, CR-M, CR-LM1, CR-LM2, CR-QM1) and dispatching rule (RND, FCFS, SPT and EDD), the three learning system measures are evaluated. The overall performance figure produced is a weighted average of the four measures. By changing the weights it is possible to favour a particular measure and then focus the scheduling and simulation on minimising this measure. The test cases (Chapter 7) will illustrate the effects of weighting one approach over another. The overall performance figure (OEM_k) for period k is evaluated as:

$$OEM_k = \alpha_1 CP_k + \alpha_2 ES_k + \alpha_3 U1_k \qquad (4.68)$$

Clearly α_i (i = 1 to 3) equals 0.333 if equal weight is given to each goal.

CHAPTER FIVE

DEVELOPMENT OF A GENERIC MANUFACTURING SIMULATOR

5.1 Introduction

Chapter three outlined the seven modules, msa 1 through 7, of the MSA sequencing and simulation methodology (see Figure 3.1). One of the most extensive parts of applying the methodology, is the development of the complex simulator capable of evaluating bottleneck machine scheduling. This chapter explains and examines this simulator which has been built into msa6.

The simulator comprises a modeler and a simulation driver. The modeler consists of two databases, the UT definition, and the Sequencing Database. The simulation driver implements the portions of the sequencing methodology (ref: Section 4.8) concerned with moving parts through the defined model. The aspects of the sequencing heuristic implemented by the simulation driver are: job launching, job loading, and removing jobs from the Criticality list. The simulation driver also calculates the results and archives the project status at the end of a simulated period.

The innovative feature of the MSA approach to simulating a manufacturing facility is the use of only one modeling entity to represent all resources. The simulation driver is based on the Event Scheduling Framework. Events are used to drive the simulation with each job "self carrying" their routes and processing data into each node or entity.

This chapter discusses the two primary areas of the MSA simulator: the modeling methodology, and the methodology of the simulation driver, i.e., how the information databases are used to simulate a manufacturing facility.

5.2 Generic Modeling Methodology

A generic manufacturing modeler is able to represent a variety of manufacturing scenarios with relatively little tailoring to the specific configuration of the facility of interest. This requires a common denominator, an element which is present in all the manufacturing facilities to be modeled. The common denominator serves as the basis for the model, examples include jobs, work centres and cells. The MSA methodology uses individual processes in the manufacturing cycle. (a "process" is defined as any activity in the product manufacturing cycle that takes a known period of time). Examples of such processes might include part machining, or quality inspection. Each process is represented by a "Universal Transfer" (UT) which is the one modeling entity defined in MSA. A UT consists of two input queues, a process that carries out a function on the product, and one output queue. How a job moves through the UT is dictated by the specific configuration of each UT and the attributes of individual jobs. The UT is illustrated in Figure 5.1 and explained in greater detail later in this chapter.

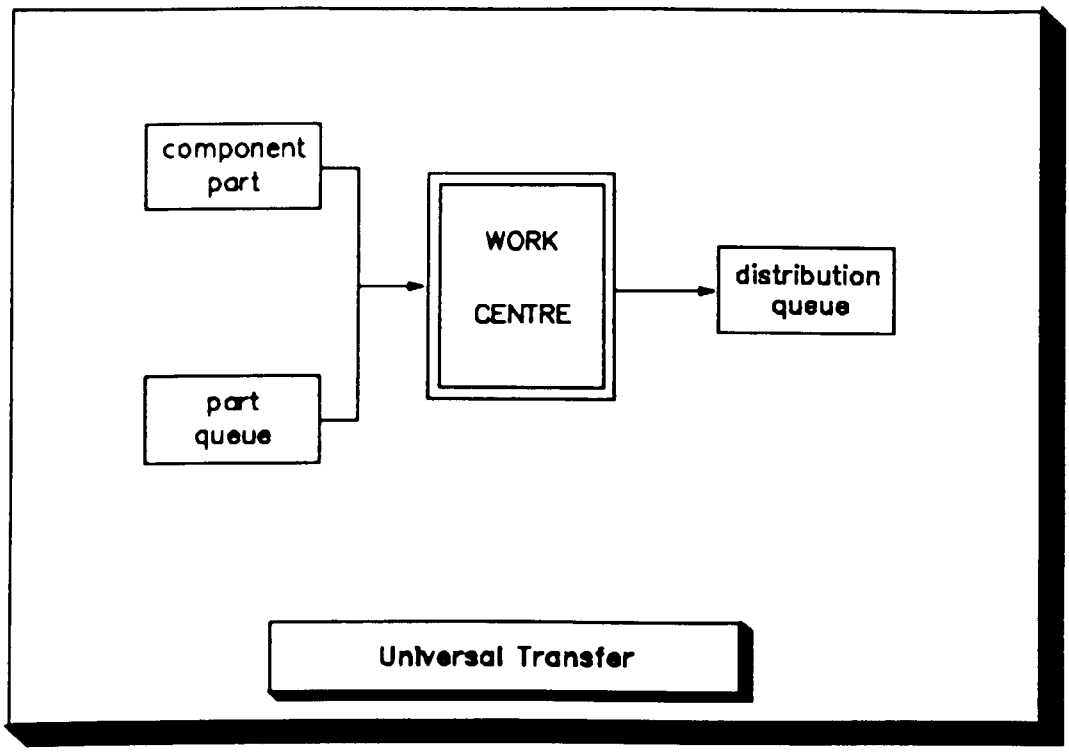


Figure 5.1: A Universal Transfer

5.2.1 FACILITY NETWORK MODEL

Unlike some other manufacturing simulation packages, the MSA modeler does not maintain an explicit representation of a facility network. Each UT is defined in isolation and each job, via the Sequencing Database, "self-carries" its route through the facility. Therefore, to ensure the presence of all required database and resource definitions it is important that the Logic Check routine is executed before launching a simulation run.

There are several advantages to having no defined network. If a new resource is to be added or one deleted a network model does not have to be accessed and modified. Only the Sequencing Database needs to be referenced to determine if any deleted resource is required by any job about

to be launched or currently being processed. Modifying routes taken by jobs through a facility requires only the BoM file to be edited.

When a job is launched the gateway components are identified and loaded onto the relevant resources. Because each UT is treated as a separate entity, routes are 100% flexible. Jobs can be launched at any resource, can visit any resource as often as is required, and can exit the system from any desired resource.

The two databases are called the *Universal Transfer Definition* and the *Sequencing Database*.

5.2.2 UNIVERSAL TRANSFER DEFINITION

A *Universal Transfer*, is used to model all resource types. Figure 5.1 illustrates the four elements of a UT: part, component and output queues, and the work centre or process:

part queue: normally used to hold sub assemblies or components that do not require any other parts to be available before processing begins at the work centre.

component queue: the second of the input queues and is normally used to hold parts to be processed by a work centre requiring other parts to be present and available. For instance, if the work centre of interest is to take three parts and assemble them into an end item, a batch of each part clearly must reside in this queue before assembly begins.

In normal operation both queues are used as described above. It is possible to operate with all parts passing through just one queue with the other queue

could be used as warehouse storage and given virtually unlimited capacity.

process: this element of a UT definition is normally used to represent the actual work centre. It may, however, also be used in a more general sense to represent any manufacturing stage for which processing times are known and linear. Linear processing times imply that if 5 minutes are necessary to produce one unit of a component C1, then when using the same resource, production of two units would require 10 minutes.

In addition to work centres, other entities that may be modeled are quality inspection stations (testing every unit) and some material handling equipment. The only prerequisite is linearity in processing time.

output queue: batches may be moved to the output queue once they have completed the UT process section.

UT definitions are stored in the *UT Definition File*, a series of records containing thirteen fields each. The fields are listed in Table 5.1 divided into three groups: information, logic, and operational. The information parameters contains two user information fields which have no direct effect on the operation of the software. The third information field is generated and used by the software to identify UT's. Logic parameters define which queue jobs move to within the UT and which rule to use when selecting a job. Operational parameters apply specifically to the UT, for example, length of queues or whether the process is operational or not.

Information Parameters

Resource Identifier: a unique number, allocated on a sequential basis by the software, to each UT defined. It is used by the software to identify a UT.

Resource Name: a user input name for the UT. For user information only; has no direct effect upon the operation of the software.

Resource Code/Group: a second alphanumeric field for user information only.

Logic Parameters

The logic parameters dictate the movement of jobs through the Universal Transfer.

Launch Job Logic: controls the order in which the queues are accessed. Each work centre is serviced by two input queues, component and part; when a work centre becomes idle these are accessed to determine the next job to be processed. A set of available jobs is generated to determine which job to process next.

The input queue may be accessed first and compared against the Criticality List, or the component queue may be accessed first. If the first queue accessed does not contain a job eligible for loading then the other queue is accessed.

Information

1. resource identifier
2. resource name
3. group/cell number

Logic

4. launch job logic
5. dispatching rule
6. unload job logic

Operational

7. operational
8. resource efficiency
9. simultaneous working
10. set up time
11. part queue length
12. component queue length
13. output queue length

Table 5.1: Universal Transfer Parameters

Dispatching Rule: dispatching rule to use or if a comparison to the Criticality List is necessary. Once the set of available jobs has been generated, a job must be selected and loaded.

Unload Job Logic: controls which queue a completed transfer batch will be moved to. The job may be moved directly to the output queue of the present UT or directly to the input queue of the next downstream UT. The only variable to consider is the receiving queue length and the question of adequate space to accept the batch.

Operational Parameters

The operational parameters control the UT operation and the number of jobs allowed to reside within the queues at any one time.

Operational: a boolean variable; indicates whether or not the resource is able to accept work. This is normally set at the beginning of a period and will remain unchanged until the end.

Resource Efficiency: a percentile value set at 100% if the process is operating at normal speed. If a work centre, for example, has a 60% efficiency rating all processing times would be multiplied by a factor of 1.4, increasing processing times by 40%. In future research utilizing the software this parameter may be used to model a learning curve experienced when a employee begins working with a new set up.

Set Up Time: five different types of set up times are defined in the MSA simulator (see Chapter 4). When the databases are being configured (module msa 3), the relevant set up time for the method selected is copied to a field in the Sequencing Database. If the method selected is a "Resource Based Set Up", then it is the value from this field in the UT definition that is copied to the Sequencing Database.

Simultaneous Working: a UT may be used to model functional groups of resources, given the pre-requisite of all the work centres in the group having identical processing times for each individual job. For instance, rather than defining five UT's for five identical work centres, one UT is defined and the Simultaneous Working variable is set to five. It is then possible to load five jobs onto one UT process at any one time. Batch completion times are held for individual jobs in the Sequencing Database, thus the five jobs would not necessarily have to be loaded or completed at the same time.

Queue Lengths: three queue length variables (*Part, Component* and *Output*) determine the number of jobs that may be held at a UT at any one time. It has only a weak control, however, over the level of WIP inventory contained within a UT. For example, queue lengths could be set to one but a job in the queue might have a quantity of 300 parts, which could be physically impossible within the actual facility.

5.2.3 THE SEQUENCING DATABASE

The UT database (discussed above) contains the individual definitions of the available resources. The Sequencing Database contains a record for every part processed by a UT. For instance, if a component C1 visits two work centres before moving to a third for inclusion in a sub assembly SA1, C1 then has two entires in the Sequencing Database.

Sequencing Database data is sourced from the Jobs Pending file, the BoM file, and the software control file. The benefit of using this database is that all required information is in one central location in a consistent database. Therefore only one database access and one database write routine needs to be included in the software. This results in more open program code and

more efficient software at execution time.

Each record in the database consists of twenty one fields, (listed in Table 5.2) divided amongst four groups:

1. Simulation Control;
2. Product Description;
3. Batch Sizing;
4. Succeeding Product Information.

Two versions of the Sequencing Database are maintained: a master version and a run time version. The master version is created from the BoM file; with fields 15 through 19 (ref: Table 5.2) are left blank. There is one set of records for each product. In the run time version, created just prior to a simulation run (module msa 3), there is a set of records, with the blank fields initialised, for each job in the Work file. If two orders for the same product are required then two sets of the same records are copied. The only difference between the two sets of records is the record number, the job number, the quantity required, and in some cases the transfer batch size.

The following list describes the twenty one fields that constitute a record in the Sequencing Database.

Simulation Control

Record Number: represents the record's number in the list of records in the Sequencing Database.

Succeeding Record Number: identifies the database record of the component or sub assembly that the current item (identified by the record number) will become part of.

The above numbers are each five digits long and are used by the simulation driver to represent jobs in the model. To access information on a job, the first five digits are read, pointing to the relevant record in the Sequencing Database.

Product Description

Product Name: the name of the end item product that the component or sub assembly will eventually become part of.

Gateway Part: a boolean flag; indicates whether the component is an initial or gateway part and if the component should be placed in the relevant input queue when the job is launched from the work file. This field is required because no UT are specifically designated as gateway resources.

Work Centre: number of the work centre (ref: UT definition field 1) that will process the part of interest.

Start and *Finish Times:* used to record the times when jobs are held in the output queue for a period of time. An example of holding in the output queue is in paint manufacture when the product has to "settle" before being "filled" into containers.

Duration: time, in minutes, required to produce one item on the work centre indicated by the *work centre* field.

Part Name: name of the part to be processed.

Required: indicates the number of individual (not quantities of) components and sub assemblies required to produce one of the current part.

Simulation Control

1. record number
2. succeeding record number

Product Description

3. product name
4. gateway part
5. work centre
6. start time
7. duration
8. finish time
9. part name
10. required
11. obtained
12. component
13. set up time one
14. set up time two
15. job number

Batch Sizing

16. process batch remaining
17. transfer batch
18. transfer quantity
19. time to finish transfer batch

Succeeding Product Information

20. succeeding part
21. succeeding work centre

Table 5.2: Sequencing Database Fields

Obtained: indicates the number of required parts present in the input queues of the current resource.

Component: represents the quantity of the current part required by the succeeding resource in the production of one unit of the next, higher level part. This field should be distinguished from the required and obtained fields, which record whether at least one transfer batch of a required component is present in the input queues.

Job Number: a unique number, automatically assigned when the job inputs a new job into the Jobs Pending file. The same number is used for all components, sub assemblies, and the end item of the particular job.

Set up Time 1 and *2:* in the master version of the database, *set up time 1* holds the value for a product based set up time; *set up time 2* holds the value for a part/process dependent set up time. When the run-time version of the database is created the *set up time 1* field is used to record the actual set up time for the chosen method.

Batch Sizing

Process Batch Remaining: equivalent to the size of the total process batch required if none of the components have been processed. If a transfer batch of the part is completed, an amount equal to the size of the transfer batch is removed from this figure.

Transfer Batch: standard size of the transfer batch used for the component. Size of the transfer batch does not have to divide equally into the process batch; therefore the "transfer quantity" parameter is used.

Transfer Quantity: quantity of parts in the transfer batch currently loaded onto the work centre.

Time to Finish Transfer Batch: clock time required to finish the current set up or processing of the transfer batch.

Succeeding Part Information

For the sake of efficiency, information concerning the downstream part is included in the current record. The contents of the succeeding part and succeeding work centre fields can be determined by using the *succeeding record number* as a pointer to the next record. However, it is included in the current record because the data in the fields is accessed several times in the course of moving a job through a UT.

Succeeding Part: best described by an illustration: consider a part C1 that is processed by resources number 1 and 2, successively, before being moved to resource 3 for inclusion in part SA1. This field, in the record for C1 on resource 1, would contain C1; in the record for work centre 2 it would contain SA1.

Succeeding Work Centre: indicates a part's work centre destination when it has finished processing at its current work centre. This field in effect links the UT's into a network.

5.3 THE SIMULATION DRIVER

This section discusses the methodology of the simulation driver. The driver creates a powerful, flexible simulation tool by using the UT and Sequencing Database.

Three simulation frameworks or world views were discussed in detail in Chapter 2. The MSA uses an event scheduling approach to simulating jobs

through a facility. For the majority of manufacturing events, set up complete and processing complete occur at discrete points in time; these are used to drive the simulation clock (non-conditional events). Two non-conditional events and three conditional events are defined in the MSA simulator. The conditional events depend on the state of the model to determine if they are ready to be executed. The two non-conditional (time dependent) events included in the MSA simulator are:

1. set up complete;
2. processing complete.

All other events are conditional on the state of the project at each event time:

3. transfer between UT's;
4. launch jobs from the Work file; and
5. load job into process.

The state of the project does not change in between discrete event times, thus the firing conditions of 3, 4, and 5 are checked only at the event times. After all five events have been fathomed, two other simulation control activities occur:

6. update clock and performance criteria; and
7. iteration.

At each discrete event time the conditions of each event are only scanned once. Because each of the five defined events may change the state of the model, the order in which they are checked and fired is important. For example, although event 2 may change the variables that effect whether a later event will fire, the firing conditions of event 1 are not affected because of model design. The events are fathomed in the order they are listed above, 1 through 5.

The software does not maintain an event list. To determine the next event time the Sequencing Database is accessed. This avoids extra processing that would be required if doubly linked lists where used to hold *event type* flags or manipulating the list pointers when new events are spawned.

The simulator is implemented in seven modules (Sim 1 to 7). Sim 1 through 5 implement the five non-conditional and conditional events, Sim 6 updates the clock, and Sim 7 controls the end of period processing. Figure 5.2 is an overview of the modules link together; and Figures 5.3 through 5.8 illustrate the operational logic of the modules. Each of the modules, which together constitute the simulator, are described.

Each event is checked. If event 1 is being fathomed, the relevant conditions of all the defined UT's are checked. Once all UT's have been checked, the simulation driver moves to event 2 and the conditions are checked for all the UT's. This process continues until all the conditions for all events have been checked in turn.

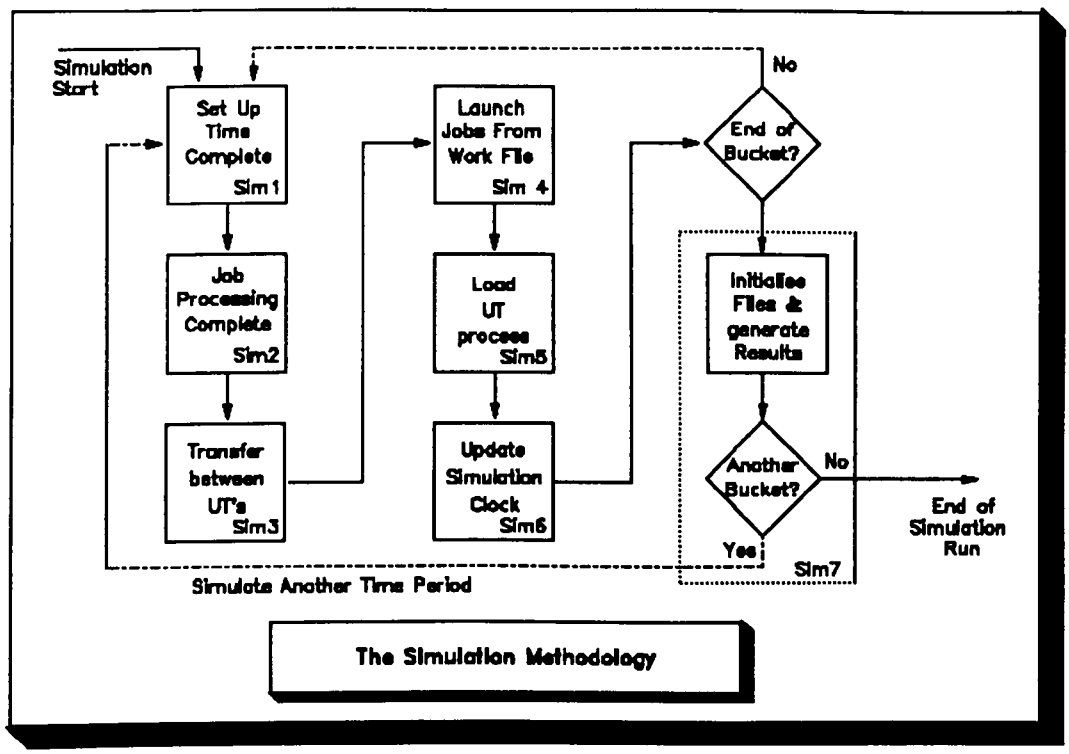


Figure 5.2: Overview of the Simulation Methodology

5.3.1 SET UP COMPLETE (Sim 1)

Figure 5.3 depicts the methodology of this module sim 1. This module is concerned only with UT's that have a "set up" status. The first step in the process determines the status of the UT.

The end of set up time has been calculated in module sim 5 and stored in the *completion time* field (field 19, ref: Table 5.2) of the Sequencing Database record. The value in the *completion time* field is compared to that of the simulation clock. If they are equal the set up is complete and processing of the part begins immediately. To generate the time for the Processing Complete event, the size of the *transfer batch* is multiplied by the *duration* (both fields are in the Sequencing Database). This gives the amount of time

required to complete the batch. Adding this value to the current value of the timer yields the event time, which is stored in the *completion time* field. The final action of this module is to change the status of the UT from "set up" to "working".

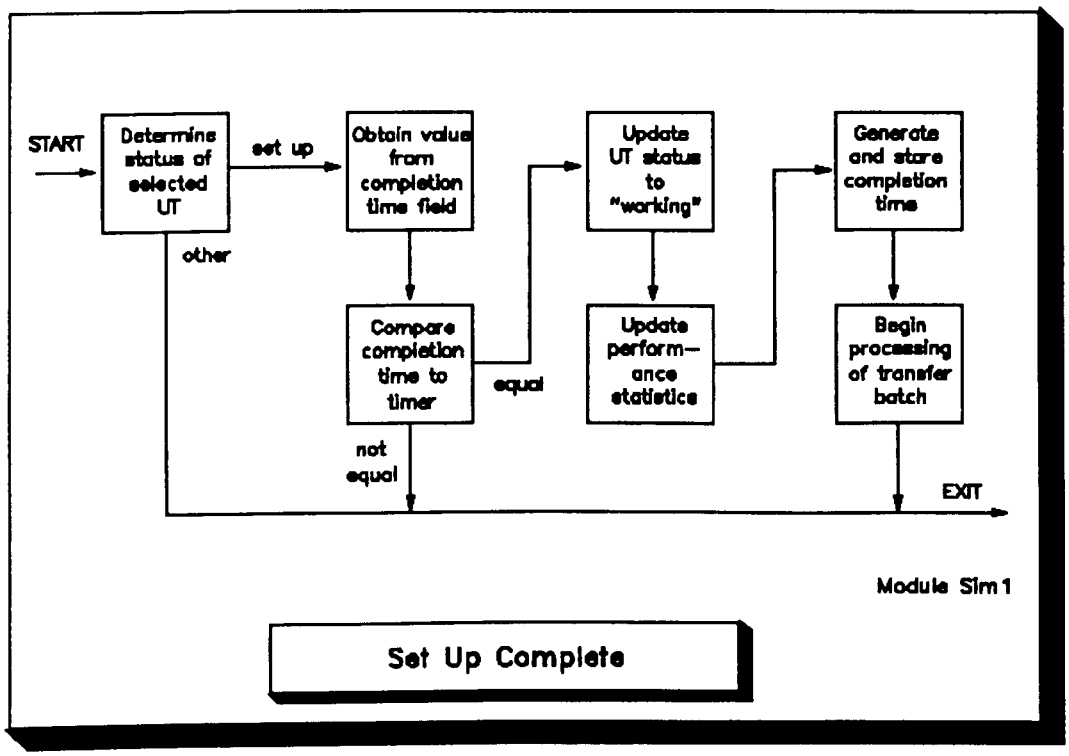


Figure 5.3: Set Up Complete Event

5.3.2 PROCESSING COMPLETE (sim 2)

The objective of this event is to determine if a job has completed processing and if so unload it from the work centre and move it to the receiving queue. Figure 5.4 illustrates these steps.

The first condition tested is whether the status of the UT is either "blocked" or "working". If any other status is encountered then the next UT is selected for

processing. A UT status of "blocked" implies a job that has completed processing at a previous event time but could not be unloaded because its receiving queue was full to capacity. If the UT status is "working", the Sequencing Database record for the currently loaded job is accessed and the value of the *completion time* field is compared to that of the simulation timer. If the two are equal the job has completed processing and is ready to move to the receiving queue.

To unload a completed batch or a previously "blocked" batch, available room to accept the job in the receiving queue is determined. If no room exists the UT status is set to "blocked" and the next UT is selected for checking. If there is room in the receiving queue the transfer batch is unloaded from the process and moved.

One of three situations and subsequent courses of action may exist after unloading a transfer batch from the process section of a UT

1. The process batch is incomplete but insufficient parts are available for a new transfer batch to be loaded. The UT status is set to "no work".
2. Sufficient parts are available and a new transfer batch is started; completion time is generated and stored and the UT status is set to "working".
3. The process batch is complete. The status of the UT is set to "idle".

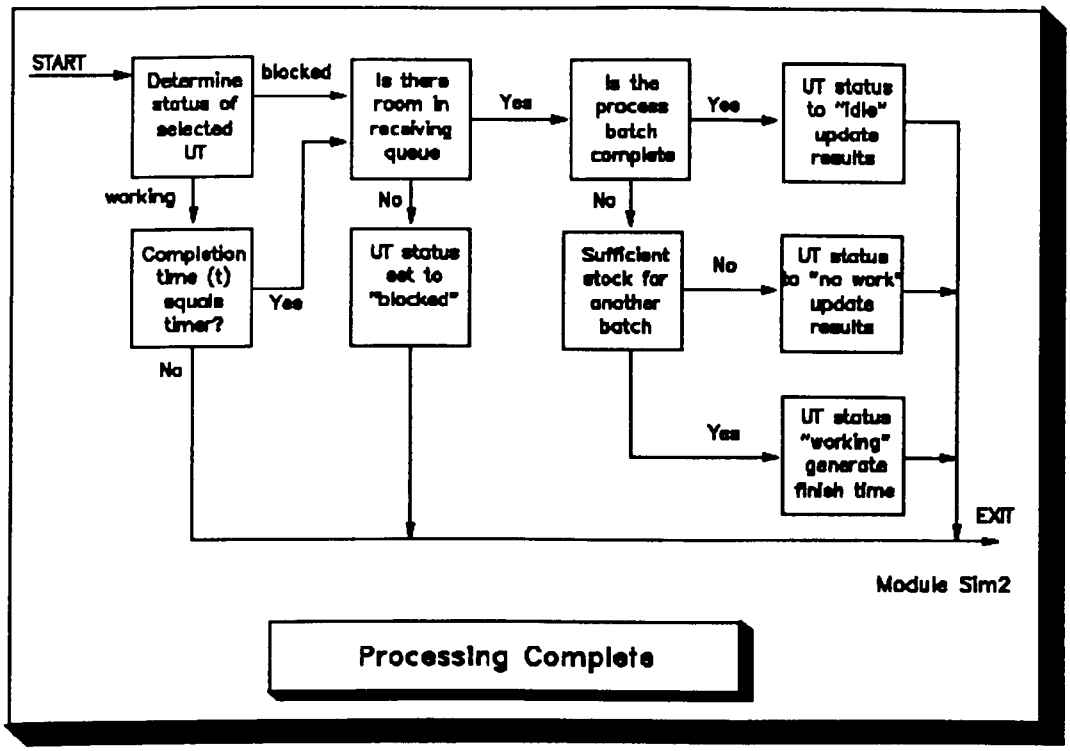


Figure 5.4: Processing Complete Event

5.3.3 TRANSFER BETWEEN UT's (sim 3)

The objective of module sim 3 (Figure 5.5) is to move jobs from the output queue to the input queue of the downstream work centre. When each event is fathomed, every job in all UT output queues is tested to determine if it can be moved. In applying this event to each job, it is determined if the UT from which the job was unloaded was the final processing stage. If so the job is removed from the model, the results database is updated with the job completion time, the job tardiness, and delivery error, and manufacturing error performance measures are calculated.

If the job requires more processing and there is sufficient space in the receiving queue (one of the input queues of the succeeding UT), the job is

transferred into that UT. Insufficient room results in the job remaining in the output queue until the next event time and having a status of "blocked".

If the model has been configured such that parts move directly from the work centre to the input queue of the downstream work centre then because the output queues will always remain empty, this module is bypassed

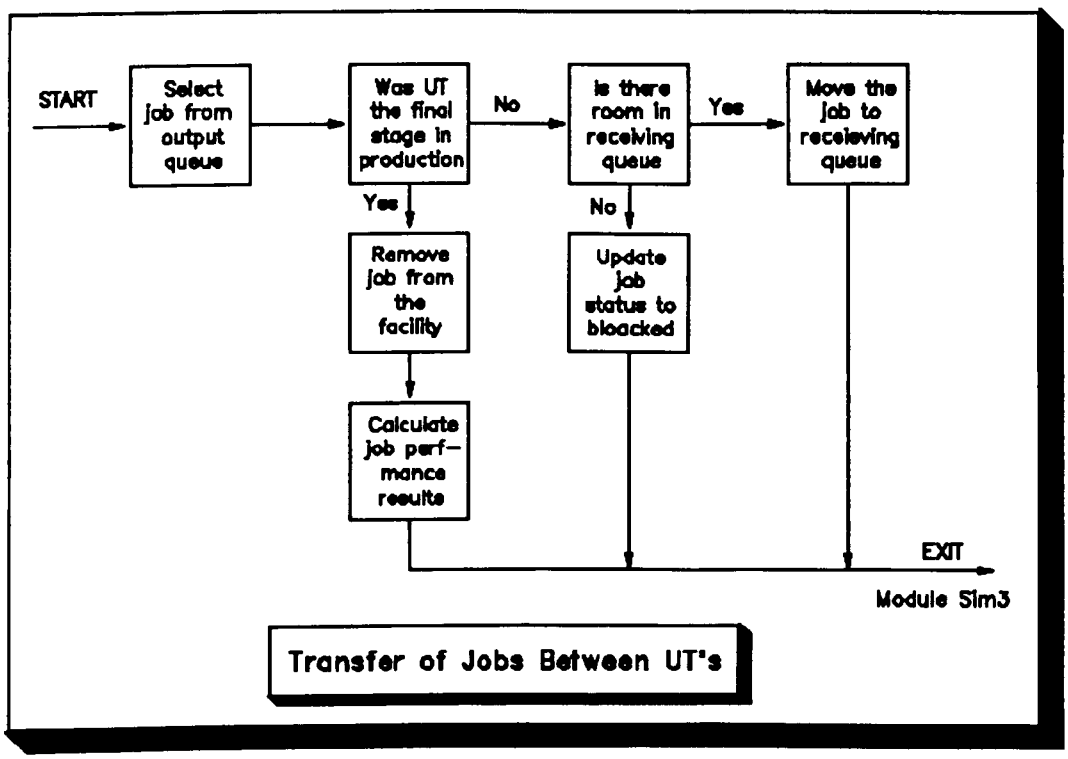


Figure 5.5: Transfer of Jobs Between Defined UT's

5.3.4 LAUNCH JOBS FROM THE WORK FILE (sim 4)

The jobs to be processed in the period under consideration are contained in the "Work" file, configured by module msa 3 (ref: Figure 3.1). The order in which the jobs appear in the "Work" file corresponds to the order in which they will be processed by the identified Critical Resource. The jobs are

selected for launching according to the strict order of the Work file. This process is illustrated in Figure 5.6. This order is a key consequence of the "critical resource scheduling" philosophy built in this research.

If the Work file is empty the module is bypassed; otherwise the first unscheduled job is selected. By cross referencing the Sequencing Database, all the gateway processes for the job in question are identified. If any of these gateway processes are non-operational or if the input queue is full to capacity, the status of the job in the work file is changed to "blocked" and none of the gateway processes are loaded with any of the job's initial components.

If all gateway processes are operational and there is room in their input queues, then all the initial components of the job in question are moved to the respective input queues of the Universal Transfers. The status of the job in the Work file is updated from "Not Scheduled" to "Launched". In the results database a record is then created containing the following information:

(i) job name;
(ii) job number;
(iii) launch time;
(iv) due date; and
(v) theoretical lead time.

The only calculation involved in this process determines the theoretical lead time (demonstrated in Chapter 4).

Once the first job has been "Launched" or "Blocked", the Work file is checked for the existence of any more jobs requiring launching. If such jobs are present the above process is executed again. The module is iterated through in this way until all jobs in the Work file have been checked.

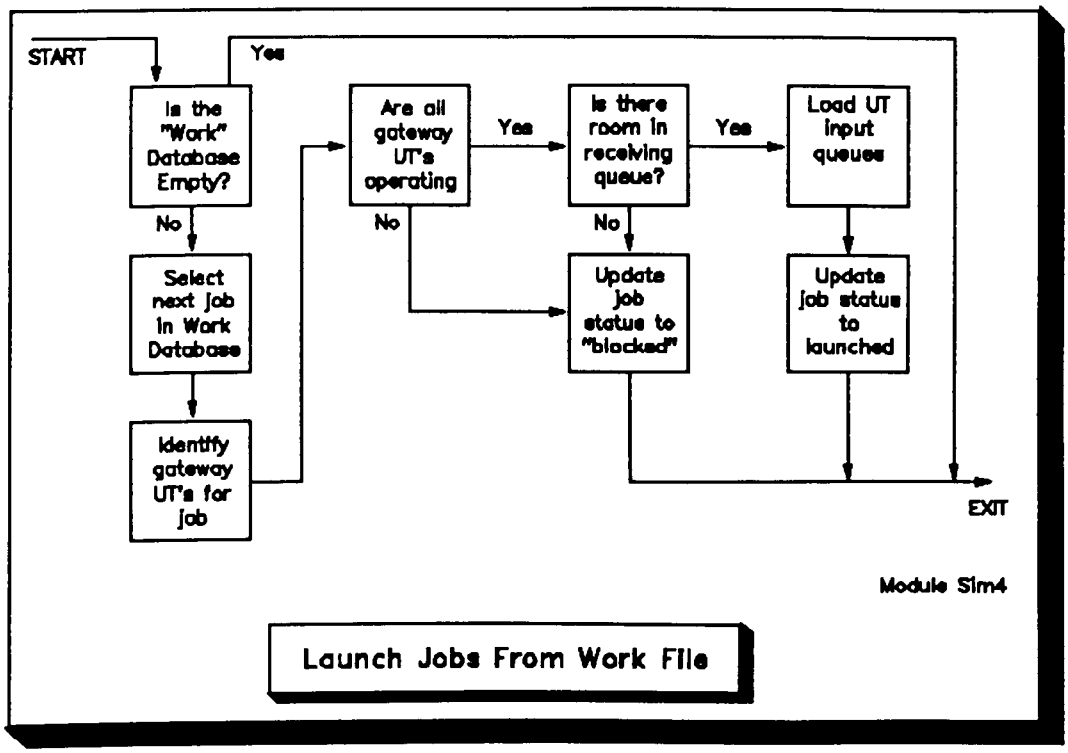


Figure 5.6: Launch Jobs From Work File

5.3.5 LOAD JOBS (sim 5)

Module sim 5 loads, if possible, jobs from the input queues onto the process section of a Universal Transfer. Figure 5.7 illustrates the method of selecting and loading jobs.

When this event is being fathomed, the simulation driver is concerned with UT's having a status of "Idle" or "No Work". Each UT is dealt with individually as illustrated in Figures 5.7a and 5.7b.

Status: Idle

An "Idle" status indicates that the process completed a job at a previous event

time but that no other jobs were available at that time for processing. This procedure (determining job availability, followed by job selection and loading) is illustrated in Figure 5.7a.

Because no job can be loaded if the input queues are empty, the next Universal Transfer is selected for consideration. If the input queues are non-empty, a check is made to determine if any jobs are available for loading, i.e., contain a full set of components in the quantities required for a transfer batch.

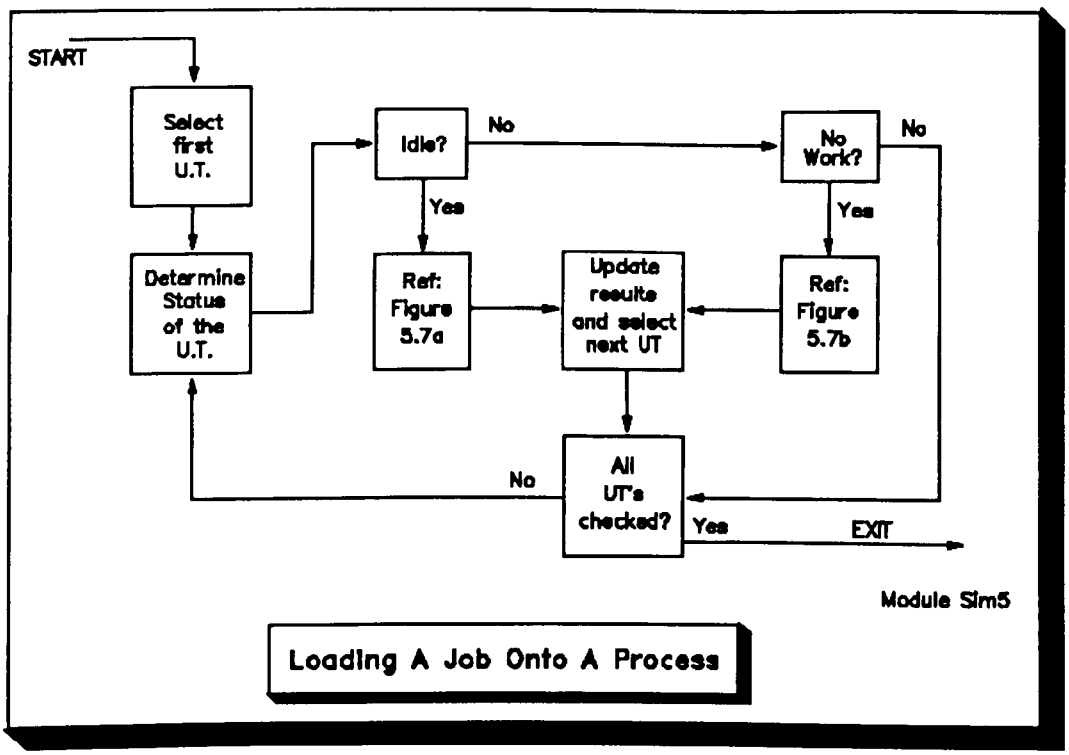


Figure 5.7: Loading a Job Onto a Process

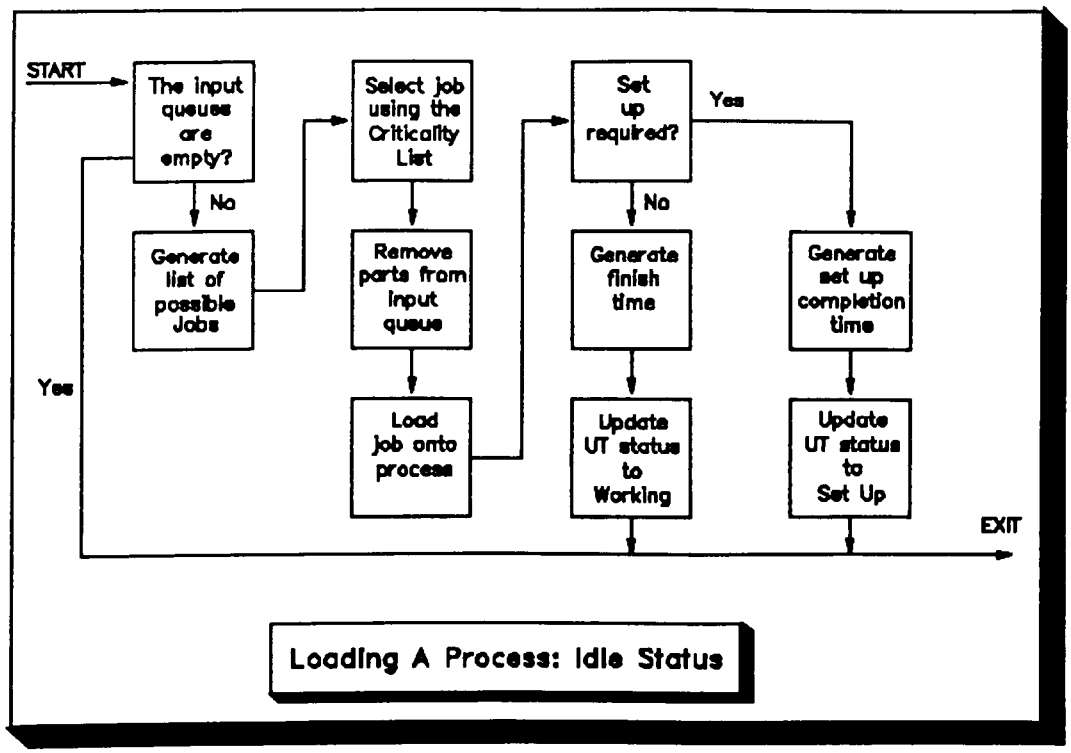


Figure 5.7a: Loading a Process With Status "Idle"

A list of available jobs is generated and compared to the Criticality List to ascertain which job appears highest on the list. This job is then selected for loading. If none of the available jobs are on the Criticality List the active dispatching rule is used to make a choice between those jobs available. Once a job has been selected a quantity of parts sufficient to produce a transfer batch is removed from the input queues.

If the model has been configured for "no set up", processing would then begin immediately. The completion time would be generated and stored, and the status of the UT updated to "working". If set up of the work centre is necessary, the time required for this procedure is stored in the *set up 1* field of the Sequencing Database record for the job to be loaded. To generate the set up complete time, the value in the *set up 1* field is added to the value of the

timer, and then stored in the *time to finish transfer batch* field.

Status: no work

The method of fathoming a UT with status "No Work" is shown in Figure 5.7b. "No Work" status indicates that the process completed work on a transfer batch at a previous event time, but a complete set of parts was not currently available to begin another transfer batch of the same job. At the event time being fathomed the input queue is again checked for availability of a complete set of parts to load for a transfer batch. If a complete set is available, the quantity of parts needed to produce the transfer batch is removed from the input queue. The process complete time is generated and stored, and the UT status is updated from "No Work" to "Working". The performance criteria recording the length of time of the UT's "No Work" status, is updated.

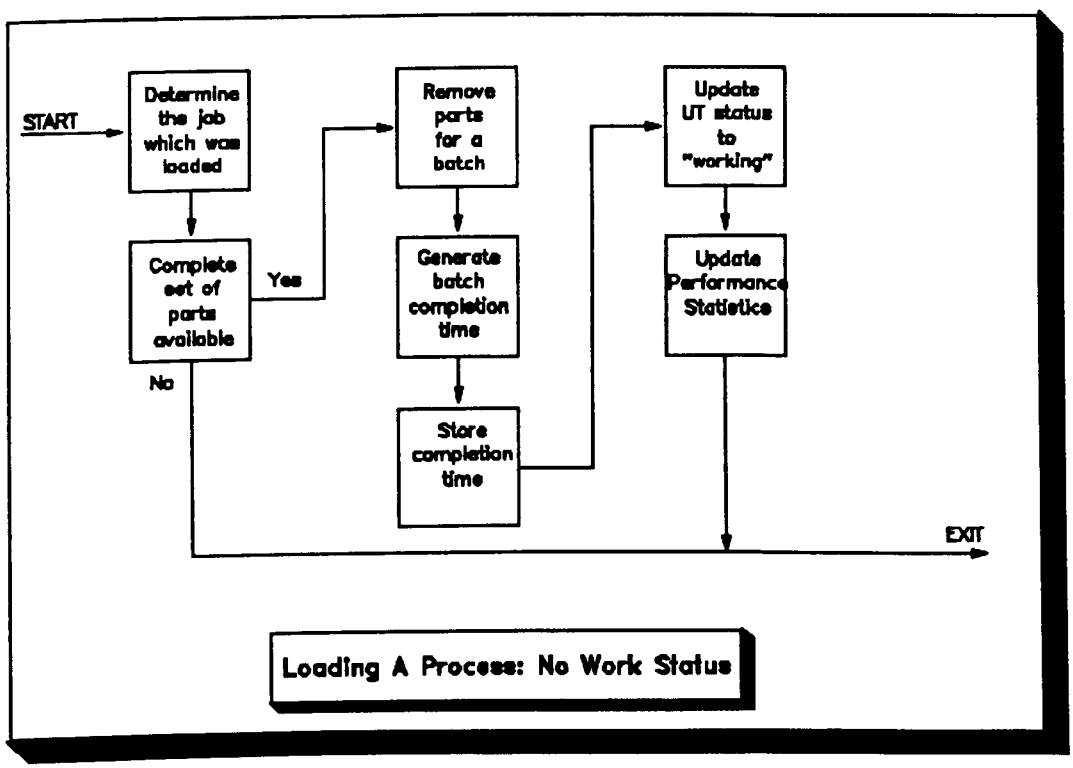


Figure 5.7b: Loading a Process With Status "No Work"

5.3.6 UPDATE CLOCK AND PERFORMANCE CRITERIA (sim 6)

No event list exists from which the simulator may generate the next event time. In the MSA program suite it is computationally more efficient to access the Sequencing Database and then examine the contents of the *time to complete transfer batch* fields. Figure 5.8 demonstrates the process of updating the simulation clock or timer in this manner.

To be updated the timer is to set to a high value which is always greater than any possible next event times of activities within the model.

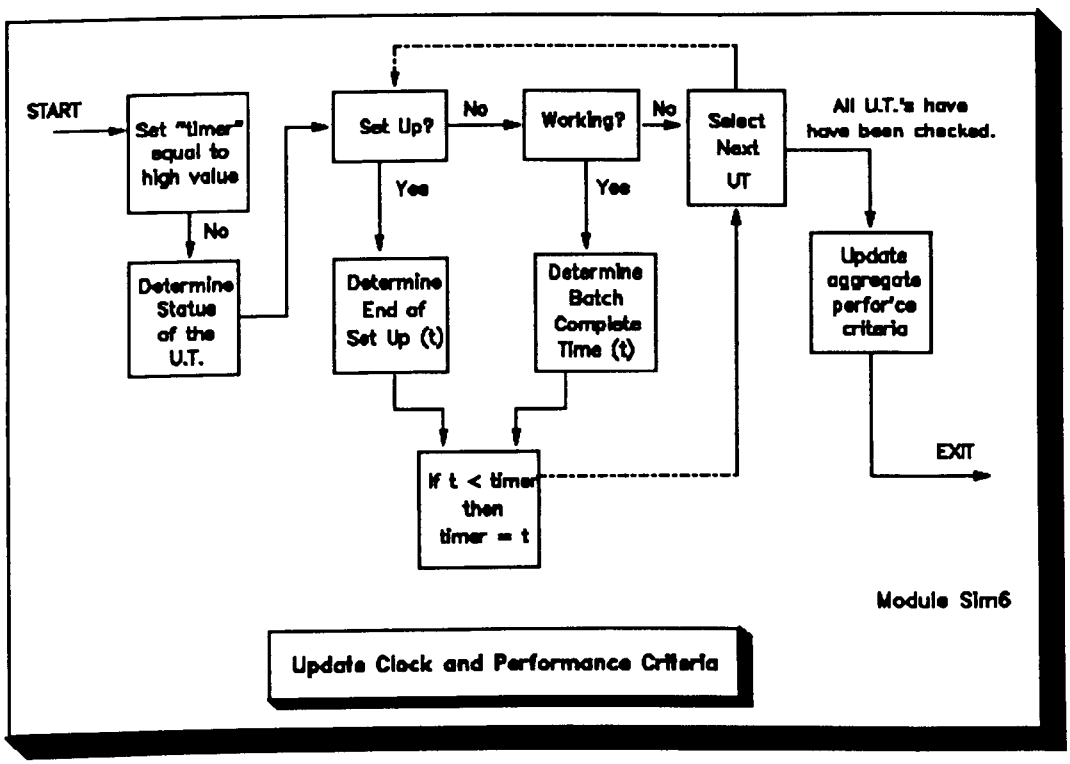


Figure 5.8: Update Clock and Performance Criteria

Only events 1 and 2 create event times. All other events are conditional on the state of the project. For this reason each UT is selected and its status determined; the UT's of interest have a status of "set-up" or "working". When such a UT is encountered the Sequencing Database is accessed. If the value of the *completion time* field is less than the value of the timer, the timer is given this value. After each UT has been checked the value of the timer variable will hold the time of the next event.

If the new value in the timer variable is less than the end of the time period, control is passed back to module sim 1 to again begin the process of fathoming the five events.

A special case exists when all the UT's are idle, the model contains no jobs, and the end of the period has not been reached.

In this instance no more jobs exist to be processed through the model. In this situation the value of the timer is updated to equal the time of the end of the period.

Another special condition exists when the next event time is greater than the time for the end of the period. Here the timer is set equal to the time for the end of the period.

If the value of the timer equals or is greater than the end of the time period then control is passed to module sim 7, *Iteration*.

5.3.7 ITERATION (sim 7)

The primary role of this module is to "clean" the databases for the next period.

There are three functions defined:

1. Remove completed jobs from the Work file and the run time version of the Sequencing Database.

2. Set the minute count to zero for the beginning of a new period. If a job overruns a period end, the *completion time* field in the Sequencing Database is updated to reflect this.

3. Update end of period aggregate statistics, for example, number of jobs graduated in the period, total tardiness, percentage tardiness.

5.4 Summary

This chapter has illustrated the simulation driver's implementation of the aspects of the sequencing heuristic, using the UT and Sequencing Databases.

Chapter 6 presents the programming approach used to implement the complete MSA methodology detailed in the previous three chapters.

CHAPTER SIX

THE MSA SOFTWARE SUITE

6.1 Introduction

The objectives of this chapter are to demonstrate:

1. The structured hierarchical nature of the software.
2. The modular approach adopted to software design and its advantages.
3. Open readable code achieved by using two generic programming models.
4. Functional groupings of the programs and the rules for defining the groups.
5. Briefly outline the programs and their use along with the database.

The MSA suite of programs is the software implementation of the generic simulator based Critical Resource scheduler outlined in chapters three, four and five.

The programs are in four functional *groups*: Control, Systems, Project and Simulation within which three *types* of programs are used: Primary, Support and Library. The supporting database consists of Software Control files and Project Files.

To implement the MSA methodology thirty one programs and seven procedure libraries have been created using approximately 13,000 lines of code. Seven System files control the operation of the software plus a further sixteen for every project defined with each simulation run from the root file set creating a further sixteen files. A maximum of 999 simulation runs (versions) are allowed off each project root file set.

6.1.1 PROGRAMMING LANGUAGE

The software is written using PowerBASIC, a high level structured form of the BASIC programming language. The primary reasons for using this language are, firstly, the programs are easily understood by any researcher who wishes to extend the research, secondly, PowerBASIC is a highly structured form of standard BASIC containing all the logic control of other high level languages like PASCAL, for example, case statements, while-loop, if-then-else and do-loop constructs.

An important feature of the language is the portability of the programs. The compiled code is executable on any IBM or IBM compatible computer with a hard disk running DOS 2.0 or later. The ability to build run time libraries of commonly used subroutines gives the capability to compile relatively compact executable files.

Only one program is compiled to a .EXE (executable) file, that is, only one directly executable program exists within the suite. All other programs which are compiled are done so to .PBC files, PowerBASIC chain files which are similar to the .EXE files but may only be accessed from within a PowerBASIC compiled .EXE file or other .PBC files. Having only one entry point into the software stops the user inadvertently booting up the software from the wrong program and maybe experiencing erratic program behaviour.

6.1.2 HARDWARE REQUIREMENTS

The MSA program suite is designed to be used on an IBM PC or compatible computer preferably with a colour graphics adapter, EGA, CGA, MCGA, or VGA.

The compiled code requires approximately 1.7 megabytes of hard disk memory and each set of project files needs from 0.1 to 2 megabytes depending on the size of the project, the number of periods simulated and the number of individual simulation runs. All programs and files are stored in one user created directory the name of which is left to the user to decide, it has no bearing on the operation of the software.

6.1.3 USING THE SOFTWARE

The default directory must contain the software, which is then launched from the DOS prompt by typing MSA <enter>.

The software is completely menu driven in order that an end user does not have to be familiar with which program to use or which file to load to achieve a certain goal. A goal of the software was to create a powerful simulation tool which was user friendly and this has been achieved by the use of more than thirty menus.

There are two types of screens defined in the software.

1. Menu Screens

The user is presented with a list of options from which a choice must be made, for example, from the Main Menu the user may select Project Data Management, Advance Simulation or Utilities. To manipulate project data the user will choose option 1, Project Data Management. Menus are denoted by a numbering scheme, the Main Menu being number "1", and via this numbering scheme the user can orientate themselves in the software. Selecting option 1 from the main menu causes the Project Data Management menu to be displayed, menu number 1.1.

2. Input Screens

On these screens the user is prompted for specific input, for example, after selecting "Create Project" from menu 1.1 the user is asked to input the three digit Project Code of the project to be created and a 21 character alphanumeric descriptor. Such screens are denoted by letter in this case screen 1.1.A and after the name has been input the Data Input menu is displayed, number 1.1.1, the A is dropped.

For any menu or input screen it is possible to select "P. Previous Menu" and move back to the menu on the level above, for example, from 1.1.1 to 1.1, intermediary input screens being ignored. If this move is going to result in a loss of unsaved data the user is alerted and given the option to cancel the command or save the data.

6.2 Programming Outline

A structured approach to the design of the software was implemented by defining two programming models. Programming models are generic outlines and rules on functionality, including:

- start up procedures when it is chained to;
- rules on how the program executes; and
- where in the code it can chain out.

The principal reason behind using Programming Models is to aid the writing/debugging process and allow anyone who may be extending the code to understand the program flow easier.

Two programming models on which the *primary* and *support* programs are based, where developed for use in this research. The library programs are simply collections of routines which are collected together in a library by functionality or usage (which group of programs will use them). During the following discussion about the program models, subroutine refers to both Sub-procedures and Functions.

Programming Model 1 applies to all programs except for the six *movement* programs (discussed later) included in the simulator which use model 2.

6.2.1 PROGRAMMING MODEL 1

Table 6.1 contains the Program Model 1. The first six lines are for information only below which are the *$include* statements, a PowerBASIC function for initialising required software libraries. All *chained to* programs require that the first line be

```
$include "common.pro"
```

in order that all variables retain their present value when the new program is "chained to". Without this line all variables used in the program will be treated as uninitialised and given a value of zero or a null string. Invariably each program uses some of the common routines included in program M-COM, hence the next line to incorporate the common routine library:

```
$include "m-com.pro".
```

The only executable statement in a program which is outside subroutines or libraries is:

```
call go
```

This statement activates the program control procedure *go* which calls the subroutines contained within the program to achieve the objectives of using the particular program. Chaining to other programs always occurs from inside go.

Before the program was chained to a control variable z was set and is used within the *case* statement inside *go* to select the portion of code to execute. In this way control of which portion of code is to be executed is achieved at the top of the program without complex Goto statements embedded in the code as in standard BASIC.

The case statement is embedded in an endless do-loop, that is, the condition for exiting the loop will never be met, such that the only way out of it is to "crash" out of it by chaining to another program. In each of the program segments of the case statement the z variable is first set to zero, this forces good programming style because z needs resetting before leaving the code segment executing.

It is possible to execute more than one segment with the case statement by setting the z variable and not issuing a chain statement, the do-loop (an endless loop) is executed again and a new program segment executed depending on the value of z.

```
Program [name]
Objectives of the program
Edition [number]     Version [number]
Date of last update [date]

Manufacturing Simulation and Analysis
Copyright S Hurley & Dr J Driscoll

$include "common.pro"
$include "m-com.pro"
$include [library program name]


SUBROUTINE GO

  $include "shared.pro"

  DO

    CASE z

      case = 01
        set z = 0
        procedure and function calls
        set z
        [chain to another program]

      case = 02
        set z = 0
        procedure and function calls
        set z
        [chain to another program]
          .
          .
          .

      case = else
        error handling routine

    END CASE

  LOOP

END SUBROUTINE GO

**** SUBROUTINE BLOCK ****

  Procedure and Function definitions.
```

Table 6.1: Programming Model 1

"Case else" is an error trapping routine to catch a situation where the z variable was not initialised when chaining from a different program. The error handling routine prints a message to the screen giving the value of z and the name of the current program before terminating execution of the software. Clearly this is a debugging tool and the end user will never encounter this situation.

Following *go* is the "Subroutine Block" containing the procedures and functions of the program. All the subroutines in a program have a common functionality, for example, processing the BoM file or the Job Pending file. In the "Subroutine Block" there is no "loose code", that is, all code contained within procedures or functions.

6.2.2 PROGRAMMING MODEL 2

Programming Model 2 (Table 6.2) only applies to the *movement* programs of the simulator and contain only two procedures: *go* and *move*. This model is similar to model 1, the first lines are user information followed by three include statements. M-SIMCOM are common routines which all the *movement* programs use for accessing data in the Sequencing Database and the UT definition file along with updating results when required. *Go* does not have the do-loop and case constructs of model 1 and also does not use the z control variable.

```
Program [name]
Objectives of the program
Edition [number]     Version [number]
Date of last update [date]

Manufacturing Simulation and Analysis
Copyright S Hurley & Dr J Driscoll

$include "common.pro"
$include "m-com.pro"
$include [library program name]

SUBROUTINE GO

  for i = 1 to resource count
    call movement(i)
  next i

  pass control to simulation driver

END SUBROUTINE


SUBROUTINE MOVE (i)

  initialise control variables and escape code

  DO

    extract mcl variable from UT definition
    x = two character string from mcl

    CASE x

      case = 01
          program segment 01
      case = 02
          program segment 02
            .
            .
      case = else
          case else
          error trapping

    END CASE

  LOOP UNTIL ESCAPE CODE IS POSITIVE

END SUBROUTINE
```

Table 6.2: Programming Model 2

When a *movement* program is accessed *go* is called to iterate through each of the defined UT's calling the *move* procedure each time. *Move* first initialises the control variables such as the escape code and extracts the *mcl* (movement control logic) variable from the relevant UT definition. The *mcl* variable is a character string (20 characters maximum) broken down into two character segments were each two character string dictates which program segment is executed within the case statement inside *move*. If *mcl* = "01" then program segment 1 is executed and then control is passed to the loop statement. If the escape code is negative then the do-loop construct is executed again by first extracting the next two characters from the *mcl* variable and executing the relevant program segment in the case statement. This process continues until a positive escape code is encountered when program flow is passed back to *go* to select the next UT.

If all UT's have been checked the last line in *go* passes program flow back to the simulation driver program.

When the UT is being defined, the user does not have to know the codes that make up the *mcl* variable but is asked a series of questions and from the answers supplied the software determines the *mcl* variable.

The modularity and flexibility of this approach is apparent in the method of calling program segments. It then becomes a relatively simple procedure to add in a new program segment and have it accessed by the *mcl* variable.

To illustrate how the *mcl* variable is used to add flexibility to the execution of the software, consider unloading a job from a work centre in a UT definition. Depending on the user set parameters the job may move to the output queue

or directly to the input queue of the downstream work centre. The user would have indicated which logic to use when the simulation configuration parameters where set before the simulation was run. Part of the transparent function of configuring the database (module msa 3, reference: Figure 3.1) is to set the value of the *mcl* variable in the UT definitions to reflect the logic chosen so that the correct pieces of code are called off in the correct order.

6.3 The MSA Programs Suite

6.3.1 PROGRAM TYPES

Three *types* of programs are defined in the MSA software suite and software control is achieved by "chaining" from one program to the next. "Chaining" passes total control from the *chained from* to the *chained to* program, for example, in Figure 6.1 program A *chains to* D and once D has completed processing it must chain to another program. Program A does not automatically regain control, program D would have to chain to it for this to be the case. Programs A and D are known as **primary** programs. Program D is analogous to program A and may also have programs it chains to before passing control onto another primary program.

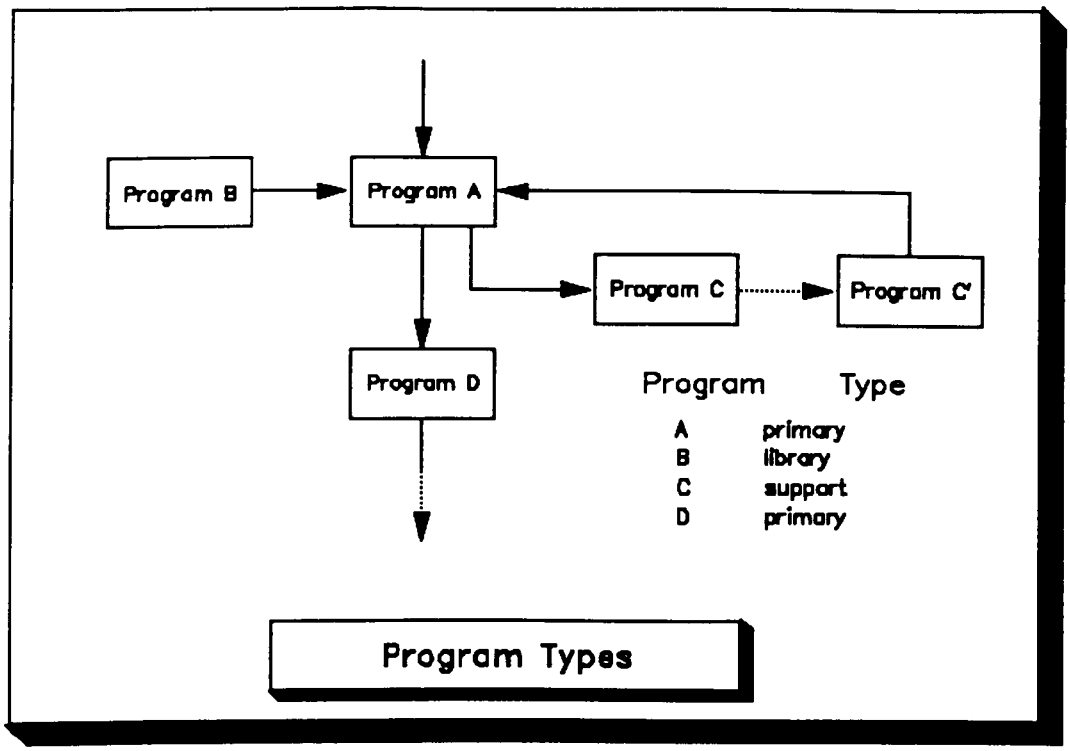


Figure 6.1: Program Types

A **support** program (program C) may be *chained to* by a primary or another support program and Program C may chain to other support programs but only to achieve a specific goal. After achieving the goal control is passed back to program A. For example, assume A is the main input program for defining a project and the user selects BoM input. Program C would create the environment for the user to input the BoM and after completing the input, control would be passed to another support program, program C', which would create the Sequencing Database entries. Once the BoM is created control is passed back to program A or to C to input another BoM. Related support programs may pass control amongst themselves as many times as is required but not pass control outside the functional subgroup.

Library programs such as program B are never chained to but are simply a

set of subroutines which a *primary* or *support* program uses. Library programs are never chained to but the routines are simply called off when required.

6.3.2 PROGRAM GROUPS

The program suite is divided into the four *groups*:

1. Control;
2. Systems Management;
3. Project Management; and
4. Simulation;

Figure 6.2 demonstrates the simple hierarchy of the four groups. Within the four groups there are primary, support and library programs.

The creation of functional groups, the rules on the size of the groups and the restrictions on which program may chain to which, was designed to keep the architecture simple and force the development process to follow a structured path.

Each program group is restricted to a maximum of four primary programs and any extensions to the software breaking this will tend to the need for the definition of a new group architecture. Each primary program is allowed to chain to a maximum of ten support programs. If this figure was any larger the code would become cumbersome and overly complex.

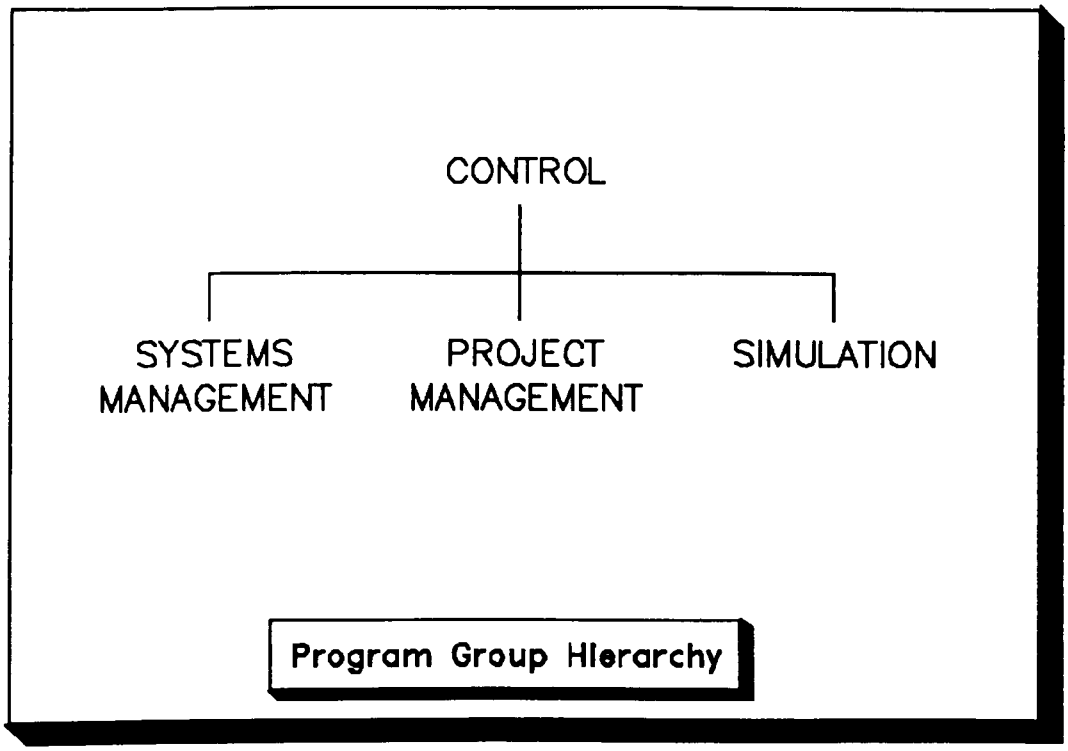


Figure 6.2: Program Group Hierarchy

The maximum number of library programs used by a group is restricted to four. Each library programs contains no more than fifteen procedures, each of these less than one hundred lines long. Again, during development, it was found that when these rules were violated the code became too complex and was therefore broken down into more manageable units.

Following is a brief description of the function of each group. Appendix B contains a full description of the role of each program within each group.

1. Group: Control

The function of programs in this group is to control the operation of the software.

2. Group: Systems

The function of the Systems Group programs is to allow the user to manipulate the seven system files. The system files contain variables which control the functionality of the software.

3. Group: Project

Outlined in Chapter 4 was the project methodology adopted to control the manipulation of user files. The programs in this group are used to create, load, edit, and delete projects.

4. Group: Simulation

The programs in this group provide the functionality for the simulation driver.

6.4 Summary

Chapter 3 outlines the MSA approach to simulating the sequencing of jobs through a manufacturing facility, Chapters 4, 5 and 6 detailed the sequencing approach, the simulation methodology and the structured software implementation.

The next requirement of the research programme is to design a structured test and evaluation programme. This follows in Chapter 7.

CHAPTER SEVEN

TEST CASE ANALYSIS OF THE MSA PHILOSOPHY

7.1 Introduction

Chapters 3, 4, 5 and 6 detailed the MSA philosophy, supporting methodology, models, and computer software. These topics are at the centre of the research described in this thesis. Chapter 7 reports on the results of the experimental test programme designed to first verify the software and secondly to test the effectiveness of the MSA methodology. The structure of the test programme, examined in detail throughout this chapter, is based on two types of test case model.

In the first part of the test programme, deterministic single machine examples are generated to test out the accuracy and performance of the extensive software suite supporting the MSA philosophy.

The second portion of the test programme examines the critical resource heuristic utilising single item BoM's with a variety of processing routes through the facility. This results in recommendations on the most effective version of the sequencing heuristic and progressive comments on the range of sequencing and critical resource selection models.

7.2 The Test Case Suite

The test case program consists of a theoretical set of problems generated from a review of previous academic work and experience of batch manufacturing scenarios. A structured set of theoretical problems is required in order to conduct a scientific verification of software and to examine parameters.

7.2.1 COMPARABLE PUBLISHED TEST PROGRAMS

Exhaustive detail of test data sets used in current published literature that concerning the scheduling of batch manufacturing facilities is not available, largely because of the bulk of such data sets. This lack of fully detailed test data eliminates any opportunity for direct comparative testing. Furthermore, the operating procedures of other approaches, as reported in the literature, vary. A review of the leading research, however, provides a guide to methods used by these researchers in constructing detailed test problems.

To remain as consistent as possible, and to support the concept of a structured test programme the main test programme parameters found in current published work are reviewed here. The main parameters in a batch scheduling test programme are identified as:

- number and size of BoM's;
- product routings;
- product processing times;
- product due date spectrum; and
- resources employed.

Bill of Materials

Fry et al. (1988) and Veral et al. (1990) used five multi-level BoM's. They attempted to generate BoM's that would negate the effects of queues by having each resource visited the same number of times. By restricting the capacity of a machine a bottleneck was artificially created.

Veral et al constructed five BoM's: one flat BoM (one main product, and many sub-components on a further level), one long BoM (one product and many sub-levels), and three BoM's between these extreme cases.

The majority of other research involving industrial scheduling has used single item BoM's; route length is generated from a probability distribution and the actual resources selected using a Uniform distribution (see Morton et al., 1988).

Two significant points on BoM structure emerge.

- First, the use of a simple BoM structure is acceptable for limited test groups.
- Second, a varied pattern of BoM's helps to avoid the generation of unrepresentative results in extensive testing.

Resources Employed and Production Requirements

With reference to the number of work centres, Harl and Ritzman (1985) used between five and ten machines, with randomly generated product routes being from one to three work centres long. A maximum of ten end items were defined. Methods of generating processing times included Normal Distributions (Fry et al.), Uniform Distribution (Goodwin and Weeks, 1986), or combining a log normally-distributed average processing time and a direct cost variable generated from a Uniform distribution (Morton et al., 1988).

The generation of due date varied across the research papers reviewed. Fry et al. employed a logical formulation:

$$dd = k(TWKCP) + r \qquad (7.1)$$

This represents the total work content on the critical path (TWKCP), multiplied by a constant k (actual k values used were 3 and 5) plus the job ready time

(r). Morton et al. (1988) generated due dates from a Uniform probability distribution:

$$dd = U[EFT, ms] \qquad (7.2)$$

where *EFT* is the earliest finish time (the earliest time that the job could be completed if there is no queuing for resources), *m* the makespan, and *s* a slack variable for generating a variety of due date tightness.

Significant points emerge on the four major parameters discussed:

- Random generation of product routing is acceptable.
- Product process times can be generated from a normal distribution.
- The relationship between due date and EFT is important and can be generated in a variety of alternative ways.
- The published test programmes limit the number of resources employed. This work will extend the size of problems to 20 resources.

A further reason for the inability to directly compare results is the significant number of permutations of assumptions used by each research group, i.e., including several different routes for one job, no set-up time allocation, no machine down time, overtime allowed, no dominant routes, etc.

From the summary of comparative academic published test programme the following primary parameters have been selected to generate the experimental data used in this chapter.

BoM type:	single and multi-level BoM's
Number of machines:	1 - 20
Number of products:	5 - 25
Number of routes:	5 - 25
Processing Time:	from Normal distribution
Due Date:	from Uniform distribution

7.2.3 TEST CASE DEFINITION

The test case suite addresses two standard problem types: single machine, and job shop.

The *Single Machine* problems are generated using deterministic and probabilistic variables to verify the operation of the software.

In the *Job Shop* the BoM's are single item with randomly generated routes. The flow of work is not unidirectional and jobs do not necessarily visit all resources before processing is completed. The simple job shop problem type is the most heavily utilised in this test programme.

7.3 Verification

In support of the MSA methodology a major suite of software (see Chapter 6) has been developed with 35 computer programmes and 16 data files per project. An important task is to verify the simulation, evaluation, and output routines built into the software.

Three consecutive methods are used in the verification process. First, the programs are created in a structured modular fashion. This facilitates the isolation of sections of code to enable extensive testing. Tests at this level consisted of extensive numbers of additional print statements that detail the operation of the software. These statements are then removed and the modular segments returned to the main code. Because of the ability to determine the accuracy of routines immediately, extensive recording of results is not employed.

The second method traces the actions of the software with deterministic test cases, using the built in reporting functions of the software. This verifies that results match expectations. One test case is used at this stage: test case VM1 (see Table 7.1), with one machine, five different products, deterministic processing times and a range of loading levels. Full trace actions where produced and manually confirmed.

Having confirmed numeric accuracy of the software, the third method verifies the software analytically, by comparing results to two known theorems.

The first theorem, states that using the SPT dispatching rule to sequence work through a single resource model will minimise flowtime. The second theorem states that the use of the EDD rule for sequencing work through a single machine will minimise the maximum job tardiness.

To perform the analytical verification four additional test problems (VM2 to VM5, Table 7.1) were developed and run with the four dispatching rules (random, FCFS, SPT and EDD). The results for flowtime are shown in Table 7.2; the results for job tardiness are shown in Table 7.3. Both tables confirm that the theorems are appropriately executed within the MSA software. The full verification test sequence is described in Appendix C. From the results obtained, the software is confirmed as accurate and acceptable for the remaining test programme.

Model Name	Processing Time	Quantity Demanded	Approx Loading
VM1	20	deterministic	50-150%
VM2	20	12	n/a
VM3	N(20,4)	N(18,3.6)	n/a
VM4	N(20,4)	N(30,6)	n/a
VM5	N(20,4)	N(48,9.6)	n/a

Table 7.1: Single Resource Verification Models

Model Name	Dispatching Rules			
	Random	FCFS	SPT	EDD
VM2	2520	2520	2520	2520
VM3	3476	3503	2997	3319
VM4	68011	69211	61146	67863
VM5	117826	118571	99504	119741

Table 7.2: Average Flow Time In Single Resource Model

Model Name	Dispatching Rules			
	Random	FCFS	SPT	EDD
VM3	333	555	414	304
VM4	6939	7019	6919	6759
VM5	15915	16015	15875	15724

Table 7.3 : Maximum Job Tardiness In Single Resource Model

7.4 Selection of Test Case Model

Having confirmed the reliability of the software, the test programme now moves to the investigation of the alternative test case models.

The model and parameter test programme are designed to address and explain:

- Working with steady state conditions;
- Stable and unstable results;
- Testing the critical resource identification and sequencing models; and
- Further analysis of the application of the "best" sequencing model.

7.4.1 STEADY STATE SIMULATION

Steady state is defined as a situation when the variabilities inherent in the simulation model are present but are no longer affected by the initial conditions (in this case an empty and idle facility).

As explained in Chapter 4, in MSA methodology new work is launched at the beginning of each time period and then subsequently simulated through a manufacturing period. At the start of production simulation, the model will be empty and idle. As a consequence, results based on accumulating values (eg. average throughput time) will not reflect real performance if the start-up period is included.

To overcome this, each simulation takes place in two parts: the start-up phase and steady state operation. In general, all simulation models pass through

the start-up phase (from t_0 to t_1) using randomly selected critical resources and a random dispatching rule.

Once steady state has been reached, specific critical resource selection and dispatching rule combinations are switched in for investigation. Simulation results are recorded on a period-by-period basis with no "accumulating results". Consequently, performance data between the pre-steady state and steady state remain separate, removing potentially misleading results from the pre-steady state periods.

The issue of determining a specific changeover period between pre-steady state and steady state has been resolved by conducting a statistical t-test and analysis of the coefficient of variance on the nine simulation runs discussed above. The results are given later in this chapter and discussed in some detail. At this juncture it is sufficient to state that the t-test identified period 70 as the beginning of steady state conditions, thus $t_1 = 69$.

Steady state is achieved by inputting a work programme requiring approximately 100% of capacity into each period. Alternatively loading could have been established at a higher level, (eg. 150%) and pre-steady state defined as the time period required to fully load all work centres. Jobs not already launched would then be removed from the input queue and the real job list started. This artificial "job swopping" approach was however discarded in this work in favour of the "work continuity" 100% loading approach.

7.4.2 STABLE AND UNSTABLE MANUFACTURE

In reviewing this test programme, one further concept is explored, that of stable and unstable production.

For a particular job with a given BoM and processing time distribution, there exists a theoretical minimum manufacturing time or theoretical lead time. When actual manufacturing lead time for that job becomes known (completion time minus launch time) then Manufacturing Error (ME), an important measure of efficiency and effectiveness can be calculated.

$$ME_i = ALT_i - tlt_i \qquad (7.3)$$

where ALT_i is actual lead time and tlt_i is theoretical lead time of job i. When simulating under steady state conditions, a widely varying, overall Manufacturing Error from period to period could be indicative of an over-simplistic test case, therefore would be ruled out as a potential test case. Such test cases are described as unstable. This phenomenon will be explained in some detail when this occurs in the test programme suite.

7.4.3 SELECTION OF CRITICAL RESOURCE AND SEQUENCING PROGRAMME

Structure of Test Sets

This section introduces the most significant portion of the test programme for evaluation of the MSA critical resource philosophy. As outlined in Table 7.4, testing the critical resource philosophy begins with nine problems constructed from three sizes of manufacturing facilities (5, 12 and 20 work centres) and three sizes of period-by-period job loads (5, 16 and 25 jobs per period). Variability in the models is obtained deriving the processing times from a normal distribution, with a mean of 20 and a coefficient of variation of 20%. Product demand is generated from a normal distribution with a mean of 20 and a coefficient of variation of 20%.

Problem Name	Number of Jobs	Number of Resources
1A1	5	5
1A3	16	5
1A5	25	5
1C1	5	12
1C3	16	12
1C5	25	12
1E1	5	20
1E3	16	20
1E5	25	20

Table 7.4: Outline of Test Cases Used To Investigate Stability and Steady State

Within each of these nine problems, a simulation is based on approximately 100% loading (total job times per period = capacity per period) and run for 100 time periods. This provides the estimated seventy period requirement needed to achieve steady state and an additional thirty periods for operating at steady state. The calculation of average quantity per batch is illustrated (using problem 1C3):

Number of Standard Minutes

12	machines
40	hours
480	standard hours
60	minutes per hour
28,800	standard minutes

Processing Time To Produce One Item

20	minute processing time
12	work centres visited/job
240	minutes

Average Capacity

$$\frac{28800}{240} = 120 \text{ items per week}$$

Average Number of Each of 16 Products Produced Per Week

$$\frac{120}{16} = 7.5$$

The calculated average batch sizes where calculated to load the facility at 100%. As a result of queuing times and resources occasionally being idle queue lengths tended to be constantly increasing. Therefore, by using the calculated batch size as a starting point batch sizes are reduced (under the conditions of variability previously described) until average queue size stabilises.

Initial Results

The results for the first nine tests are shown in the time series' depicted in Figures 7.1, 7.2 and 7.3. Results are also displayed in Table 7.5, where

average weekly manufacturing error is shown over the 100 period simulations, working on a progressive 10 period moving average.

The time series' indicate an unstable modeling situation in two of the five job test cases (1A1, 1C1) and all of the five resource test cases (1A1, 1A3, 1A5), reference Table 7.4. Although such a scenario may exist in reality such models are unsuitable for this initial investigation of the MSA heuristics.

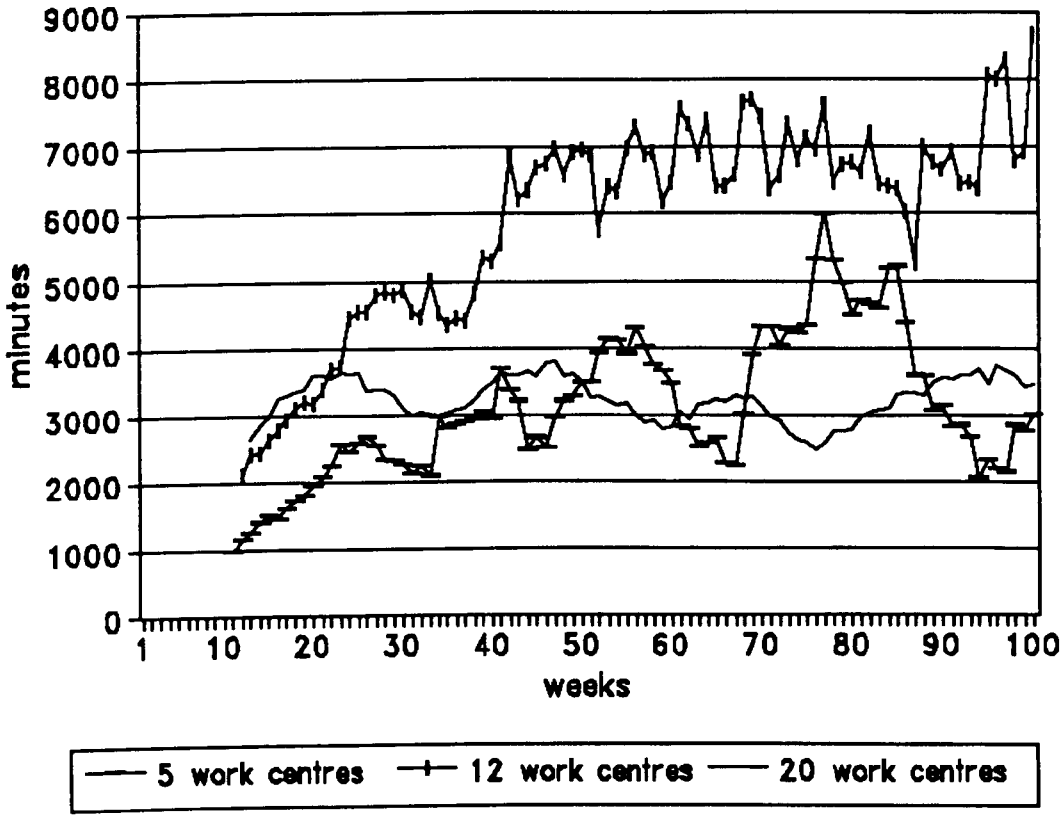


Figure 7.1 : Moving Average Manufacturing Error for 100 Periods with 5 Jobs

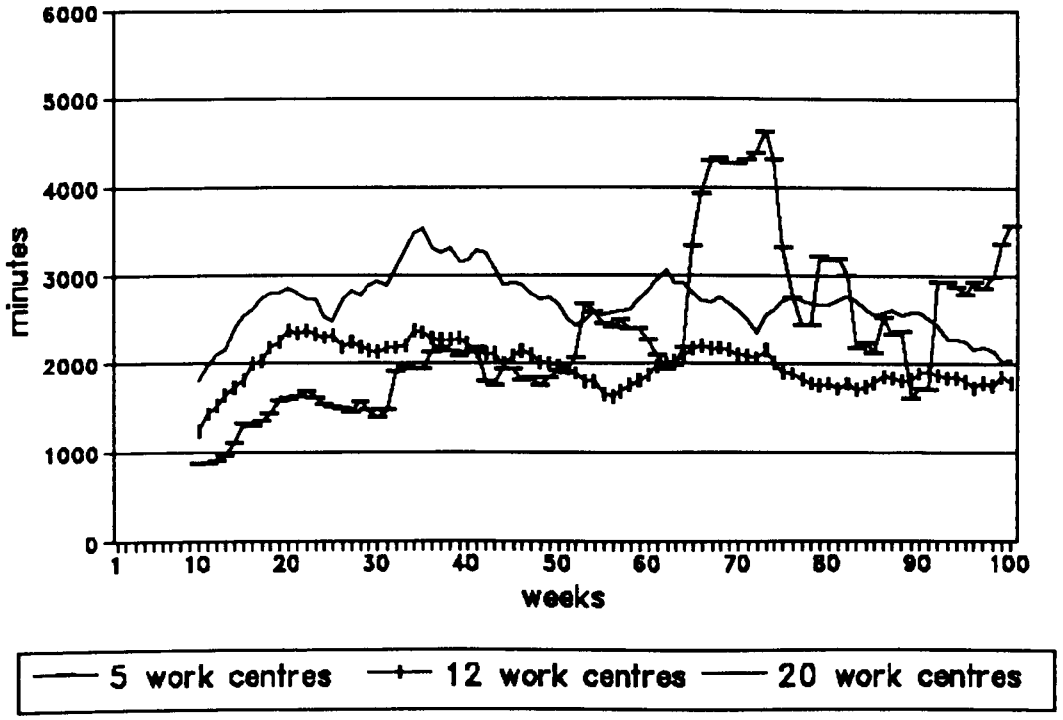


Figure 7.2 : Moving Average Manufacturing Error for 100 Periods with 16 Jobs

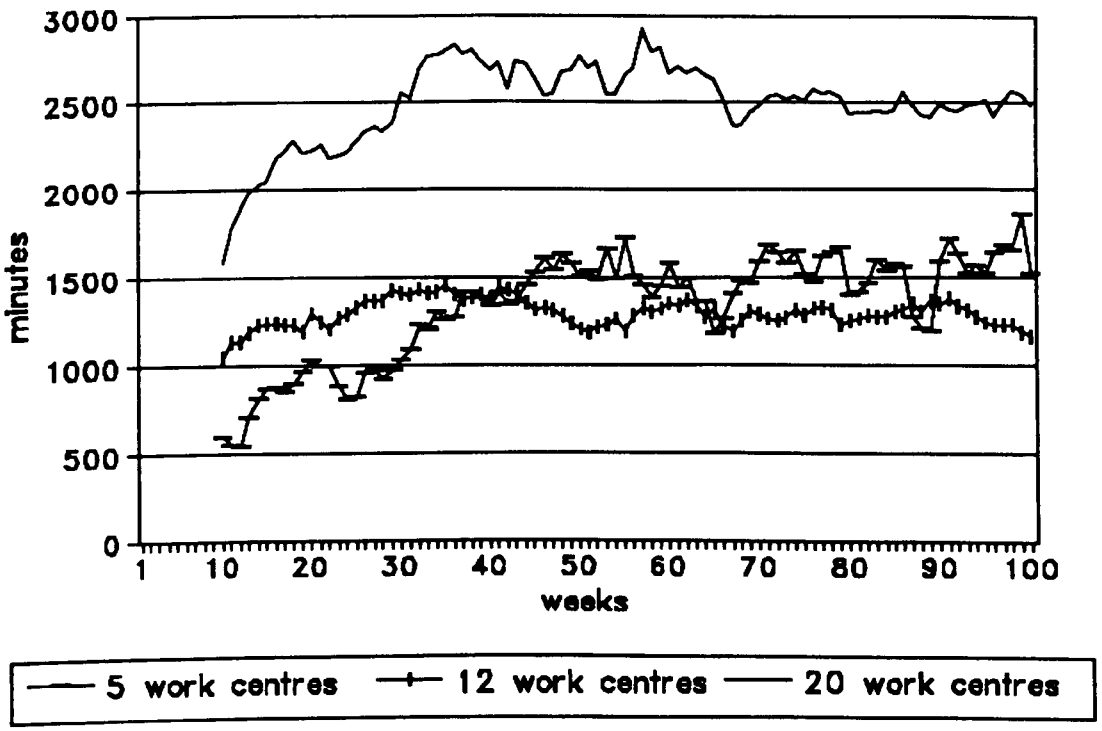


Figure 7.3 : Moving Average Manufacturing Error for 100 Periods with 25 Jobs

Using the last 30 periods, a 95% confidence interval $(1-\alpha = 0.95)$ is derived with 30 degrees of freedom:

$$(x-t_{n-1,1-\alpha/2}\, S/(n)^{\frac{1}{2}}, x+t_{n-1,1-\alpha/2}\, S/(n)^{\frac{1}{2}})$$

where S is the standard deviation of the sample defined by:

$$S = \left[\frac{\sum_{i=1}^{n} (x_i - x)^2}{n-1} \right]^{\frac{1}{2}}$$

A standard t-test has been performed on the data. The values for the time series average, of the Manufacturing Error for the last thirty periods (from all the models) fall within the t-test limits over than 90% of the time for all of the models. If this test is taken on its own, all models would be judged to be in steady state.

Further examination of the results reveals that the 20-facility 5-job problem, although relatively stable on the time series graph, proves to be too small a problem for use in the simulation. This conclusion is confirmed by investigating the standard deviation of the time series average of the Manufacturing Error (see Table 7.5) for periods 70 to 100. The coefficient of variation of problems 1A1, 1A3 and 1A5 in this period range from 63 to 109%, highly unstable. Problems 1C1 has a coefficient of variation of 74%, again unstable. Problems 1C3, 1E3, 1C5, and 1E5, however, have a coefficient of variation ranging from 16 to 22%, stable enough to be used for detailed analysis of the methodology.

Model Name	Number of jobs and resources	mean manu'g error	Standard Deviation	Coefficient of Variation
	5 jobs			
1A1	5 m/c	3554	2750	77
1C1	12 m/c	7365	5468	74
1E1	20 m/c	3261	818	25
	15 jobs			
1A3	5 m/c	2707	2944	109
1C3	12 m/c	1789	327	18
1E3	20 m/c	2363	509	22
	25 jobs			
1A5	5 m/c	1500	949	63
1C5	12 m/c	1253	203	16
1E5	20 m/c	2468	398	16

Table 7.5: Steady State Statistics

The instability of the small models is due to the fact that the processing times, generated from a normal distribution, are too heavily dependent on the value of the random number seed being used resulting in skewed mean and the variance values of the observations.

Thus, as a result of inherent instability, five of the nine scenarios are unsuitable for further analysis. Of the remaining four (1C3, 1E3, 1C5 and 1E5) are the most attractive for detailed investigation; the problem selected was the 12 work centre, 16 job case (1C3). The other three stable simulations would potentially be beyond the memory of the current computer software and available CPU time. Furthermore, although judged to be in steady state with respect to the t-test and the coefficient of variation, the 20 work centre, 16 job

case appears to show a downward trend in the time series (see the graph in Figure 7.2) and therefore can also be discounted.

Detailed Test Programme

For the selected 12 work centre, 16 job problem a full investigation of the sixteen critical resource selection-dispatching rule combinations is executed, with five levels of loading and two types of transient loading (200 and 300%). 200% transient loading consists of an additional 100% loading in the single period 101 and then a continuation of steady state (100%) inputs up to period 130. 300% transient loading is identical to the 200% version but with 200% extra work input in period 101. Since steady state inputs are maintained for periods 102-130 no spare capacity is available to remove this congestion.

The full test sequence (illustrated in Table 7.6) contains 112 test cases from all combinations of seven loading levels, four critical resource identification methods, and four dispatching rules (7x4x4 = 112). The 112 test cases are replicated three times with three different random number seeds to generate the processing times and due dates. This produces 336 test case simulations in this initial investigation of the MSA methodology.

Random Generator (3)	Test Loading (7)	Heuristic CR (4)	DR (4)
First seed	40	random	random
Second seed	55	LM1	FCFS
Third seed	70	LM2	SPT
	85	QM1	EDD
	100		
	transient overload		
	+100%		
	+200%		

Table 7.6: Outline of Test Cases

There are two parts of interest when examining the effectiveness of the MSA critical resource scheduling approach: the techniques of critical resource identification and the benefits of using the critical resource as the focus for developing efficient schedules.

As identified in the initial test programme, steady state is judged to begin at week 70. However, to be confident of steady state and to ensure that the jobs being worked on by the facility are introduced during steady state, the simulation is continued until the end of the week 100. The evaluation of MSA models starts at this point. The results reported in the rest of this chapter are from running the simulation a further 30 periods to period 130. The demand and due date patterns are reproduced exactly for each replication and the due date offset is set at 400% of the total work content of the critical path. On average, 500 jobs are completed within the 30 week continuation of steady state modeling.

The results, shown in detail in Appendix D and discussed in the remainder of this chapter, allow a five week "grace period" for the influence of the critical resource selection model and dispatching rule to take effect. The effective time over which results are collected is therefore period 105-130.

Performance Measures and Benchmark Test

The MSA software evaluates 37 performance measures for each period simulated. In line with current research, the six principal performance measures used in this analysis are as follows:

Effectiveness:

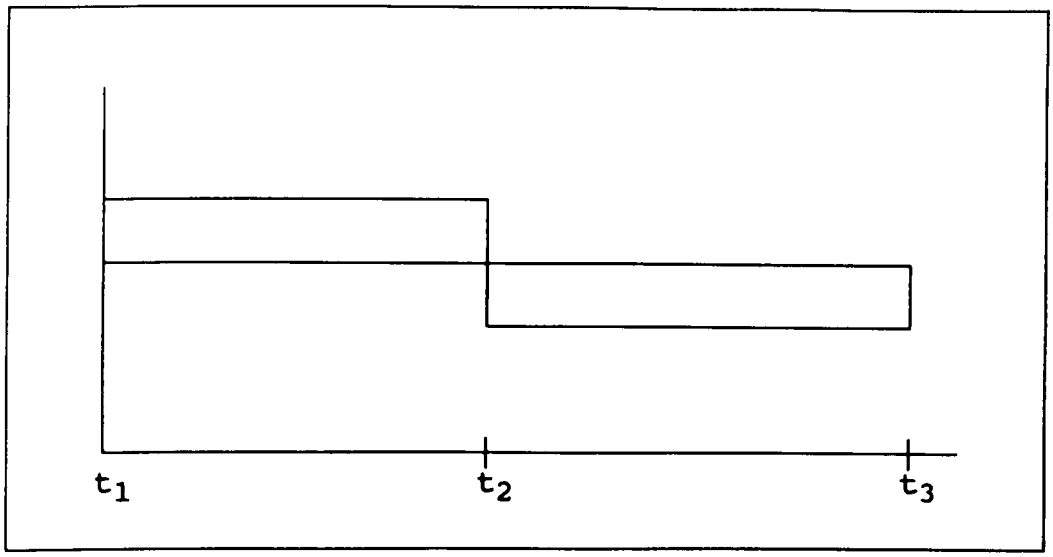
1. mean flow time (1) and (2);
2. total manufacturing error;
3. mean earliness;

Customer Satisfaction:

4. mean tardiness;
5. percentage tardy;
6. mean delivery error.

Mean Flow Time (1) is calculated at the point when the job is released to the facility; Flow Time (2) is calculated when a job begins processing. This point is illustrated in Figure 7.4:

Note that a tardy job is one that has positive lateness, i.e., it is completed after the due date. Delivery error is the absolute value of the lateness of jobs, both early and late.



t_1 = order release time
t_2 = begin processing
t_3 = completion time

Mean Flow Time (1) = $t_3 - t_1$
Mean Flow Time (2) = $t_3 - t_2$

Figure 7.4 : Illustration of Evaluation of Mean Flow Time Criteria

The benchmark used is the random-random approach: random choice of the critical resource and random ordering of the criticality list. In the majority, if not all, of the cases, use of the other sequencing heuristics should outperform the benchmarks.

7.5 Critical Resource Identification and Sequencing Test Results

The results of 112 primary and 35 secondary tests, each replicated three times, are reported in Appendix D. The layout of the results in the Appendix, and the structure of the following review of the results, is in the following:

- Overall summary;
- Summary by loading and evaluation criteria;
- Individual Criteria Results;
- Time average queue length results;
- Testing the critical resource selection method; and
- Restricting the queue size.

7.5.1 OVERALL SUMMARY

Two questions are to be answered at this point; can any of the three CR models surpass random selection and secondly can any of the three dispatching rules surpass random ordering.

Tables 7.7 and 7.8 illustrates the overall performance for the critical resource selection and the dispatching rules respectively. The tables are generated in the following manner:

1. There are sixteen combinations of selection method and dispatching rule. Each combination was simulated through each of the seven levels of loading, seven performance results were collated per selection/dispatching combination. These basic results are reported in Tables D.14 through D.27 (D = Tables in Appendix D).

2. At each loading level each performance criteria was averaged and ranked with respect to the selection and dispatching rule. These averaged results are reported in D.7 through D.13.

3. Tables D.3 through D.6 were generated by summing the ranks from Tables D.7 through D.13. These cumulative ranks are again ranked.

a. Table D.3 is generated by summing all the ranks for all the performance measures, at each level loading.

b. Table D.4 sums the rank of each performance criteria at all levels of loading.

c. Table D.5 is the average percentage deviation from the value for random (both dispatching and selection), summed for each level of loading.

d. Table D.6 is the average percentage deviation from the value for random (both dispatching and selection), summed for each performance criteria at all levels of loading.

4. Tables D.3 through D.6 contain an eighth "Totals" column. The percentage deviation of the totals from random are reported in Tables 7.7 and 7.8.

The overall summary performance of the Critical Resource selection models is given in Table 7.7. When examining the average rank achieved, all three models appear out performed by random choice. The results, however, are cumulative results from a series of tests with considerable number of marginal results hence Table 7.7 does not distinguish the extent of performance. When examining comparative performance, two of the Critical Resource Selection models out performed random selection; and one model in particular, the LM2 dynamic loading model performed very well, averaging a 16 percent improvement across all tests over the random choice.

The LM2 model for Critical Resource selection is therefore considered a contribution to the state of art in "bottleneck scheduling" work.

Table 7.8 shows the performance overall of three dispatching rules employed in this research. In this analysis there is little dispute that the SPT (shortest processing time) rule was a significant winner in all four selection approaches; across the seven levels of production loading, and across the seven evaluation criteria by rank and percent improvement. On average SPT outperformed random ordering by 11.4 percent each test.

The overall summary analysis therefore obviously identifies the LM2-SPT production scheduling approach as the combination that can best produce overall gains in manufacturing effectiveness.

	Ranking From Random			
	Loading		Criteria	
LM1	-14	(2)	-13	(3)
LM2	-9	(1)	-8	(1)
QM1	-42	(3)	-37	(2)

	Percent Improvement From Random			
	Loading		Criteria	
LM1	-19	(3)	-20	(3)
LM2	+112	(1)	+112	(1)
QM1	+36	(2)	+31	(2)

Table 7.7 : Effectiveness of Critical Resource Models Against Random

	Ranking From Random	
	Loading	Criteria
FCFS	+25 (2)	+25 (3)
SPT	+71 (1)	+66 (1)
EDD	+32 (3)	+32 (2)

	Percent Improvement From Random	
	Loading	Criteria
FCFS	+34 (3)	+49 (2)
SPT	+80 (1)	+79 (1)
EDD	+40 (2)	+40 (3)

Table 7.8 : Effectiveness of Dispatching Rule Against Random

7.5.2 SUMMARY BY PROBLEM LOADING AND EVALUATION CRITERIA

Problem Loading

Consider the range of "loading levels" as falling into three categories; underloading (40, 55, 70 and 85%), steady state loading (100%) and transient overloading (200 and 300%). The question then arises is there a variation in performance for critical resource selection model and dispatching rule across the three categories of loading.

Concentrating on percentage differences as shown in Table 7.9 (Table D.4 Appendix D) there is no significant difference between random critical resource selection and the three structured critical resource selection models, up to overload levels. This is explained by the potential for excess capacity to even out performance. However, in the overload models the LM2 and QM1 models considerably outperformed random critical resource selection. This is a key point to note: when the manufacturing facility is under heavy loading the value of the non-random models emerge.

	40%		55%		70%		85%	
	value	rank	value	rank	value	rank	value	rank
DISPATCHING RULE								
R-FCFS	18	2=	0	3	2	2	-3	2
R-SPT	19	1	16	2	11	1	13	1
R-EDD	18	2=	19	1	-3	3	-11	3
CRITICAL RESOURCE IDENTIFICATION METHOD								
R-LM1	0	1=	-7	2	3	2	-4	2
R-LM2	0	1=	-1	1	0	3	-1	1
R-QM1	-1	2	-17	3	6	1	-6	3

	100%		200%		400%		Totals	
	value	rank	value	rank	value	rank	value	rank
DISPATCHING RULE								
R-FCFS	4	2	6	3	7	1	34	3
R-SPT	6	1	13	1	2	3	80	1
R-EDD	5	3	9	2	3	2	40	2
CRITICAL RESOURCE IDENTIFICATION METHOD								
R-LM1	-11	3	9	3	-9	3	19	3
R-LM2	-3	2	33	1	84	1	112	1
R-QM1	2	1	16	2	36	2	36	2

Table 7.9: The Percentage Difference Of All The Performance Measures From the Random Value Averaged For Each Level Of Loading

With reference to Table 7.9, the LM2 method for critical resource selection clearly performs best of the four methods examined in any overloaded work programme.

With respect to dispatching rule SPT was the top performer in the underloaded and normal categories. In the 100 and 200% loading cases the SPT was overall the best approach but noticeably deteriorated at the 300% loading level. No real conclusion can be drawn from this one low performance but it does identify an interesting area of future work with a more extensive study of critical resource scheduling with extensive overloaded production programmes.

Evaluation Criteria

The seven performance criteria used may be classified as: three manufacturing related criteria (Flowtime (1) and (2), and mean manufacturing error) and four customer related criteria (mean earliness, mean tardiness, percentage tardy and mean delivery error). The percentage difference from random for each criteria averaged over all levels of loading is shown in Table 7.10. For the critical resource selection model, the three manufacturing criteria showed no significant difference between all models. For customer performance good mean early results were recorded for the two models LM2 and QM1. No conclusive difference is evident for critical resource selection models over all the customer performance criteria.

With reference to dispatching rules SPT based sequencing heuristics was ranked first in 5 of the 7 performance criteria, ranking second in the other two.

The significance of each critical resource selection models and dispatching rule with respect to specific evaluation criteria is examined in more detail in the next section.

	mean flow time(1)		mean flow time(2)		mean early		mean manuf'g error	
	value	rank	value	rank	value	rank	value	rank
DISPATCHING RULE								
R-FCFS	2	2=	1	3	2	2	20	1
R-SPT	4	1	4	1	15	1	7	2
R-EDD	2	2=	2	2	-17	3	3	3
CRITICAL RESOURCE IDENTIFICATION METHOD								
R-LM1	-1	1	-2	2=	4	3	-1	1
R-LM2	-2	2	-1	1	129	1	-2	2
R-QM1	-3	3	-2	2=	63	2	-4	3

	mean tardy time(1)		percentage jobs tardy		mean delivery error		total	
	value	rank	value	rank	value	rank	value	rank
DISPATCHING RULE								
R-FCFS	20	2	0	3	4	3	49	2
R-SPT	33	1=	11	1	5	2	79	1
R-EDD	33	1=	8	2	9	1	40	3
CRITICAL RESOURCE IDENTIFICATION METHOD								
R-LM1	-10	2	-7	3	-3	1	-20	3
R-LM2	-4	1	2	2	-10	2=	112	1
R-QM1	-18	3	5	1	-10	2=	31	2

Table 7.10: The Percentage Difference From Random of Each Performance Measure Averaged Over All the Levels of Loading

7.5.3 CRITERIA SPECIFIC RESULTS

Tables D.7 through D.13 are the average results for each Critical Resource selection method and dispatching rule at each of the seven levels of loading. This section discusses important aspects of these results.

Mean Flowtime (1) and (2)

Tables D.7 and D.8 reported the aggregated values for mean flowtime (1) and (2) for each level of loading. Mentioned earlier was that the mean flowtime measure could be used to indicate the levels of WIP within a facility, the lower the mean flowtime, the lower the WIP inventory level.

In the static single resource model the use of the SPT dispatching rule is credited with producing the lowest mean flow time value of any dispatching rule. When SPT is used as part of the MSA sequencing heuristic for loadings 40% through 100% it exhibited the same properties. For transient loading of 200% and 300% the effectiveness of the SPT rule declined, especially at the 300% level. By comparison, earliest due date, mentioned by Panwalker and Iskander (1977) and Goodwin and Weeks (1986), due date based rules performed well in heavily loaded shops with respect to flowtime criteria. This is borne out by the EDD rule achieving the highest ranking in the facilities loaded at 200 and 300%.

Customer Satisfaction

No provision was included within the methodology for attempting to complete jobs as close to the due date as possible, that is, minimising the absolute value of lateness. The methodology is designed only to produce what is required as quickly and efficiently as possible with due dates only being considered when the EDD dispatching rule is used. Four evaluation criteria

were used to judge the due date performance under the seven levels of loading: average mean early, mean tardiness, percentage number of jobs tardy and mean delivery error. The cumulative value for each criteria with respect to each dispatching rule and each critical resource selection method are included in Tables D.9, D.11, D.12, and D.13 respectively.

With reference to the tables both the SPT and EDD rules are clearly superior. Consider percentage number of jobs tardy and average mean earliness. The SPT rule is a clearly produces the best results in 6 out of 7 loading models for both criteria. With respect to the mean tardiness performance measure SPT is only the "best" rule in an underloaded shop (40-85% loading) whereas in the 100, 200 and 300% loading situations models EDD produced the most significant results.

Delivery Error (Table D.13) is a measure of absolute lateness, the EDD rule was superior in 5 of the 7 loading models, ranking second in the other two cases.

Analysis of the critical resource selection method results on the loading levels of 40 to 85% produced insignificant results with no definite superior method. The conclusion drawn from this is that the application of the critical resource selection method in an underloaded facility will have no beneficial effects.

Due date performance of the selection methods is more clearly differentiated in models with loadings of 100 through 300%. For mean tardiness and mean delivery error random selection is clearly the best method by between 11 and 15% (ref: Table D.10 and D.12). For the average mean earliness and percentage of jobs tardy LM2 and QM1 are the better methods by approximately 45%.

The conclusion to be drawn from these results is that LM2 and QM1 results in fewer jobs with positive lateness suggesting that more jobs on average are early (or on time) than in the other methods. The inference then is LM2 and QM1 are the best methods for getting jobs through the facility the quickest but at the expense of a relatively smaller number of jobs which are late causing higher values of the mean tardiness measure.

Manufacturing Error

The Manufacturing Error of a job is the difference between the theoretical flowtime and the actual flowtime. The difference in the two values is as a consequence of the jobs spending time in queues waiting to be processed.

Table D.10 is the sum of the mean manufacturing error for each level of loading for each dispatching rule and each critical resource selection method. The results in this table indicate no outstanding selection method, however with respect to the dispatching rule SPT produced superior results in the under loaded facility up to 100%. For the 200 and 300% loading models random dispatching ranked highest.

Conclusions on Individual Criteria

SPT retained the property of minimising flowtime when used as part of the MSA sequencing heuristic.

The LM2 and QM1 methods were the most effective at moving the largest number of jobs through the facility in the shortest time, i.e., minimising percentage tardy and maximising the mean earliness.

7.5.4 QUEUEING DYNAMICS

To analyse the queuing dynamics of the models four criteria were collated: mean queue length, maximum average queue length, standard deviation and the coefficient of variation. These criteria are important in accessing which heuristic resulted in the lowest amount of WIP inventory and in stability of the model.

Each model contained twelve resources and results were collated for 25 periods hence 25x12 = 300 observations were taken per model per combination of selection method and dispatching rule. Each model was replicated three times resulting in criteria such as standard deviation of queue length being derived from 900 observations.

Queue Length

Table D.28 contains the results for the average value for the time average queue length over the whole model. Table D.30 contains results from calculating the average queue length of each resource and finding the maximum of these averages. The two tables present these results with respect to the selection method used to identify the critical resource and the dispatching rule.

In the models with loading 85% or less the results are again inconclusive with no outstanding dispatching rule or selection method.

The most significant result is that the EDD dispatching rule produced a lower average queue length and the lowest maximum average queue length for models with loading of between 100 and 300%, i.e. EDD resulted in the lowest

average WIP inventory levels. No conclusive results are possible for loading levels of 40 - 85% for which no dispatching rule was outstanding.

Considering the mean queue length, loading 100 - 300%, EDD was between 9 and 13% higher than the second ranked dispatching rule (ref: Table D.28). The difference was less marked for maximum average queue length, the range being between 0.1 and 15% (ref: Table D.30).

Model Stability

Model stability refers to the amount of variability in the queue lengths facing work centres. Model stability is measured by the standard deviation and the coefficient of variation of the observations of the queue length.

SPT dispatching resulted in the lowest value for the standard deviation figure for the loading levels less than 100%, however, at levels 100 to 300% the results become inconclusive.

Clearly randomly selecting the critical resource each period resulted in a more stable model, illustrated by being ranked first for six of the seven levels of loading when considering the coefficient of variation and five out of seven for the standard deviation. This indicates an interesting extension to this research, namely, to discover why the more structured methods of selecting the critical resource, which selected the same resource more often, produce models which are more unstable with respect to queue length variance. The whole question of fixed or semi-permanent critical resources against infinitely variable critical resources is one well worth further investigation with the MSA software and consequently has been studied in Section 7.5.6.

Finally the QM1 method produced the highest coefficient of variation for loading levels 100 to 300%.

7.5.5 ANALYSIS OF THE SELECTION MADE BY CRITICAL RESOURCE ID METHODS

Reported in this section is the stability of the critical resource selection methods, that is, how often the same resource was chosen to be the critical resource during a simulation run.

One conclusion drawn from the previous two sections was that identifying and using a critical resource in the under loaded models did not have a significant effect upon the results produced. Subsequent analysis of the selection models then concentrates on the 100, 200 and 300% loading models. Table D.40 indicates the percentage of times the same resource was chosen by a particular selection method over the whole of a simulation run. For example, consider the sequencing heuristic LM2-FCFS when applied to the model with 100% loading. The column headed "1st" has the value 43 indicating during the simulation run the same resource was chosen in 43% of the periods. The second most selected resource (column headed "2nd") was chosen in 40% of the periods simulated.

As in the analysis of the evaluation criteria the results of the coincidence of critical resource choice have been summed and averaged with respect to the dispatching rule and selection method, these results are presented in Table 7. 11 (Table D.39 Appendix D)

Model	100%			200%			300%		
	1st	2nd	3rd	1st	2nd	3rd	1st	2nd	3rd
LM1	37	28	17	67	17	9	64	20	12
LM2	55	32	10	85	11	4	90	9	2
QM1	93	4	3	90	10	0	58	42	0
random	67	15	6	72	15	9	65	26	8
FCFS	63	24	10	82	8	9	83	12	2
SPT	63	19	9	92	7	1	66	27	5
EDD	50	27	14	76	20	4	68	29	3

Table 7.11: Aggregate Values for Percentage Times The Same Resource is Chosen to be the Critical One

There are two significant results from Table 7.11. The first result found shows that there is a direct relationship between the leveling of loading and the coincidence of choice of LM2. The second result, reveals an inverse relationship between loading and the coincidence of choice of the QM1 method. Thus loading increases LM2 becomes more stable and QM1 becomes less stable.

The cumulative results for the effect of the dispatching rule produce inconclusive results. An interesting point to note from the non-aggregated data (Table D.40) under the QM1 model (100 and 200% loading) the random, FCFS and SPT rules under LM2 and QM1 choose the same resource in all simulated periods. The EDD rule under LM2 and QM1 however, choose the same resource in only 60 and 70% of the periods simulated. One possible conclusion drawn from this is that the EDD rule is affecting the location of the critical resource more so than the other rules when implemented under the QM1 selection method.

7.5.6 USING THE STABILITY OF CHOICES

The previous sections of this chapter have discussed the results of the 336 simulation runs outlines in Section 7.5 and Table 7.7. Taking the results for the stability of selection of the critical resource a further thirty of the thirty five secondary simulations were performed, replicated three times. These extra simulation runs take the most often selected resources under LM2 (3 of) and QM1 (2 of) and use the manual selection routine to make these the critical resource for all the periods simulated using the EDD and SPT rules. Only two critical resources were used from the QM1 models because in the worse case behaviour only two were selected for all periods simulated. This process was performed for the three levels of loading 100, 200 and 300%, the basic results are presented in Tables D.45 through D.50. To aid the analysis the data was summed with respect to the resource used and dispatching rule for each level of loading, the results of which are shown in Table 7.12 and 7.13 (Table D.43 and D.44 Appendix D).

The clear result is that if the future work plan is known (in this case 30 weeks in advance) then the LM2 method should be employed to decide which resource to identify as critical for the whole simulated period. The EDD dispatching rule should then be used to sequence the jobs. Over all three loading levels the EDD approach out performed all other methods with all three of the resources chosen under LM2.

Table 7.14 and 7.15 compares the best result, worst result and those from manually choosing the most often chosen resource under LM2 and using the EDD rule. For example, the "best" value for Flowtime (1) with 100% loading

comes from when the sixteen combinations of selection method and dispatching rule were simulated through the 100% loading model.

The percentage improvement figure is the manual method over the best recorded method in normal application of the sequencing heuristic. The significant result is the percentage improvement in the value for mean earliness, from 33% improvement (100% model) up to 157% improvement (300% model). These improvements are not the result of a relatively small number of jobs being early because of the 28% improvement in the 300% loading model, in the percentage number of jobs tardy.

Effectiveness of the sequencing methodology being measured by mean flow time, and mean manufacturing error, along with the method of using the same resource produced results comparable with the best results from normal application of the heuristic.

In summary if the work plan is known with certainty for a number of weeks in the future then it would be beneficial to use the LM2 method to locate the overall critical resource and use this for all the periods with the EDD rule to sequence the jobs.

Rank	CR id method	mean flow time(1) value	rank	mean flow time(2) value	rank	mean early value	rank	mean manuf'g error value	rank
SPT									
1	LM2	9007	4	9750	5	870	5	7501	5
2	LM2	9325	8	9780	6	1534	1	7647	6
3	LM2	9320	7	9869	7	1097	3	7700	8
1	QM1	9032	5	9663	4	679	8	7320	3
2	QM1	9341	9	10201	9	828	7	7693	7
EDD									
1	LM2	8448	2	9064	1	1384	2	6574	2
2	LM2	8227	1	9257	2	862	6	6517	1
3	LM2	8706	3	9352	3	1030	4	6574	2
1	QM1	9637	10	10356	10	536	9	7953	9
2	QM1	9054	6	10121	8	450	10	7384	4

Table 7.12: Aggregated Measures of Inventory and Effectiveness using SPT and EDD and One Critical Resource the Critical Resource For 100-300% Loading

Rank	CR id Method	mean tardy		% of all jobs tardy		mean delivery error		Average Rank	
		value	rank	value	rank	value	rank	value	rank
	SPT								
1	LM2	4236	8	52	5	5280	8	40	5
2	LM2	4366	9	45	2	5900	10	42	6
3	LM2	4391	10	48	3	5318	9	47	8
1	QM1	3548	4	58	7	4060	3	34	4
2	QM1	3804	6	57	6	4458	6	50	9
	EDD								
1	LM2	3273	2	43	1	3048	1	11	1
2	LM2	3136	1	49	4	3997	2	17	2
3	LM2	3400	3	48	3	4430	5	26	3
1	QM1	3934	7	61	8	4470	7	60	10
2	QM1	3633	5	66	9	4083	4	46	7

Table 7.13 : Aggregated Measures of Customer Performance using SPT and EDD and One Critical Resource 100-300% Loading

Rank CR id method	mean flow time(1)	mean flow time(2)	mean early	mean manuf'g error
100%				
best	5941	6487	1332	4197
worst	6437	7243	615	4736
manual	5809	6190	1774	4024
percent improvement	2	5	33	4
200%				
best	8477	9483	768	6726
worst	10585	11195	155	8895
manual	8659	9215	1374	6887
percent improvement	-2	3	79	-2
300%				
best	10953	12739	392	9237
worst	13008	14788	15	12457
manual	10477	11788	1006	8810
percent improvement	5	7	157	5

Table 7.14: Best, Worst and Manual Choice Results

of Inventory and Effectiveness

For 100-300% Loading

Rank CR id Method	mean tardy	% of all jobs tardy	mean delivery error
100%			
best	637	34	1541
worst	1264	51	2269
manual	864	38	2669
percent improvement	-35	-12	-73
200%			
best	2915	52	3119
worst	4683	83	5092
manual	3535	41	4909
percent improvement	-21	21	-57
300%			
best	5827	69	5864
worst	8468	96	8810
manual	5419	50	6425
percent improvement	7	28	-10

Table 7.15: Best, Worst and Manual Choice Results for Customer Satisfaction For 100-300% Loading

7.5.7 EFFECT OF MAXIMUM QUEUE SIZE

In the normal application of the sequencing heuristic it was shown that the LM2-SPT approach produced the most reliable results. For investigation of the effect of limiting the size of the queues which may reside in front of a work centre, the LM2-SPT method was further investigated. Throughout the first 336 simulation runs, to test all the combinations of the heuristic, the queue size was set at 20 lots. The analysis of restricting the queue size was performed using the LM2-SPT heuristic, with queue sizes restricted to 15, 10, 7, 5 and 4 lots for the 100% loading model. The simulation was run with smaller queue sizes but the effect resulted in the model becoming blocked and the software running out of working memory. The results from these simulation runs are presented in Tables 7.16 and 7.17.

The results show, as expected, that restricting queue sizes below a "critical" size, leads to significant degradation of the results.

From the above analysis an interesting future area of research is indicated. If the models used were balanced flow lines then restricting the queue size may not have had this detrimental effect on performance. This point is demonstrated by successful implementations of JIT manufacturing. The further research proposed is to develop algorithms which measure the approximate closeness to flow line conditions of a facility. This "evaluation" could then be used to guide the selection of the appropriate sequencing heuristic.

queue length	mean flow time(1)	mean flow time(2)	mean early	mean manuf'g error
fifteen	6294	6743	1360	4606
ten	6973	7406	1368	5251
seven	8359	8858	1102	6773
five	9049	9453	912	7585
four	9089	9493	549	7606

Table 7.16: Measures of Inventory and Effectiveness Using LM2-SPT in Model with 100% Loading and Restricted Queue Sizes

queue length	mean tardy	% of all jobs tardy	mean delivery error
fifteen	1136	40	2496
ten	1776	40	3144
seven	3113	48	4215
five	3651	57	4562
four	3283	70	5832

Table 7.17: Measures of Customer Performance Using LM2-SPT in Model with 100% Loading and Restricted Queue Sizes

7.6 Using The Simulator

This section discusses the operating overhead involved in using the MSA simulator. There are two reasons for introducing this data, the first being that any researcher wishing to implement the work will have a general idea of the operating overheads, memory and time requirements, secondly, to present an idea where improvements could be made to the efficiency of the simulation software.

7.6.1 TIME TO CREATE A MODEL

In creating any model a certain overhead time is required regardless of the size of the model. Overhead operations include: input the three digit character code, time for the software to create an empty file set and time for the user to input the calendar. It the UT's are created in the "global define" mode then it simply takes as long as needed to type in the names of the resource. The control parameters are set to the default values by the software. Start up takes 7.5 minutes to create a 10 product, 5 resource model; perform a "logic check" and launch a default simulation. A 20 product version takes 13 minutes, a one product version, 2 minutes. The most time consuming item is inputting the BoM database and the Jobs Pending file. To create a project using data files from those previously defined took less than a minute.

When the test cases where being created, several programs where written to generate the data files. When applying the software, once the BoM's have been input they can be copied within the MSA software into other projects.

Not directly addressed by this project was the importing of data files from an outside source such as an MRP system. However, if the format could be determined and the files stored as an ASCII file on an MS-DOS formatted disk then utilities could be written to access them with reasonable effort.

7.6.2 STORAGE REQUIREMENTS OF THE MODELS

Table 7.18 illustrates the storage requirements of the nine problems used to investigate the stability of the methodology and whether steady state was possible. The values are for the models at week 0:

Res's	Number of Products 5	15	25
5	14k	18k	22k
12	18k	24k	30k
20	20k	31k	41k

Table 7.18: Storage Requirements of Initial Models

Clearly an initial model takes a relatively small amount of memory.

The amount of memory required for the standard simple job shop test case with 12 resources and 16 individual BoM's is 25k.

The model was simulated up to steady state at period 100. To store the complete model with all the data files for all 100 periods required approximately 6.5 megabytes (6,758k). After the model had been simulated for 100 periods the contents of the working files where copied over the "root" file set so as to define a new initial point for the model. The memory requirement is then only for one period, namely week 100. The storage

requirement for this model at steady state was 103k. The main reason for the significant difference is the presence of 504 entries in the run time Sequencing Database.

If the user selects to write all the simulation actions to file, the storage requirement can be excessively large. If the user chooses to write all actions to memory, the standard simple job shop run for thirty periods would generate a run file of 3.6 Megabytes. Obviously the size of the run file is dependent on the number of actions occurring which is dependent on the number of resources and jobs in the facility, whether set-up is used or batches split and sent ahead.

7.6.3 TIMES TO RUN A SIMULATION

The times discussed here are from simulations run on IBM PS/2's computers with 80386 processors and installed 80387 math co-processors. The software was run on a virtual drive created in 80 Ns RAM. At the end of the simulation run all the data was copied to the hard disk and erased from the virtual drive to make room for the next simulation run.

The time to simulate a period is dependent on many factors, namely, the number of products, resources, jobs in the system and the sequencing heuristic used. Critical resource selection methods LM2 and QM1 are dynamic models which each first simulate through a period to identify the critical resource. Once the critical resource is identified then the period is simulated again using the critical resource as the focus. Table 7.19 presents simulation times from steady state at week 100 to week 130 using different sequencing heuristics under different levels of congestion.

The conclusion drawn from the table is that the application of the dispatching rule inside a sequencing heuristic does not add significant amounts of time. The random and LM1 critical resource selection methods take approximately the same time to execute given a certain level of loading. The LM2 and QM1 model also take a similar time to run which is 33% more than the random and LM1 models. The other significant point to note is that doubling and tripling the transient loading level caused the simulation run time to increase by 33% each increment.

A last note on the efficient use of the software. When the data is stored generally, it is appended to the file set for the particular simulation project-version combination. Purging the files of simulated weeks which will no longer be used speeds up the archiving process and reduces the memory overhead of the software.

Model No	CR Id and DR	100% transient loading	200% transient loading	300% transient loading
	Random			
1	random	1.40	2.27	3.38
2	FCFS	1.38	2.20	3.20
3	SPT	1.33	2.13	3.23
4	EDD	1.38	2.13	3.28
	LM1			
5	random	1.50	2.33	3.63
6	FCFS	1.46	2.32	3.46
7	SPT	1.50	2.13	3.38
8	EDD	1.50	2.28	3.48
	LM2			
9	random	2.37	3.50	5.12
10	FCFS	2.22	3.28	4.90
11	SPT	2.32	3.52	4.98
12	EDD	2.37	3.52	5.61
	QM1			
13	random	2.23	3.67	4.92
14	FCFS	2.27	3.38	4.98
15	SPT	2.20	3.30	4.85
16	EDD	2.38	3.52	5.02

Table 7.19 : Hours Required to Simulate 30 periods Beginning At Steady State with 100, 200 and 300% Loading

7.7 Analytical Summary

The objective of this chapter has been to test the developed methodologies. Testing the methodologies indicated that in an overloaded work programme, the LM2-SPT approach is the most effective. If the work plan is fixed for a number of periods the LM2 method is most effective for selection of an overall critical resource which is used then with EDD dispatching for job sequencing.

Critical Resource Selection

1 If the work plan is known only on a weekly basis, the LM2 selection method is the most effective.

2. If the future work plan is known (in this case 30 weeks in advance) then the LM2 method should be employed to decide which resource to identify as critical for the whole simulated period. The EDD dispatching rule should then be used to sequence the jobs.

3. There appears to be no advantage to using a particular critical resource selection method in underloaded models (40-85% loading).

4. LM2 and QM1 resulted in the highest average mean earliness and lowest percentage tardy.

5. When deciding which resource is critical, loading appears to be more important than queueing dynamics, as evidenced by the greater success of the LM2 method over the QM1 method.

6. The LM1 static approach to locating the critical resource produces results which in the majority of cases are worse than when the critical resource is randomly selected.

Dispatching Rule

7. In the single resource model the SPT rule will always minimise flowtime. The SPT rule retains this property when used as part of the MSA sequencing heuristic in models with up to 100% loading.
8. The sequencing heuristics involving SPT produce the maximum mean earliness and lowest percent of jobs tardy.
9. The EDD method proves to be the best at minimising flowtime in an overloaded shop. This point is in concurrence with Goodwin and Weeks (1986).
10. EDD, when used as part of a critical resource sequencing heuristic, retained the property of minimising mean tardiness.

CHAPTER EIGHT

FUTURE WORK

8.1 Introduction

This chapter details four directions for possible extension of the research reported in this thesis: modeling and simulation, locating the critical resource, methods used to sequence work and enhancements to the developed software.

The end user should be a priority when developing extensions to the MSA methodology. The simulation package may be sophisticated in its implementation and operation, but final application of the sequencing heuristic on the shop floor must be kept in mind. An objective of this research has been to develop sequencing methods that are both effective and implementable. Some methods, such as Load Level, are not truly implementable in the current job shop environments because information is often not integrated into a common database. The development of sophisticated simulations should be aimed to produce sequencing systems that are easy to implement and maintain, and result in a simple-to-use job prioritising system.

8.2 Modeling and Simulation

The following key points indicate how the simulator can be made more efficient and functional. However it is essential that the goal of generic applicability and ease of use should not be compromised.

8.2.1 RELAXATION OF ASSUMPTIONS

Similar to the majority of research on sequencing and scheduling of job shops, several simplifying assumptions have to be made to produce more tractable models.

Learning Curves: In the present version of the software, work centres begin processing a new job at the standard output rate, i.e., 100% efficiency. A learning curve could be included in the software using the efficiency field in the UT Database to model the build-up to the standard output rate.

Variable Processing Times: In the current MSA modeler the processing times of jobs are fixed. This scenario is only realistic whenever the work centre operates without human intervention. Variability can be added to processing times with the use of probabilities. Two methods are proposed:

1. Variability at the work centre for all jobs, or
2. Specific part/work centre variability.

The first method could be achieved by adding an extra field to records in the UT database; this field would then be used to contain the variance for the processing time.

To implement the second method a field could be added to the Sequencing Database, containing information about a specific job on a specific work centre. The field would contain the variance for the processing time. For CNC work centres this would be zero in most cases, since the standard output rate could be met. For a specific part, manually machined, the variable in this field would reflect the complexity of the part.

Downtime: The effectiveness of the simulator could be enhanced by the development of an event list mechanism that would reduce the amount of time required to run a simulation. Event times for work centre breakdowns could then be probabilistic or deterministic in nature and added to the event list. The time at which the work centre is brought back on line could also be an event added to the list.

Two methods to simulate this are proposed. The first would add machine breakdown events to the event list at the beginning of the period to be simulated. A second method would develop a routine enabling the user to interrupt the simulation and change the status of a UT from operational to non-operational.

If downtime events are to be added to the simulator, extra functionality would be necessary. If no extra functionality was implemented (as in the present version), queues would amass in front of the non-operational work centre until it became operational. This problem might be alleviated through secondary routes or by stoppage of all processing of jobs on route to the non-operational work centre.

8.2.2 SIMULATING BACKWARDS

The launch date of a job directly affects its progress through the facility. For this reason the performance of a job has usually been gauged against the Theoretical Flowtime. When not using a simulation, calculating the true Latest Start date of a job is a difficult process, given that batches may be split and the job may spend an indeterminate amount of time queuing. An interesting area of research would be to run the simulation driver backwards. Jobs would be released to the facility at their due date and simulated back through

the facility until they "graduate" from the facility into the Job Pending File generating a launch date.

If the generated launch time had already been passed, the simulation could be run again and the priority of the job raised.

8.3 Locating The Critical Resource

8.3.1 MULTIPLE CRITICAL RESOURCES

To date, research has tended to concentrate on the existence of only one Critical Resource or Bottleneck. An alternative perspective would be to group resources into critical resource sets. Investigations may then focus on the interactions between of these groups and their effect on the scheduling function.

8.3.2 SECONDARY ROUTES

When using the MSA modeler it is only possible to input one route for a job. A possible approach to lowering the load on a potential Critical Resource would be to have secondary routes available. Changing job routes might allow different work centres to become the critical resource. The question would then become where within the facility is the optimal location for the critical resource .

8.3.3 METHODS TO IDENTIFY THE CRITICAL RESOURCE

In this initial version of the MSA methodology two critical resource identification methods (CR-QM1 and CR-LM2) proved to be the most successful. Further research might add new methods to identify the critical

resource.

Queuing Models

The present queuing based model uses the time average number of jobs in the queue facing a resource. Extensions to this approach would use time average processing time of queues or a weighted average of the both measures.

Loading and Queuing Models

The LM2 critical resource identification method first simulates through the facility using one of the dispatching rules and no defined critical resource. The resource chosen as critical is that which has the highest loading for the period simulated. An extension of this model might develop a weighted average of the loading and the queuing statistics.

Proactive Identification

The most successful methods of identifying the critical resource (CR-QM1 and CR-LM2) required the period of interest to be simulated twice. Method CR-LM1 was designed to proactively locate the critical resource without simulating through the facility. Future research would attempt to improve the CR-LM1 method by making it more sensitive to current shop loading and potential shop loading in the period about to be simulated.

8.4 Methods Used to Sequence Work

8.4.1 DISPATCHING RULES

Future research might consider an increased number of dispatching rules and an investigation into their effect. At present the dispatching rule is manually

chosen. This process, however, could be improved by linking the selection of the rule to the state of the facility and the jobs about to be launched and the importance of particular performance criteria, e.g., due date or inventory levels.

8.4.2 BATCH SIZING

When sizing the transfer batch no reference is made to the loading on the shop. Future methods of determining the transfer batch size could be linked to both the state of the facility and to the jobs about to be launched. Additionally, an interesting enhancement of the research would be variable batch sizes for each of the components, sub-assemblies, and job end item.

8.4.3 JOB RELEASE

Jobs are released to the shop floor for processing at the beginning of each week. Loading the gateway resources with a relatively large number of jobs causes a "wave" effect at the start of the week . A time-phased mechanism for the release of jobs could be developed to avoid this transient overload. To make this implementable on the shop floor, a release schedule would have to be generated by the software and made available to the foreman responsible for the release of jobs. In this way the simulations take the state of the shop into account when generating the schedule. The duty of the foreman would then be to follow the timetable of job release. Thus specific details of the state of the shop would not be required by the foreman when deciding which job to launch.

8.4.4 DUE DATE OFFSET

In the present methodology the launch date of a job is the beginning of the

week in which the latest start date (LSD) falls. The LSD is calculated by stepping back from the due date by an amount equal to the total work content on the critical path multiplied by the due date offset. The Due Date offset is arbitrarily set. In future versions the due date offset could be linked dynamically with the projected loading on the facility and the estimated queuing times: the greater the loading, the higher the Due Date Offset.

8.5 Enhancements to the Developed Software

8.5.1 PROGRAMMING LANGUAGE MEMORY LIMITS

The run-time Sequencing Database has to be loaded into an array for the operation of the software. At present, the array containing database information can store a maximum of 2000 records. If each job has twelve processing stages, as in the main test case models, a maximum of 150 (2000/12) jobs can be "live" at any one time. An excess will cause an out-of-bounds condition on the array and a consequent software crash. If the size of the array is increased, the software crashes with an out-of-memory error. To apply the MSA software suite in an industrial environment, this figure would have to be dramatically increased. The increase might be achieved through the use of extended memory or through an alternative database design.

Use Of Extended Memory

When a simulation is running all the associated data (Sequencing Database, Resource Database, etc) are loaded into RAM. The size of RAM is restricted by DOS to a 640k maximum; the actual working size on a standard PC is approximately 580k. By using DOS extenders (e.g. Phar Lap) and by programming the MSA software in C, it may be possible to use extended memory and thus have the ability to handle larger data files.

Database Design

Some data is replicated in the Sequencing Database. To reduce the storage requirements, the database could be split into two versions. A data version would contain information on a job; and a dynamic version would hold information about each component and sub-assembly of a job.

8.5.2 USE OF AN EVENT LIST

In the present implementation of the MSA methodology, once an event time has been fully processed, the whole model is scanned to determine the next event time. In extreme cases, over 5 hours were required to simulate some models through thirty periods. An event list mechanism could be implemented to speed up each simulation run.

8.6 The Future Potential of Critical Resource Modeling

By dynamically locating the critical resource with respect to the current product mix, attention is focused on capacity constraining resource. The sequencing methods ensure that the critical resource schedule is protected by giving critical resource bound jobs the highest priority.

The implementation of the methodology on the shop floor will be a simple to use job prioritizing system. The proposed extensions to the methodology and implementation mentioned in this chapter will make the model more realistic and the simulation more sophisticated while still maintaining generic applicability and simplicity of use.

CHAPTER 9

CONCLUSIONS

9.1 Introduction

This thesis has detailed the development, implementation and testing of the MSA Generic Simulation-based Critical Resource Scheduling Methodology. This chapter presents the conclusions drawn from the research results (reported in Chapter 7) using the theoretical test cases generated to test this methodology

9.2 The Concept of Critical Resource Modeling

1. Positively identifying the critical resource focuses attention on the capacity-constraining resource. Increasing the throughput of this resource may potentially raise the throughput of the facility.

2 Limiting the priority levels on the shop floor to two (critical resource-bound and non critical resource-bound jobs) results in a sequencing mechanism easier to implement, understand, and maintain.

3 Dynamic critical resource identification allows the effects of a changing product mix to dictate the location of the critical resource and thus the priority of jobs, period by period.

9.3 Methods Used to Identify The Critical Resource

1. If the work plan is known only on a weekly basis, the LM2 selection method is the most effective.

2. If the future work plan is known (in this case 30 weeks in advance) then the LM2 method should be employed to determine which resource to identify as critical for the whole simulated period. The EDD dispatching rule should then be used to sequence the jobs.

3. There appears to be no advantage to using a particular critical resource selection method in underloaded models (40-85% loading).

4. LM2 and QM1 yield in the highest average earliness and lowest percentage tardy.

5. When determining which resource is critical, loading appears to be more important than queueing dynamics, as evidenced by the greater success of the LM2 method over the QM1 method.

9.4 Methods Used to Sequence Work

1. In the single resource model, the SPT rule will always minimise flowtime. The SPT rule retains this property when used as part of the MSA sequencing heuristic in models with up to 100% loading.

2. The sequencing heuristics involving SPT produce the maximum mean earliness and lowest percent of jobs tardy.

3. The EDD method proves to be the best at minimising flowtime in an overloaded shop. This point is in concurrence with the conclusion of Goodwin and Weeks (1986).

4. When used as part of a critical resource sequencing heuristic, EDD retained the property of minimising mean tardiness.

9.5 Effectiveness of the Generic Simulator

1. The modeling approach uses only the Universal Transfer to represent all resources. The goal of this approach was to simplify the modeling process and allow generic applicability.

2. The data input requirements were reduced by using a restricted data set simplifying and speeding up the model development process.

3. The functionality of launching a series of simulations from the same data file set but with different control parameter settings, allows for extensive "what if?" analysis to be performed in a relatively simple manner.

9.6 Effectiveness of Software Developed

1. User friendly and functional menu and user information screens have been developed and implemented.

2. The extensive project manipulation software proved a powerful concept and tool when performing the test case simulation runs.

3. When editing of job routes is required, employing only one modeling entity and having no explicit network model to maintain and update proves to be an effective approach.

4. The implementation of the simulation allows several runs to be generated from the same root file set; this allows direct comparison of results.

5. The ability to stop and restart simulation runs is useful when determining steady state.

9.7 The Future Potential of Critical Resource Modeling

1. Kanban scheduling, although an effective To achieve the full benefits from Kanban scheduling JIT, although an effective method of operating a facility, requires a long implementation lead time. Use of critical resource scheduling can increase throughput and focus the flow of parts without capital investment or reorganisation of the manufacturing facility.

2. Automation and updating capital equipment are expensive processes. Through the use critical resource scheduling attention is directed to capacity-constraining resources, thus indicating where capital investment will be most effective. The MSA simulator may be used in a "what if?" mode to investigate the effects of capital investment in capacity-constraining resources.

3. In an era of "zero-inventory", it is increasingly important to identify the critical resource and maintain a buffer of work before it. This will ensure the critical resource schedule is protected if there is a temporary problem upstream in the manufacturing facility.

9.8 PUBLICATIONS FROM THIS RESEARCH

Driscoll, J. and Hurley, S.F., (1992), "The Application of a Generic Critical Resource Scheduler to a Manufacturing System," in refereed proceedings **Flexible Automation and Information Management** ed. Eyada, O.K. and Ahmed, M.M., CRC Press, USA. pp. 677-688.

Driscoll, J. and Hurley, S.F., (1989), "The Development of a Generic Bottleneck Scheduler," included in post conference book **Production Research Approaching The 21st Century**, ed Pridham, M. and O'Brian, C. pp 199-207.

9.9 CONTRIBUTION OF THIS RESEARCH

The research described in this thesis contributed the following:

1. Development of a new methodology for modeling a manufacturing facility.

2. The use of a restricted set of data in order to gain the maximum amount of benefit from the least effort.

3. Development of a new and effective critical resource sequencing methodology. This included critical resource identification techniques and methods to use the critical resource as the focus for sequencing jobs through the facility.

REFERENCES

Axsater, S., (1984), "Lower Bounds For Economic Lot Scheduling Problem Using Aggregation," **European Journal of OR**, Vol. 17, pp. 201.

Baker, K.R., (1974), **Introduction To Sequencing and Scheduling**, Wiley, New York.

Banks, J. and Carson II, J.S., (1984), **Discrete-Event System Simulation**, Prentice-Hall, Englewood Cliffs.

Bechte, W. (1988), "Theory and Practice Of Load-Orientated Manufacturing Control," **IJPR**, Vol. 26(3), pp. 375-395.

Bechte, W., (1982), "Controlling Manufacturing Lead Times and Work In Progress Inventory by Means of Load-Orientated Order Release," in **APICS 1982 Annual Conference Proceedings**, pp. 67-72.

Bellofatto, W.R., (1974), "Lead Time vs The Production Control System," **Production and Inventory Management**, Vol. 15(2), pp. 14-22.

Blackstone Jr., J.H., Phillips, D.T. et al., (1982), "A State-of-the-art Survey Of Dispatching Rules for Manufacturing Job Shop Operations," **IJPR**, Vol. 20(1), pp. 27-45.

Boctor, F.F., (1982), "The Two-Product, Static Demand, Infinite Horizon Lot Scheduling Problem," **Management Science**, Vol. 23, pp. 798.

Browne, J., (1988), "Production Activity Control - A Key Aspect Of Production Control," **IJPR**, Vol. 26(3), pp. 415-427.

Burbidge, J.L., (1987), **IFIP Glossary Of Terms Used In Production Control**, North Holland, Amsterdam.

Buxey, G., (1989), "Production Scheduling: Practice and Theory," **European J. of OR**, Vol. 39, pp. 17-31.

Carrie, A., (1988), **Simulation of Manufacturing Systems**, Wiley, Chichester.

Cleland, D.I. and Bidanda, B., (1990), **The Automated Factory Handbook: Technology and Management**, Tab Press, Blue Ridge Summit.

Coffman, E.G. et al., (1976), **Computer and Job Shop Scheduling Theory**, John Wiley and Sons, New York.

Conway, R.W. et al, (1967), **Theory of Scheduling**, Addison-Wesley, Reading.

Conway, R.W., (1965), "Priority Dispatching and Work-In-Progress Inventory in a Job Shop," **J of Industrial Engineering**, Vol. 16(2), pp. 123-130.

Conway, R.W. and Maxwell, W.L., (1962), "Network Dispatching by Shortest Operation Discipline," **Operation Research**, Vol. 10, pp. 51.

Dar-El, E.M. and Wysk, R.A., (1982), "Job Shop Scheduling - A Systematic Approach," **J Of Manufacturing Systems**, Vol. 1?(1), pp. 77-88.

Dzielinski, B.P. and Baker, C.T., (1960), "Simulation of a Simplified Job Shop," **Management Science**, Vol. 6(3), pp. 311.

Eilon, S. and Cotterill, D.J., (1975), "A Modified SI Rule in Job Shop Scheduling," **IJPR**, Vol. 7, pp. 135.

Elmaghraby S E, (1978), "The Economic Lot Sizing Problem (ELSP): Review and Extensions," **Management Science**, Vol. 24, pp. 587.

French, S., (1982), **Sequencing and Scheduling**, Wiley, New York.

Fry, T.D., Philipoom, P.R. et al., (1988), "Dispatching In a Multistage Job Shop Where Machine Capacities Are Unbalanced," **IJPR**, Vol. 26(7), pp. 1193-1223.

Garey, M.R., Johnson, D.S. and Sethi, R., (1976), "Complexity of Flow Shop and Job Shop Scheduling," **Mathematics of O.R.**, Vol. 1(2), pp. 117-129.

Gere Jr, W., (1966), "Heuristics In Job Shop Scheduling," **Management Science**, Vol. 13, pp. 167-190.

Giffler, B. and Thompson, G.L., (1960), **Operations Research**, Vol. 8(4), pp. 487-503.

Goldratt, E.M. and Fox, R.E., (1984), **The Goal : Excellence In Manufacturing (2nd ed.)**, North River Press.

Goldratt, E.M. and Fox, R.E., (1986), **The Race**, North River Press.

Gonzalez, T. and Sahni, S., (1978), "Flowshop and Jobshop Schedules: Complexity and Approximation," **Operations Research**, Vol. 26(1), pp. 36-52.

Goodwin, J.S. and Goodwin, J.C., (1982), "Operating Policies For Scheduling Assembled Products," **Decision Sciences**, Vol. 13, pp. 585-603.

Goodwin, J.S. and Weeks, J.K., (1986), "Evaluating Scheduling Policies In A Multi-Level Assembly System," **IJPR**, Vol. 24, pp. 247-257.

Harl, J.E. and Ritzman, L.P., (1985), "A Heuristic Algorithm For Capacity Sensitive Requirements Planning," **J. of Operations Management**, Vol. 5(3), pp. 309-326.

Harrington, J., (1984), **Understanding the Manufacturing Process**, Marcel Dekker Inc., New York.

Hax, A.C. and Candra, D., (1984), **Production And Inventory Management**, Prentice-Hall, Englewood Cliffs.

Holstein, W.K, (1968), "Production Planning and Control Integrated", **Harvard Business Review**, May-June, pp.121-140.

Hendry, L.C. and Kingsman, B.G. (1989), "Production Planning Systems and Their Applicability to Make-To-Order Companies," **European J of O.R.**, Vol. 40, pp. 1-15.

Hsu, W.L., (1983), "On the General Feasibility Test of Scheduling Lot Sizes for Several Products on One Machine," **Management Science**, Vol. 29, pp. 93.

Huang, P.Y., (1984), "A Comparative Study of Priority Dispatching Rules in a Hybrid Assembly Shop," **IJPR**, Vol. 22(1), pp. 375-387.

Hurley, S.F., (1987), "JIT at Crown Paints Limited," **unpublished Masters' Dissertation**, School of Management, Lancaster University.

Jackson, J.R., (1956), "An Extension of Johnson's Rule on Job-Lot Scheduling," **Navel Research Logistics Quarterly**, Vol. 3(3), pp. 201-224.

Jackson, J.R., (1957), "Simulation Research on Job Shop Production," **Naval Research Logistics Quarterly**, Vol. 4, pp. 287-295.

Jackson, J.R., (1963), "Job Shop-Like Queuing Systems," **Management Science**, Vol. 10(1), pp. 131-142.

Jacobs, F.R., (1983), "The OPT Scheduling System : A Review Of A New Production Scheduling System," **Production and Inventory Management**, Vol. 24(3), pp. 47-51.

Jacobs, F.R., (1984), "OPT Uncovered : Many Production Planning and Scheduling Concepts Can Be Applied With or Without The Software," **Industrial Engineering**, Vol. 16(10), pp. 32-41.

Kanet, J.J., (1988), "Load-Limited Order Release In Job Shop Scheduling Systems," **J of Operations Management**, Vol. 7(3), pp. 44-58.

Karmarkar, U.S., Kekre. S., et al., (1989), "Capacity Analysis of a Manufacturing Cell," **J Of Manufacturing Systems**, Vol. 6(3), pp. 165-175.

Law, A.M. and Kelton, D., (1981), **Simulation Modelling and Analysis**, Mcgraw-Hill, New York.

Law, A.M. and Kelton, D., (1990), **Simulation Modelling and Analysis (2nd ed)**, Mcgraw-Hill, New York.

Lenstra, J.K. et al., (1977), "Complexity of Machine Scheduling Problems," **Annals of Discrete Mathematics**, Vol. 1, pp. 343-362.

Lenstra, J.K. and Rinnooy-Kan, A.H.G., (1978), "Complexity of Scheduling Under Precedence Constraints," **Operations Research**, Vol. 26(1), pp. 22-35.

Lundrigan, R., (1986), "What Is This Thing Called OPT?," **Production and Inventory Management**, Vol. 27(2), pp. 2-11.

Manne, A.S., (1960), "On The Job-Shop Scheduling Problem," **Operations Research**, Vol. 8(2), pp. 219-223.

Morton, T.E. and Dharan, B.G., (1978), "Algoristics Single-Machine Sequencing with Precedence Constraints," **Management Science**, Vol. 24(10), pp. 1011-1020.

Morton, T.E., Lawrence, S.R. et al., (1988), "SCHED STAR: A Priced-Based Shop Scheduling Module," **Unpublished Research Paper**, CMU.

Panwalker, S.S. and Iskander, W., (1977), "A Survay of Scheduling Rules," **Operations Research**, Vol. 25(1), pp. 45-61

Pritsker, A.A.B., (1979), "Compilation of Definitions of Simulation," **Simulation**, Vol. 33(8), pp. 61-63.

Rochette, R. et al., (1976), "A Statistical Comparison of the Performance of Simple Dispatching Rules For a Particular Set of Job Shops," **IJPR**, Vol. 14(1), pp. 63-75.

Root, J.G., (1965), "Scheduling With Dead Lines and Loss Functions on k Parallel Machines," **Management Science**, Vol. 11(3), pp. 460-475.

Rothkopf, M., "Scheduling Independent Tasks on Parallel Processes," **Management Science**, Vol. 12(5), pp.437-477.

Russel, R.S. and Taylor, B.W., (1985), "An Evaluation of Sequencing Rules For An Assembly Shop," **Decision Sciences**, Vol. 16(?), pp. 196-212.

Salvador, M.S., (1978), "Scheduling and Sequencing," in **Handbook of Operations Research: Models and Applications**, Moder, J.J. and Elmaghraby, S.E. (eds.), Van Nostrand Reinhold, New York.

Schonberger, R.J., (1983), "Selecting The Right Manufacturing System : Western And Japanese Approaches," **Production and Inventory Management**, Vol. 24(2), pp. 33-43.

Swann, D., (1986), "Using MRP For Optimised Schedules (Emulating OPT)," **Production and Inventory Management**, Vol. 27(2), pp. 30-37.

Taha, H.A., (1988), **Simulation Modelling and SIMNET**, Prentice-Hall, Englewood Cliffs.

Tersine, R.J., (1985), **Production/Operations Management**, North-Holland, Amsterdam.

Wild, R., (1977), **Concepts in Operations Management**, Wiley, Chichester.

Wilson, G.T., (1985), "Kanban Scheduling - Boon Or Bane," **Production and Inventory Management**, Vol. 26(3), pp. 134-141.

Zale, R.S., (1990), **PowerBASIC Reference Guide**, Spectra Publishing, Los Angeles.

BIBLIOGRAPHY

Adam Jr., E.E. and Elbert, R. J. (1982), **Production and Operations Management (Concepts, Models and Behaviour) 2nd ed.**, Prentice-Hall, Englewood Cliffs

Adams, J. et al, (1988), "The Shifting Bottleneck Procedure For Job Shop Scheduling," **Management Science**, Vol 34(3), pp. 391-401.

Almodovar, A.R., (1988), "Manufacturing Simulators Save Time and Provide Good Data For Layout Evaluation," **Industrial Engineering**, Vol 20(6), pp. 28-33

Altiok, T. and Kao, Z., (1989), "Bounds for Throughput in Production/Inventory Systems in Series with Deterministic Processing Times," **IEE Transactions (IE R&D)**, Vol. 21(1), pp. 82-85.

Alty, J.L. and Coombs, M.J., (1984), **Expert Systems : Concepts and Examples**, NCC.

Anthony, R.N., (1965), **Planning and Control Systems: A Framework For Analysis**, Harvard University Press.

APICS, (1987), **APICS Examination Study Guides Certificate Program**, APICS.

Ashcroft, S.H., (1989), "Applying The Principles Of Optimised Production Technology In A Small Manufacturing Company," **Engineering Costs and Production Economics**, Vol. 17(1-4), pp. 79-88.

Ashley, S., (1983), "Artificial Intelligence In The Plant," **American Metal Market**, Vol. 56(1), pp. 7-12.

Ashton, F.A., (1985), "Master Production Scheduling For Original Equipment Automotive Suppliers," **Production and Inventory Management**, Vol. 26(4), pp. 70-79.

Bahl, H.C., et al, (1987), "Determining Lot Sizes and Resource Requirements," **Operations Research**, Vol. 33(3), pp. 329-345

Baker, J.C., (1987), **Manufacturing Systems Strategy Project : IDEF0 User Manual**, Advanced Tech Centre, US Air Force.

Baker, K.R. and Bertrand, J.W.M., (1981), "A Comparison Of Due Dates Selection Rules," **AIIE Transactions**, Vol. 6, pp. 123-131.

Barnetson, P., (1970), **Critical Path Planning**, Newnes-Butterworths, ?.

Bell, D. and Pugh, J., (1987), **Software Engineering**, Prentice-Hall, Englewood Cliffs.

Bensoussan, A., et al, (1983), **Mathematical Theory of Production Planning**, North Holland, Amsterdam.

Ben-Arieh, D. and Moodie, C.L., (1988), "Knowledge Based Routing and Sequencing for Discrete Part Production," **J. of Manufacturing Systems**, Vol. 6(4), pp. 287-297.

Bestwick, P.F. and Lockyer, K., (1982), **Quantitative Production Management : Solutions Manual**, Pitman, ?.

Bignell, V. et al. eds., (1986), **Manufacturing Systems: Context, Applications and Techniques**, Blackwell, Oxford.

Bitran, G.R. and Chang L., (1987), "A Mathematical Programming Approach To Deterministic Kanban System," **Management Science**, Vol. 33(4), pp. 427-441.

Blackstone, J.H. and Cox, J.F., (1985), "MRP Design Implementation Issues For Small Manufacturers", **Production and Inventory Management**, Vol. 26(3), pp. 65-75.

Blick, R.G., (1969), **Heuristics for Scheduling the General n/m Job Shop Problem**, GE R&D Centre Report 69-C-162, Schenectady, New York.

Boerstra, M.L. (ed.), (1985), **Engineering Databases**, Elsevier.

Borgoine, J., (1988), "Beyond MRP II", **Production Engineer**, January, pp. 31-33.

Bratley, P. and Fox, B.L. et al (1987), **A Guide To Simulation (2nd ed)**, Spinger-Verlag.

Buffa, E.S., (1983), **Modern Production/ Operations Management (7 ed)**, Wiley, New York.

Burbidge, J.L., (1988), "Operation Scheduling With GT and PBC," **IJPR**, Vol. 26(3), pp. 429-442.

Burnham, J.M., (1987), "Some Conclusions About JIT Manufacturing," **Production and Inventory Management**, pp. 7-10.

Bylinsky, G., (1983), "An Israeli Shakes Up US Factories," September, **Fortune**, pp. 120-131.

CAD, (1987), Boeing Launch PMS in Europe, **CAD**, Vol. 19(3), p. 66

Carlson, R.C., (1979), "Use of Dynamic Lot Sizing To Avoid Nervousness In mrp Systems," **Production and Inventory Management**, Vol. 20(3), pp. 49-58.

Carrol, D.C., (1965), "Heuristic Sequencing of Single and Multiple Component Jobs," **Unpublished Ph.D. Dissertation**, MIT, Cambridge.

Casey, C., (1987), "Accounting For Progress," **Production Engineer**, February, pp. 18-19.

Chanland, R., (1984), "Application Of The OPT System For Detail Resource Planning," in **Readings In Material and Capacity Requirements Planning**, APICS, pp. 79-81.

Conner, S., (1987), "... and select the right automation software.," **Process Engineering**, Vol. 68(3), pp. 33-35.

Conterno, R. and Ho, Y.C., (1988), "Ordering Scheduling Problem In Manufacturing Systems," **IJPR**, Vol. 26(9), pp.1487-1510.

Control Systems, (1988), OPT Shows The Way Forward, **Control Systems**, vol. 1, pp. 34-35.

Corke, D.K., (1988), **A Guide To CAPM**, Institute of Production Engineering.

Crosby, L.B., (1984), "The JIT Manufacturing Process : Control Of Quality And Quantity," **Production and Inventory Management**, Vol. 25(4), pp. 20-33.

Dar-El, E.M. and Rabinovitch, M., (1988), " Optimal Planning and Scheduling of Production Lines," **IJPR**, Vol. 26(9), pp. 1433-1450.

David, H.I. and LeFevre, D.C., (1981), "Finite Capacity Scheduling With OPT," in **APICS Conference Proceedings 1981**, pp. 178-181.

Dellaert, N.P., (1989), "Multi-Item Production Control For Production To Order," Engineering Costs and Production Economics, Vol. 17(1-4), pp. 167-173.

Delporte, C.N., (1977), "Lot Sizing and Sequencing For n Products On One Facility," **Management Science**, Vol. 23, pp. 1070.

DeMeyer, A., (1987), "The Integration of Information Systems In Manufacturing," **Omega**, Vol. 15(3), pp. 229-238.

Dobson, G. and Karmakar, U.S. et al, (1987), "Batching To Minimise Flow Times On One Machine," **Management Science**, Vol. 33(6), pp. 784-799.

Dogramaci, A. and Surkis, J., (1979), "Evaluation of a Heuristic for Scheduling Independent Jobs on Parallel Identical Processors," **Management Science**, Vol. 25(12), pp. 1208-1216.

Dumond, J. and Mabert, V.A., (1988), "Evaluating Project Scheduling And Due Date Assignment Procedures: An Experimental Analysis," **Management Science**, Vol. 34(1), pp. 101-118.

Durhman, N.R. and Kochlar, A.K., (1981), "Interactive Computer Simulation For Evaluation Of Manufacturing Planning And Control Strategies," in **Proceedings of Uksc**, UKSC, pp. 82-89.

Elghobary, H. and Aziz, T., "Computerised Scheduling Technique For Productivity Improvement," **Computers in Industrial Engineering**, Vol. 13(1-4), pp. 165-169.

Ellis Jr., W. and Lodi, E. (1988), **Structured Programming Using Turbo BASIC**, Academic Press Inc, Scott Valley.

Elmaghraby, S.E., (1966), **The Design Of Production Systems**, Rheinhold Publishing Corp.

Eloranta E, (1988), "Managing Production For Customer Service And Plant Efficiency," **Industrial Engineering**, Vol. 20(3), pp. 36-49.

Elsayed, E.A. and Boucher, T.O., (1985), **Analysis and Control of Production Systems**, Prentice-Hall, Englewood Cliffs.

Everdell, R., (1981), "MRPII, JIT and OPT : Not A Choice But A Synergy" in **APICS Readings in Zero Inventory**, pp. 144-1490.

Falster, P., (1987), "Planning And Controling Production Systems Combining Simulation And Expert Systems," **Computers In Industrial Engineering**, Vol. 8(2-3), pp. 161-172.

Falster, P. and Mazumder, R.B., (1984), **Modelling Production Management**, North Holland, Amsterdam

Fearson, H.E., Ruch, W.A. et al., (1983), **Fundamentals Of Producton/Operations Management 2ed**, West Publishing Co.

Finch, B.J. and Cox, J.F., (1986), "An Examination Of JIT Management For The Small Manufacturer : With An Illustration," **IJPR**, Vol. 24(2), pp. 329-342.

Fishman, G.S. and Kiviat, P.J., (1967), "The Analysis of Generated Simulation Time Series," **Management Science**, Vol. 13(7), pp.

Foysyth, R., (1984), **Expert Systems : Principles and Case Studies**, Chapman and Hall.

Franta, W.R., (1977), **The Process View Of Simulation**, North-Holland, Amsterdam.

Frazier, J.M. and Cannon, N.P., (1987), "Using Spreadsheets (Practical Guidelines For The Design Of Menus)," **Computers in Engineering**, Vol. 13(1-4), pp. 300-303.

Frijters, L., Kok, T. et al., (1989), "Lot-Sizing And Flow Production In An MRP-Environment," **Engineering Costs and Production Economics**, Vol. 17(1-4), pp. 65-70.

Fry, T.D., Philipoom, P.R. et al., (1989), "Due Date Assignment in a Multistage Job Shop," **IEE Trans (IE R&D)**, Vol. 21(2), pp. 153-161.

Fujita, S., (1978), "The Application of Marginal Analysis To The Economic Lot Sizing Problem," **AIIE Transactions**, Vol. 10, pp. 354.

Gilbert, P., (1983), **Software Design And Development**, SRA.

Goldratt, E.M., "Computerised Shop Floor Scheduling," **IJPR**, Vol. 26(3), pp. 443-455.

Goldratt, E.M., (1985), "Devising A Coherent Production-Finance-Marketing Strategy Using The OPT Rules," **BPICS Control**, Vol. 11(3), pp. 7-12.

Grant, H.K., (1988), "Simulation in Designing and Scheduling Manufacturing Systems" in **Design and Analysis of Integrated Manufacturing Systems**, National Academic Press, pp. 134-147.

Graves, S.C., (1981), "A Review Of Production Scheduling," **Operations Research**, Vol. 29(4), pp. 646-675.

Green, F.E., (1989), "When JIT Breaks Down On The Line," **Industrial Management**, Vol. 31(1), pp. 26-29.

Groover, M.P., (1987), **Automation, Production Systems, and Computer Integrated Manufacturing**, Prentice-Hall, Englewood Cliffs.

Guenther, W.C., (1973), **Concepts Of Statistical Inference**, McGraw Hill, New York.

Gupta, D. and Buzzacott, J.A., (1989), "A Framework For Understanding Flexibility of Manufacturing Systems," **Industrial Management**, Vol. 8(24), pp. 89-97.

Gupta, M.C., Judt, C., et al., "Expert Scheduling System For A Prototype Flexible Manufacturing cell: A Framework," **Computers and O.R.**, Vol. 16(4), pp. 363-378.

Gupta, T. and Ghosh, B.K., (1989), "A Multi-Objective Two-Stage Approach To the Design Of Communication Networks Subject To Production Constraints for Computerised Manufacturing Systems," **IJPR**, Vol. 27(7), pp. 587-598.

Hackman, S.T. and Leachman, R.C., (1989), "A General Framework For Modelling Production," **Management Science**, Vol. 35(4), pp. 478-495.

Hall, P.E. and Holden, G.K., (1973), **Computer Guide 9: Production Control**, NCC.

Harrison, M.C., (1985), "The Concepts Of OPT - The Way Forward," **BPICS Control**, Vol. 11(4), pp. 7-15.

Haynsworth, H.C., (1984), "A Theoretical Justification For JIT Scheduling," Production and Inventory Management, Vol. 25(1), pp. 1-3.

Heiko, L., (1989), "JIT and Fermat's Principle of Least Time," **Industrial Management**, Vol. 31(4), pp. 19-21.

Higgins, J.C., (1985), **Computer Based Planning Systems**, Edward Arnold.

Hilton, H.P. and Proth, J.M., "Using Timed Petri Nets For The Scheduling Of Job Shop Systems," **Engineering Costs & Production Economics**, Vol. 17(1-4), pp. 149-154.

Hinkman, R., (1987), "Combining JIT and MRP," **Production Engineer**, April, pp. 35-36.

Hitomi, K., (1979), **Manufacturing Systems Engineering**, Taylor and Francis.

Hogg, R. V. and Ledolter, J., (1987), **Engineering Statistics**, McMillan, New York.

Huber, L.E., (1988), "Using Project Management Techniques In Manufacturing Systems," **Industrial Management**, Vol. 30(2), pp. 19-22.

Hunt, V.D., (1987), **Dictionary of Advanced Manufacturing Technology**, Elsevier, New York.

Hurrion, R.D. (ed), (1988), **Simulation**, IFS, Bedford.

Ignizio, J.P., (1982), **Introduction To Linear Goal Programming**, Sage.

Ignizio, J.P., (1982), **Linear Programming In Single and Multiple Objective Systems**, Prentice-Hall, Englewood Cliffs.

Infotech Ltd, (1980), **Infotech State Of The Art Report**, Infotech Limited, Vol. 8(6).

Jagdev, H. and Davies, B.J., (1981), "An Interactive Job Shop Control System," in **Proceedings of the 22nd International Machine Tool Design and Research Conference**, pp. 39-45, Macmillan, New York.

John, T.C., (1989), "Tradeoff Scheduling In Single Machine Production Scheduling For Minimising Flow Time and Maximum Penalty," **Computers in O.R.**, Vol. 16(5), pp. 471-479.

Johnson, L.A. and Montgomery, D.C., (1974), **Operations Research In Production Planning, Scheduling and Inventory Control**, Wiley, New York.

Johnson, S.M., (1954), "Optimal Two- and Three-Stage Production Schedules With Set Up Times Included," **Naval Logistics Research**, Vol. 1(1), pp. 61-68.

Karmarkar, U.S., (1987), "Lot Sizes, Lead Times and In-process Inventories," **Management Science**, Vol. 33(3), pp. 409-423.

Kendall, D.G., (1953), "Stochastic Processes Occurring In The Theory of Queues and Their Analysis by the Method of Imbedded Markov Chains," **Anals of Material Statistics**, Vol. 24, pp. 338-354.

Khatu, S., (1979), "Computerised Job Shop Scheduling For Smaller Companies," **Tool and Production (USA)**, Vol. 44(11), pp. 100-101.

Kim, Y., "On the Superiority of a Backward Approach in List Scheduling Algorithms for Multi-Machine Makespan Problems," **IJPR**, Vol. 25(12), pp. 1751-1759.

King, J., (1985), **Management Of Engineering Production**, New York University Press.

Knasel, T.M., (1986), "Artificial Intelligence In Manufacturing : Forecasts For The Use Of AI In The USA," **Robotics**, Vol. 2(4), pp. 357-362.

Kochlar, A.K., (1979), **Development Of Computer Based Production Systems**, Edward Arnold.

Koelsch, J.R., (1988), "Software an Uphill Battle," in **Machine & Tool Blue Book**, pp. 47-66.

Koulams, C.P. and Smith, M.L., (1988), "Look Ahead Scheduling For Minimising Machine Interference," **IJPR**, Vol. 26(9), pp. 1523-1533.

Krajewski, L.J., King, B.E. et al., (1987), "Kanban, MRP, And Shaping The Manufacturing Environment," **Management Science**, Vol. 33(1), pp. 39-57.

Krupp, J.A.G., (1984), "Why MRP Systems Fail : Traps To Avoid," **Production and Inventory management**, Vol. 25(3), pp.48-53.

Kusiak, A. (ed), (1988), **Artificial Intelligence In Industry : Implications For CIM**, IFS.

Lang, D.W., (1977), **Critical Path Analysis**, Hodder and Stoughton.

Law, A.M. and Larmey, C.S., (1984), **An Introduction to Simulation Using SIMSCRIPT II.5**, CACI, Los Angeles.

Lee, S.M. and Ebrahimpour, M., (1987), "Just-In-Time," **Management Decisions**, Vol. 25(6), pp. 50-54.

Lilly, R.T., (1986), "Successor To MRP II," **Manufacturing System**, Val 4(12), pp. 38.

Lue, T.W., (1987), "Using Integrated Spreadsheets For Production And Facilities Planning," **Computers In Industrial Engineering**, Vol.13(1-4), pp. 88-91.

Manning, E.J., (1985), "OPT: A Production Technique Case Study," **BPICS Control**, Vol. 11(3), pp. 17-21.

Mather, H., (1982), "Making It All Work," **Production and Inventory Management**, Vol. 23(3), pp. 67-73.

McManus, J.J., (1987), "Control In The OPT Environment," **Production Engineer**, July, pp. 20-21.

Meleton, M.P., (1986), "OPT - Fantasy Or Breakthrough?," **Production and Inventory Management**, Vol. 27(2), pp. 13-21.

Melnyk, S.A. and Carter, P.L., (1987), **Production Activity Control**, Dow Jones-Irwin, Illinois.

Miller, R., (1985), "Artificial Intelligence : A New Tool For Manufacturing," **Manufacturing Engineering**, April, pp, 56-62.

Mohanty, R.P. and Govindrajan, S., (1989), "Design of a Production Planning and Control System For a Toolroom: A Case Study," **Engineering Costs and Production Economics**, Vol. 16(1), pp. 81-90.

Moore, F.G. and Hendrick, J.E., (1980), **Production/Operations Management 8ed**, Irwin Inc.

Muth, J.F. (ed.), (1963), **Industrial Scheduling**, Prentice-Hall, Englewood Cliffs.

Nellemann, D.O. and Smith, L.F., (1982), "JIT vs JIC Production/Inventory Concepts Borrowed Back From Japan," **Production and Inventory Management**, Vol. 23(2), pp. 12-21.

Newman, P., (1988), "Scheduling In CIM Systems," in **AI In Industry**, Kusiak, A. (ed), pp. 359-402

Norton, N., (1988), "Breath New Life Into Your MRP System," **Production Engineer**, October, pp. 46-51.

Orlicky, J., (1975), **Material Requirements Planning**, McGraw-Hill, New York.

Painter, C.W., (1987), "Last Rites For MRP," **Production Engineer**, April, pp. 33-36.

Palaniswami, S., (1987), "Expert Simulation For Manufacturing Planning," **Engineering Costs and Production Economics**, Vol. 11(1), pp. 56-62.

Pegels, C.C., (?), "The Toyota Production System - Lessons For American Management," **International J. of Production Management**, Vol. 4(1), pp. 3-11.

Pendlebury, J. and Yeomans, J., (1985), "OPT : The Challenge To Decision Makers," **BPICS Control**, Vol. 11(4), pp. 17.

Plenert, G., (1985), "Are The Japanese Production Methods Applicable In The United States," **Production and Inventory Management**, Vol 26(2), pp. 121-129.

Plenert, G. and Best, T.D., (1986), "MRP, JIT and OPT : What's Best," **Production and Inventory Management**, Vol. 27(2), pp. 22-29.

Plossl, G.W., "Throughput Time Control," **International J. of Production Research**, Vol. 26(3), pp. 493-499.

Potts, C.N. and Wassenhove, L., (1988), "Algorithms For Scheduling A Single Machine To Minimise The Weighted Number Of Late Jobs," **Management Science**, Vol. 34(7), pp. 843-858.

Pounds, W.F., (1963), "The Scheduling Environment," in **Industrial Scheduling**, Muth, J.F. and Thompson, G.L., (eds.), Prentice-Hall, Englewood Cliffs, pp. 5-12.

Pritsker, A.A.B., (1988), "SLAM II as a Simulation Tool," **J. Of Computer Applications in Engineering**, Vol. 3(1), pp. 28-35.

Production Engineer, (1988), Manufacturing Debated at Cambridge, **Production Engineer**, Vol. 67(11), p. 3.

Raman, N., Rachamadugu, R.V. et al., (1989), "Real-time Scheduling of an Automated Manufacturing Centre," **European J. of O R**, Vol. 40(?), pp. 222-242.

Rathmill, K. (ed.), (1988), **Control And Programming In Advanced Manufacturing**, IFS.

Robertson, D.C., (1969), **Project Planning And Control : Simplified Critical Path Analysis**, Heywood.

Russell, R.S. and Sumichrast, R.T., (1989), "Using PERT and Simulation To Set Promise Dates," **Industrial Management**, Vol. 31(1), pp. 14-16.

Sahni, S., (1979), "Preemptive Scheduling With Due Dates," **Operations Research**, Vol. 27(5), pp. 925-934.

Sampson, J., (1988), "Focal Point Scheduling - A Study At Sheerness Steel," **Production Engineer**, April, pp. 19-23.

Sargent, R.G., (1988), "Event Graph Modelling For Simulation With An Application To Flexible Manufacturing Systems," **Management Science**, Vol. 34(10), pp. 1231-1251.

Schmidt, B., (1987), **Model Construction With GPSS-FORTRAN Version 3 (Advances In Simulation)**, Springer-Verlag.

Schmidt, J.W. and Taylor, R.E., (1970), **Simulation and Analysis of Industrial Systems**, Irwin, Homewood, Illinios.

Schonberger, R.J., (1984), "JIT Production Systems : Replacing Complexity With Simplicity In Manufacturing Management," **Industrial Engineering**, Vol, 16(10), pp. 52-61.

Schonberger, R.J., (1986), **World Class Manufacturing : The lessons Of Simplicity Applied**, Free Press.

Schrage, L.E., (1966), **Some Queuing Models for a Time-Shared Facility**, Cornell University, New York.

Sen, T., Raiszadeh, F. et al., (1988), "A Branch and Bound Approach To Bicriterion Scheduling Problem Involving Total Flowtime and Range Of Lateness," **Management Science**, Vol. 34(2), pp. 254-260.

Sepehri, M. and Raffish, N., (1986), "Developing And Implementing Control Systems For Repetitive Manufacturing," **Industrial Engineering**, Vol. 18(6), pp. 34-46.

Shannon, R.E., (1975), **Systems Simulation: The Art and The Science**, Prentice-Hall, Englewood Cliffs.

Shivan, J., Joyce, R., et al, "Production and Inventory Management Techniques - A Systems Prospective," in Kusiak, A. (ed.) **Modern Production Management Systems**, North-Holland, Amsterdam.

Shunk, D.L. (ed.), (1989), **Optimization of Manufacturing Systems**, North-Holland, Amsterdam.

SoftTech Inc, (1980), **Architect's Manual ICAM Definition Method : IDEF0**, CAM International, Indiana.

Southern, T., (1987), "Sorry But We Are Out Of Stock," **Production Engineer**, April, pp. 34-35.

Srikanth, M.L. and Cavallaro Jr., H.E., (1987), **Regaining Competitiveness (Putting The Goal To Work)**, Spectrum Publishing.

Stewart, S., (1982), "Using OPT To Schedule A Machine Shop," in **Proceedings of APICS 25th Annual Conference**, pp. 92-93.

Stidham, S., (1974), "A Last Word on $L = \alpha W$," **Operations Research**, Vol. 22(?), pp. 417-421.

Sundaram, R.M. and Sundrarajan, R.M., (1987), "Lot Sizing In Cellular Manufacturing Systems," **Computers in Industrial Engineering**, Vol.

13(1-4), pp. 55-60.

Szymankiewicz, J. and McDonald, J., (1988), **Solving Business Problems By Simulation (2nd ed)**, McGraw-Hill, New York.

Taffler, R., (1979), **Using Operational Research : A Practical Introduction To Quantitative Methods In Management**, Prentice-Hall, Englewood Cliffs.

Veral, E.A. and LaForge, R.L., (1990), "The Integration of Cost and Capacity Considerations in Material Requirements Planning Systems," **Decision Sciences**, Vol. 21(3), pp. 507-520.

Vollmann, T.E., (1986), "OPT As An Enhancement To MRP II," **Production and Inventory Management**, Vol. 27(2), pp. 38-47.

Vollmann, T.E., Berry, W.L. et al., (1987), **Manufacturing Planning and Control Systems**, Irwin.

Mayrhauser, A. Von, (1990), **Software Engineering: Methods and Management**, Academic Press Inc, Boston.

Voss, C.A. (ed.), (1986), **JIT Manufacture**, IFS.

Walker, T.C. and Miller, R.K., (1986), **Expert Systems 1986**, SEAI.

Which Computer?, (1983), "Survey: Production Control Systems," **Which Computer?**, pp. 62-70.

Wollam, C.R., (1988), "Two Job, M-Machine Sequencing Analysis," **Industrial Engineering**, Vol. 20(1), pp. 18-23.

Woodgate, H.S., (1977), **Planning By Network : Project Planning and Control Using Network Techniques**, Business Books.

Woolsey, R.E.D. and Swanson, H.S., (1975), **Operations Research For Immediate Application : A Quick And Dirty Manual**, Harper and Row.

Yoshikawa, H. and Burbidge, J., (eds.), (1987), **New Technologies For Production Management Systems**, North Holland, Amsterdam.