CRANFIELD INSTITUTE OF TECHNOLOGY

THE CIM INSTITUTE

PhD THESIS

Academic Year 1988

IP-SHING FAN

INTELLIGENT FLEXIBLE MANUFACTURING SYSTEM CONTROL

SUPERVISOR : DR P J SACKETT

SEPT 1988

SUMMARY

This research proposes a generic decision making system structure for real time despatch control in small Flexible Manufacturing Systems. This is to satisfy the requirement for low cost control systems that can be flexibly adapted to a wide range of production environments.

A simulation environment has been developed to emulate the detail real time despatch control of flexible manufacturing systems. This environment allows analysis of the decision making process and its effects. A model of a modular type FMS is used to study decision making in real time FMS control.

Real time control is dynamic, the decision criteria change with the production states of the system. Decision making is based on both quantitative and qualitative factors. Apart from production quantity and time which are quantitative, there are installation dependent and production situations better expressed in states which are non-numeric. The knowledge based representation developed from artificial intelligence work is superior in modelling both mathematical scheduling research and discrete states information.

Recognising the importance of system particular knowledge to effective control of the system, system independent functions are separated out to form elements of a generic control system architecture. This generic architecture contains elements of information handling to process information to service the scheduling decision making element. A core for regulating information flow and a data interface definintion allows this control architecture to be hardware independent. The decision making mechanism dependent on machinery hardware configuration and particular production characteristics can then be designed and interface to the architecture to form a complete control system.

A decision design methodology has been designed to guide the design of the scheduling decision making system. The methodology addresses the design of work queue formation timing and the characteristics for each resource in the system. These are then integrated into a complete work flow control system by the resolution of contentions between the individual queues.

The application of the design methodology and control system architecture is illustrated.

"…..to separate the essential from the accidental."

Sherlock Holmes
The Adventure of the Priory School.

ACKNOWLEDGEMENT

Greatest thanks go to my parents and my brother and sisters who looked after them while I was away from home.

Sincere thanks to Dr P J Sackett, without whose guidance and support this thesis could not be completed.

The great hospitality of the various English and international friends is much appreciated, which made my stay most enjoyable. The list is too long to be detailed.

The Association of Commonwealth Universities is thanked for the award of the Scholarship that made this study possible. Thanks also go to the efficient staff of the Commonwealth Scholarship Commission and The British Council who administered the Scholarship.

Last but not least, grateful thanks to fellow colleague Alistair Heslop who performed a painstaking proofread that root out most, if not all, of the syntax errors in the thesis. The responsibility of the correctness of both the syntax and contents of this thesis can only be mine.

AUTHOR

Ip-shing Fan studied Industrial Engineering at the University of Hong Kong and graduated with First Class Honours in 1983. He completed graduate engineer training with Qualidux Industrial Co Ltd., a major plastic product manufacturer in Hong Kong. He gained experience in plastic processing and metal working technologies, especially in mould and die making. He became a specialist in CNC machines operations, having commissioned a 3-1/2 axis CNC-copy milling machine and completed an Endorsement Certificate course with distinction on CNC Machine Tool Technology and Practice in the Hong Kong Polytechnic. He was awarded the British Commonwealth Scholarship and started his PhD research programme with the Computer Integrated Manufacturing Institute in Cranfield Institute of Technology in 1985.

His research interest covers the whole spectrum of manufacturing control, with specific emphasis on application to small business common in the Far East. He has a special interest in personal computers and their applications in providing affordable computer technology in enhancing productivity.

Publications related to the work done on this thesis are:
Fan I S, Sackett P J, "A Prolog simulator for interactive flexible manufacturing systems control", Simulation, v50 n6, June 1988.

Sackett P J, Fan I S, "Goal identification for flexible manufacturing system control", IFIP Conf Workshop on Knowledge Based Production Management Systems, Galway Aug 23-25, 1988.

Fan I S, Sackett P J, "An universal architecture for flexible manufacturing cell control", Proc 4th Int Conf on Computer Aided Production Engineering, 20-23 Nov, 1988.

Paper accepted for publication:
Sackett P J, Fan I S, "The control of a flexible manufacturing system by short term goal identification", International Journal of Advanced Manufacturing Technology.

Publications in preparation are:
Nelder G, Sackett P J, Fan I S, "Process planning for effective use of flexible manufacturing systems".

Sackett P J, Fan I S, "Decision making design for flexible manufacturing systems control".

Fan I S, Sackett P J, "Database design for real time flexible manufacturing systems control".

CONTENTS

CHAPTER1 INTRODUCTION
1.1 STATEMENT OF PROBLEM 1
1.2 OBJECTIVE 2
1.2 RESEARCH METHODOLOGY 3
1.4 THESIS STRUCTURE 5

CHAPTER 2 FLEXIBLE MANUFACTURING SYSTEM
2.1 DEVELOPMENT HISTORY 7
2.2 CLASSIFICATION 11
2.3 ROLES OF DESIGN AND CONTROL 12
2.4 OPERATION CONTROL OF FLEXIBLE MANUFACTURE . . 16
2.4.1 Operation objectives 16
2.4.2 Control factors 18
2.5 CONTROL SYSTEMS IMPLEMENTATION. 21
2.6 GENERIC CONTROL SYSTEM 23

CHAPTER 3 SCHEDULING CONTROL RESEARCH REVIEW
3.1 JOB SHOP SCHEDULING 25
3.2 ARTIFICIAL INTELLIGENCE 28
3.3 HIERARCHY AND STRUCTURE OF FMS DECISION MODELS . 32
3.4 QUEUEING MODELS 34
3.5 SIMULATION AND HEURISTICS 35
3.6 KNOWLEDGE BASED SCHEDULING 37
3.7 INTELLIGENT CONTROL SYSTEMS 39

CHAPTER 4 PILOT RESEARCH MODEL
4.1 OBJECTIVES 42
4.2 SIMULATION FOR REAL TIME CONTROL 42
4.3 KNOWLEDGE BASED MODEL 43
4.4 MODEL OF MODULAR SMALL FMS CELL 45
4.4.1 FMS model hardware 45
4.4.2 System component characteristics 46
4.4.3 System assumptions 47
4.5 CHAMELEON - A PROLOG FMS ENVIRONMENT . . . 48
4.5.1 Prolog simulation 48
4.5.2 Chameleon simulator 50
4.5.3 Prototype simulator structure. 57
4.6 RESULTS AND CONCLUSIONS 63
4.6.1 Production data set 63
4.6.2 Results 63
4.6.3 Findings 68

CHAPTER 5 GENERIC CONTROL ARCHITECTURE
5.1 CONTROL. 71
5.2 FORMAL SCHEDULING DECISIONS STRUCTURE . . . 72
5.2.1 Generic resource decision model 72
5.2.2 Integrating queues 76
5.3 UNIVERSAL CONTROL ARCHITECTURE 77
5.3.1 De-coupling control from scheduling 77
5.3.2 Control core 77
5.3.3 Scheduling engine 78
5.4 FMS SCHEDULING DECISIONS 80
5.4.1 Tool and fixture configuration 80
5.4.2 Job scheduling 80
5.4.3 Material handling system handling 81
5.4.4 Interaction of scheduling decisions 81
5.5 KNOWLEDGE FOR JOB SCHEDULING 82
5.5.1 Information requirement 82
5.5.2 Knowledge for good decisions 83
5.6 DATA INTERFACE 84
5.6.1 Forms of data 84
5.6.2 Lifespan of data 84
5.6.3 Static database 85
5.6.4 Dynamic database 86
5.6.5 Communications with low level devices 88

CHAPTER 6 SCHEDULING SYSTEMS DESIGN CASES
6.1 A SCHEDULING SYSTEM DESIGN METHODOLOGY . . 89
6.2 A DESPATCH SYSTEM FOR PRODUCTION SHOP . . . 91
6.3 A DESPATCH SYSTEM FOR JOB SHOP 92
6.4 AN ADAPTIVE RULES SYSTEM FOR DYNAMIC SHOP . . 93
6.5 MATCHING SCHEDULING SYSTEM TO PRODUCTION . . 96

CHAPTER 7 CONTROL ARCHITECTURE IMPLEMENTATION
7.1 CONTROL ARCHITECTURE DESIGN 97
7.1.1 Control core 97
7.1.2 Knowledge based job scheduling modules 98
7.1.3 Transport scheduling module 105
7.2 CODING SCHEDULING MODELS 106
7.2.1 Simple production shop despatch scheduling 106
7.2.2 Goal identification adaptive scheduling 109
7.3 SIMULATION ILLUSTRATIONS 115
7.3.1 Simulation experiments 115
7.3.2 Validation of control structure 119

CHAPTER 8 DISCUSSION AND CONCLUSION
. 120

CHAPTER 9 RECOMMENDATIONS FOR FURTHER WORK 122
REFERENCES 123
APPENDIX
APPENDIX A WORK PLAN DETAILS 137
APPENDIX B RULE FORM OF GOAL IDENTIFICATION DESPATCH 142

FIGURES

1	A Modular Flexible Manufacturing System	1
2	Role of generic control systems	2
3	'Define-build-learn' cycle	3
4	KTM 'step-by-step' FMS	10
5	Longbridge FMS layout	14
6	Anderson Strathclyde FMS layout	14
7	BAe Preston FMS layout	15
8	Hattersley Newman Hender FMS layout	15
9	Generic controllers within the manufacturing spectrum	23
10	Elements of Artificial Intelligence	28
11	Knowledge based vs conventional programs	29
12	Knowledge based model	44
13	Universal FMS control model	45
14	Chameleon screen - user interaction	51
15	Chameleon event log	53
16	Sample Chameleon reports	54
17	Analysis of utilisation log	55
18	Chameleon simulation logic	57
19	Pilot Chameleon structure	58
20	Chameleon screen - event queues	60
21	Event queue polling logic	61
22	Real time solution vs Steady state solution	71
23	Jackson network of queue model	73
24	Solberg network of queue model	73
25	Intelligent processor network of queue model	73
26	Universal architecture model	77
27	Functional model of scheduling engine	79
28	Database organisation of universal architecture	85
29	Scheduling design methodology	89
30	Adaptive job selection rule	95
31	Improved Chameleon structure	97
32	Scheduling modules hierarchy	98

LIST OF TABLES

Table 1 Configuration of pilot experiment 64
Table 2 Production results for pilot experiments 66
Table 3 Start up times for pilot experiments 67
Table 4 Production results for demonstration experiments . . . 117
Table 5 Start up times for demonstration experiments 118

LIST OF LISTINGS

Listing.1 Chameleon kernel 98
Listing.2 Message interpreter 99
Listing.3 System mode monitor 100
Listing.4 Job control module 100
Listing.5 Job assign module 101
Listing.6 Job select module 102
Listing.7 Job selection rules 103
Listing.8 Despatch shop message interpreter 106
Listing.9 Despatch shop decision router 107
Listing.10 Despatch shop job control 108
Listing.11 Despatch shop job assign 108
Listing.12 Goal scheduling message interpreter 109
Listing.13 Goal scheduling decision router 110
Listing.14 Goal scheduling job control 112
Listing.15 Goal scheduling rule list monitor 114

GLOSSARY

AI	- artificial intelligence
AMRF	- Automated Manufacturing Research Facility
AGV	- automatic guided vehicle
CMM	- co-ordinate measurement machine
CNC	- computer numerical control
DNC	- direct numerical control
FMS	- flexible manufacturing system
GT	- group technology
ISO	- International Standards Organisation
PLC	- programmable logic controller
RGV	- rail guided vehicle

CHAPTER 1
INTRODUCTION

1.1 STATEMENT OF PROBLEM

The hardware development of flexible manufacturing technology has advanced to the stage where linking of standard machine tools with the necessary material transport system is readily available [Fig.1]. It is now economically viable for small to medium shops to replace conventional machining practice. In the wide variety of prospective production shops that can now apply flexible manufacturing technology, there are correspondingly different production control environments and dynamics. There is a need for work control systems that can match the variety of production requirements to the simple modular approach of flexible manufacturing systems [Fig.2]. Classical methodologies designed for custom implementing large flexible manufacturing systems are expensive in the context of small systems. They do not meet the operational requirements for these classes of FMS. This inhibits the further take up of FMS.

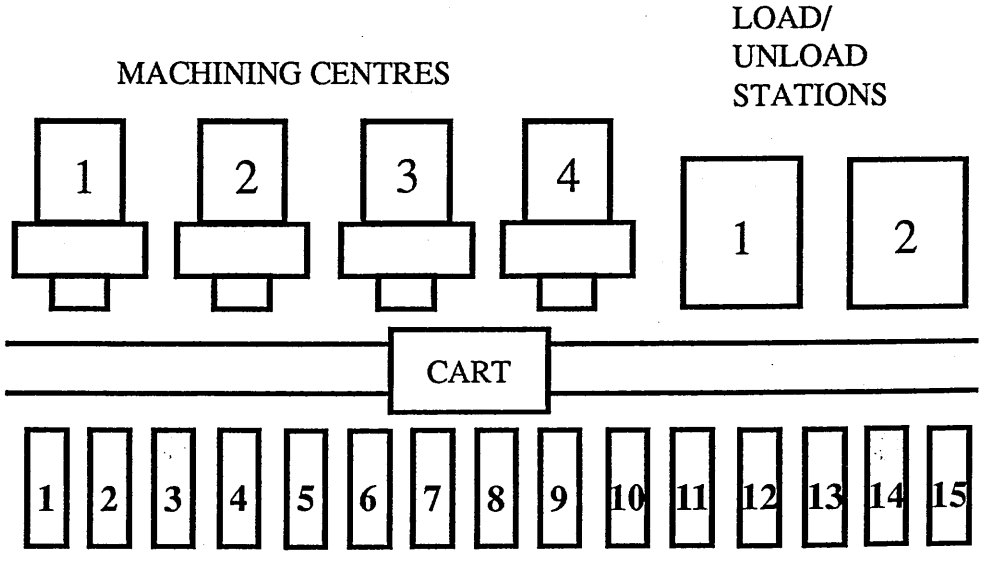


Fig 1. A Modular Flexible Manufacturing System

1.2 OBJECTIVE

This research investigates the development of a generic control system framework which can be rapidly and economically tuned to the particular requirement of flexible manufacturing applications. It was believed that production scheduling research combined with artificial intelligence techniques could achieve intelligent real time control of flexible manufacturing systems.

The objective was to investigate the application of artificial intelligence techniques to the structure of intelligent flexible manufacturing system control. The goal is to establish a framework that can accommodate appropriate solutions to different requirements rather than to find a better system for a particular problem. The implementation of this structure could then be explored with the building of prototypes.

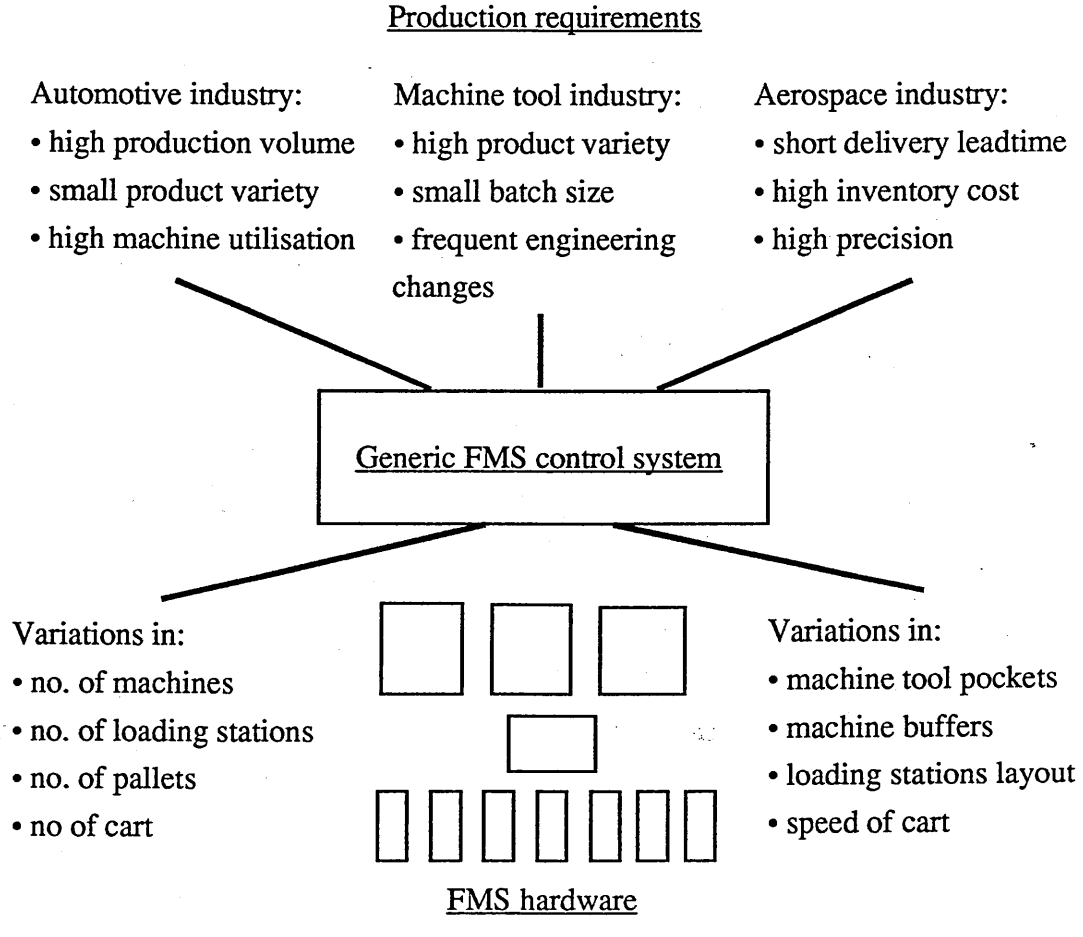


Fig 2. Role of generic control system

1.3 RESEARCH METHODOLOGY

It was proposed to emulate the development of flexible manufacturing systems control systems and gather the knowledge of flexible manufacture control concepts through this development process. In this development, a 'define-build-learn' cycle [Fig.3] was used with a specially developed simulation system. Functional concepts for flexible manufacture control were first defined and built into the system, the corresponding system characteristics were then studied, and the results used for developing further concepts definition and build.

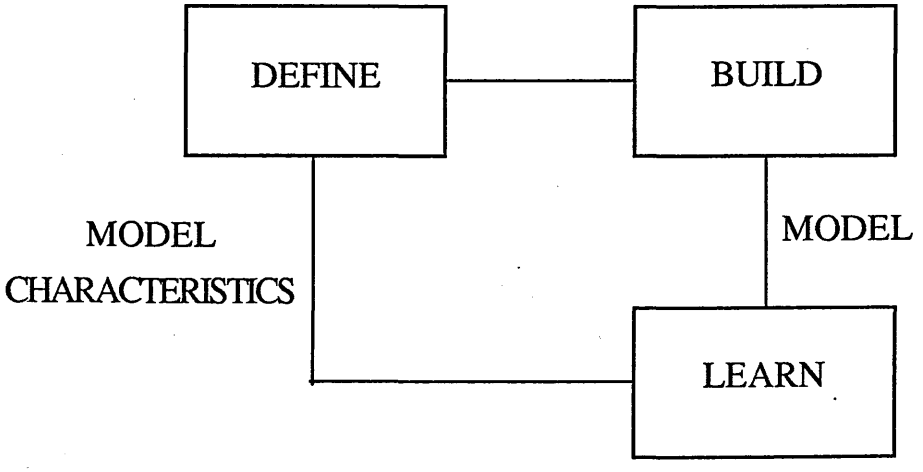


Fig 3. Define-build-learn cycle

A simulation environment, 'Chameleon' was conceived, structured and established for the study of control systems building. This environment and all software work was implemented in Prolog to exploit the power of advanced knowledge representation techniques. This simulation environment allows the production characteristics and the control concepts of small flexible manufacturing systems to be investigated in detail. It was developed with extensive user interface and interaction points. The flexible manufacturing system hardware model chosen was coded with the comprehensive interaction logic and data interface as defined in Chameleon to work with different scheduling systems. The 'define-build-learn' cycle was then used to develop the inference engine for job scheduling; increasing levels of intelligence were incorporated with each development.

A pilot FMS model was used to investigate the suitability of Prolog as a FMS simulation language and explore the difficulties of the FMS scheduling problem. Results of this pilot study were used to refine the decision making control model and improve the final version of Chameleon.

To ensure the relevance of research to industrial needs, literature research was backed up by an attachment to Kearney Trecker and Marwin Ltd in Brighton, UK. KTM is one of the leading FMS suppliers in Europe with more than 20 FMS installations; most of these are small systems consisting of modular machinery and can be expanded in a step-by-step manner. Site visits during research included:

Kearney Trecker and Marwin, Brighton;
Austin Rover, Longbridge;
Hattersley Newman Hender, Ormskirk;
Anderson Strathclyde, Glasgow;
Caterpillar, Motherwell - closed;
Mazak, Worcester;
British United Shoes Machinery, Leicester;
Plessey Aerospace, Tichfield;
Edwards High Vacuum, Shoreham.

Relevant exhibitions attended included:

CADCAM 86, 87, 88;
AMT-CIM 86, 87;
Automan 87;
PEP 87;
Mach 88.

A thesis project undertaken by a MSc student was initiated in 1987 to study the various factors considered important. This demonstrated the complexity of factors involved and their interactions.

The concepts of this research were applied in two further MSc student projects in 1988 that successfully developed decision support systems for scheduling of group technology cells in sites of the industrial sponsors.

1.4 THESIS STRUCTURE

Chapters One and Two define the problem and its context. The history of flexible manufacturing systems and its impact on piece part manufacturing is traced. The different roles of FMS hardware design and real time control are discussed. Issues in the operation of FMS and the detail implementation into control systems are presented. The importance of a 'generic' control system is highlighted.

Chapter Three reviews previous work in the field of scheduling with specific applications in FMS. The different approaches used in conventional research to attack the large and multi-facet job shop scheduling problem are outlined. A brief account of artificial intelligence research work relevant to scheduling research is given. The work in structuring of the FMS control problem together with the different models in solving particular problems are studied. The use of simulation models and knowledge based techniques as more flexible tools to investigate FMS problems is presented. Current work in developing intelligent control systems are discussed.

Chapter Four reports the work in a pilot model to study intelligent control of flexible manufacturing systems. Artificial intelligence techniques in the form of knowledge based model and Prolog language are experimented for the representation of control systems. The FMS model and the related assumptions are described. The design of the Chameleon FMS simulation environment is detailed. Results of pilot simulation experiments are presented.

Chapter Five formalises a general framework to accommodate decision making in the real time control of flexible manufacturing systems. Control theory concepts are used in the analysis of FMS control decision needs. The network of queues model is extended to locate decision timings and choices for FMS control. The design of scheduling systems becomes an integration of individual workstation queue formation decisions into a coherent system to resolve the contention problems in decisions. An architecture is proposed for FMS control consisting of a control core that is hardware independent to form the interface between machinery hardware and scheduling system. A data analysis of the decision making requirement of FMS control results in a data interface that isolates the scheduling control functions and low level communications functions.

Chapter Six refines FMS control decision making design into a methodology. The

application of the methodology is illustrated by the design of sample scheduling systems for very different production environments. The inherent flexibility of a modular flexible manufacturing system hardware to satisfy demands for different production requirements is demonstrated. The importance of matching an equally flexible scheduling control system to exploit the capability is discussed. The difficulties in quantifying the success of control design processes is emphasised.

Chapter Seven describes experience in implementing the decision architecture and the scheduling systems developed. The coding of the implementations of the scheduling systems for the production shop and the goal identification system is used to illustrate the power of the architecture concept. This points to the path of implementation of the results of this work.

Chapter Eight further discusses the work of the thesis and the conclusions.

Chapter Nine suggests means of implementing the architecture and further research work based on the architecture.

CHAPTER 2
FLEXIBLE MANUFACTURING SYSTEMS

2.1 DEVELOPMENT HISTORY

"For 150 years batch manufacture of engineering components has been carried out in factories using these (conventional) machine tools as originally intended. Big factories have had more machine tools and more people but always the same basic organisation, or lack of it. Because the organisation has not changed, increased size has destroyed communication, and is destroying skill. A different type of organisation could use the same men and equipment with great benefits if it were oriented towards the current problems" [1]. Williamson, the 'father' of Flexible Manufacturing Systems embarked on the revolution to use computers in a new organisation of production in 1960s. We are still in the process of integrating computers intelligently to manufacturing environment for ultimate human benefits.

Application of computers to control machine tools dates from the 1950's. Massachusetts Institute of Technology was sponsored by the United States Air Force to develop computer controlled numerical machines to machine complex contour surfaces needed for aerospace production. The result of this development is machine tools that can perform complex cutting motions in multi-axis machining. The cost and reliability of computers limited this technology to certain high cost and complex product machining, especially in aerospace industries.

Conventional machine tool productivity has been progressively improved using various automation techniques. A major weakness in conventional machining is the limited number of operations a machine can perform given a tooling setup. Setup times for the preparation of the tools and fixtures are high when compared with processing times. Automatic machines using mechanical cams, electrical sequencers and programmable logic controllers were developed to combine more operations together with only one setup. They are effective in large volume production where the cost of the lengthy setup can be shared by a long production run. The study of economic batch sizing is a response to this problem.

The cost of computers has now dropped to the point where previously expensive technology becomes affordable. Computer numerical control technology combined with conventional automation technology in new generation machine tool controls. CNC is

used to combine a number of simple operations, pocket milling and drilling cycles and perform complex machining. The operations flexibility of CNC machines is used to reduce the cost of setup, in both time and tooling. Sackett and Beddis [2] developed a tooling grouping approach for CNC machine shop organisation to maximise benefits from stand alone CNC machines operations. Carter [3] discussed the importance of better utilisation of CNC machines with DNC control. To further exploit the computer flexibility in reducing setup cost, automatic tool and pallet changes were developed to allow setup operations be done away from the machines, further increasing machine utilisation. It is only natural that automatic material handling systems be developed to link individual CNC machines together to reap further benefits of computer controlled production, giving rise to flexible manufacturing systems.

The first documented production system that encompassed this technology and indeed led to patents of flexible manufacturing system is the Molins System 24 built for machining aluminium parts for cigarette making machineries. Williamson [1,4,5] detailed the concept and rationale for a new manufacturing organisation to perform batch manufacture, most of the logic rings as true now as it did 22 years ago. However, the full system built for Molins was stopped before it went into production. IBM in Rochester, USA also operated a system for a period of time. However, these systems were before their time; though the basic technologies were available, they were not mature enough to attract sufficient resources to eliminate all the difficulties.

Advanced computer technology enabled further development of flexible manufacturing systems in the 1970's. Several widely publicised systems were developed in USA and Japan[6,7,8,9]. Various surveys put the worldwide FMS population in 1980 at between 60 to 125. The late 1970's were a time of awakening for most Western manufacturing countries, with the sudden realisation of Japanese eminence in consumer products production after the recession resulting from the 1973 oil crisis. A general vision of Japanese superiority in the areas of robotics, computers and production concepts came as a severe cultural shock. Various national projects were initiated to ensure parity in manufacturing efficiency, including the Automated Small-batch Production programme and SCAMP FMS in UK, the Advanced Manufacturing Research Facility of the National Bureau of Standards in USA and the Methodology for Unmanned Manufacture and Laser FMS Complex in Tsukuba, Japan. Considerable academic research in studying FMS characteristics and application of production control techniques was done.

A common feature of these pre-1980 first generation systems is that they were specially

designed and built for particular applications. There were a variety of machines, material transfer and computer control solutions, each used according to applications requirements and supplier capability and preference. Machine tool suppliers tested their equipment in integrating machinery hardware and computer communication and control software. Through the experience of these systems, techniques for the design and operation of FMS were developed, as in simulation, probing, tooling, communication protocol and financial justification.

By 1980, hardware development began to stabilise. Machines with automatic tool change and suitable interface to automatic material transfer systems became established. Major suppliers in the world market emerged as White-Sundstrand, Kearney & Trecker, Cincinnati Milacron in USA; Yamazaki, Toyada, Mori-Seiki from Japan, KTM from UK, Scharmann, Deckel and Maho from West Germany, Comau, Mandelli from Italy. Automatic guided vehicle is the material transfer system of choice for large systems with complex layout; rail guided system for smaller system that can be laid in a straight line. Cell control computers are invariably PDP or VAX from Digital Equipment. Tool replenishment systems became a key feature in the newer generation systems as this was a weak link in the first generation systems. For complex systems, the inclusion of wash stations and coordinate measurement machines with on-line statistical process control became popular. All these systems were designed around a particular family of products.

Whilst many of these massive projects were underway, another development was in the form of individual CNC machines equipped with automatic tool change and automatic pallet change systems. Some of these systems use a common carousel pallet pool accessed by two machines. These systems require more worker attention in general and are usually run manned, they are much more efficient than stand alone machines. They are particularly well accepted in Japan [10,11] as they provide a cheap method in attaining some benefits of FMS. Because of the constant operator/s presence, these systems are in fact more flexible than others as operators can rectify production complications and attend to production changes.

The cost of FMS technology impedes its widespread acceptance. Apart from the machines, the cost for the material handling systems and computer systems is high, with system integration and design to top everything up. Another approach championed by KTM in UK and adopted by many other suppliers is the 'step-by-step' approach [12][Fig.4]. The system is built up of standard machines, load/unload stations and

Step-by-step Manufacturing Systems

KTM pioneered the concept of step-by-step manufacturing systems; and are still the only one to spell out in detail the significant elements of each step. Not only the technical features but also the benefits.

The critical parameters established by KTM remain unchanged.

The rate of change is dictated by the user based on his ability to assimilate the technology, financial resources and market demands.

Each step must utilise previous hardware.

Each step must be a viable production system in its own right to generate the finances for the next step.

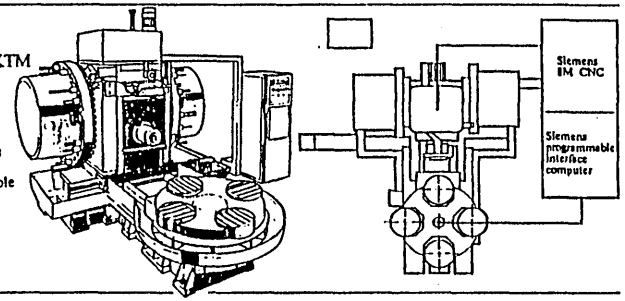
Below the progress of a typical manufacturing cell is charted.

Stage 1

Stand alone 4 pallet KTM Flexlmatic FM100 machining centre

☐ Benefits

All the essential elements of FMS;
A machining centre capable of unmanned operation.

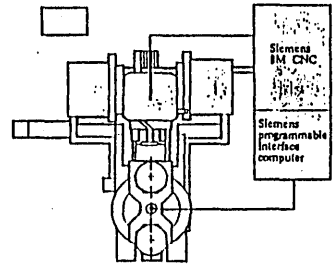


Stage 2

Stand alone 2 pallet KTM Flexl-matic FM100 machining centre. Linking of machine envisaged.

☐ Benefits

Similar to the 4 pallet machine, but configured for easy up-grading to machining cell capability.

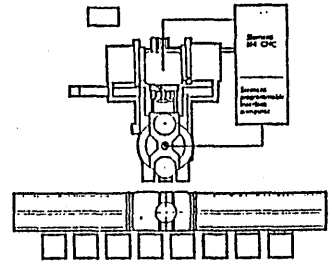


Stage 3

Stand alone multi-pallet machining cell. Up to 15 pallets per machine. Rail-guided pallet transport vehicle controlled from CNC system.

☐ Benefits

Extended unmanned capability through additional pallet storage.

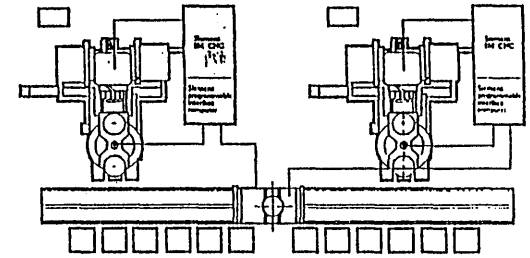


Stage 4

Up to 4 machines linked together. Up to 15 pallets per machine. Pallets are dedicated to each machine. Load/unload at each pallet stand. Rail-guided vehicle controlled from CNC systems.

☐ Benefits

Increased manufacturing capability without increasing transporter costs.

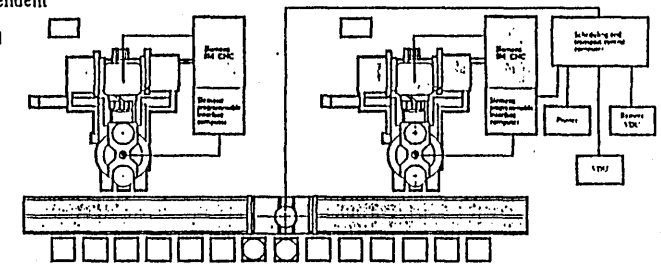


Stage 5

Up to 4 machines linked together. Up to 15 pallets per machine. Independent scheduling and transport control computer. Any pallet to any machine. Dedicated load/unload stations.

☐ Benefits

Introduction of scheduling computer and dedicated load/unload stations dramatically increases flexibility of installation.



Stage 6

Full FMS host computer

☐ Benefits

Host computer upgrades system to full FMS with all the interface capabilities for the true 'paperless' design-to-production manufacturing facility.

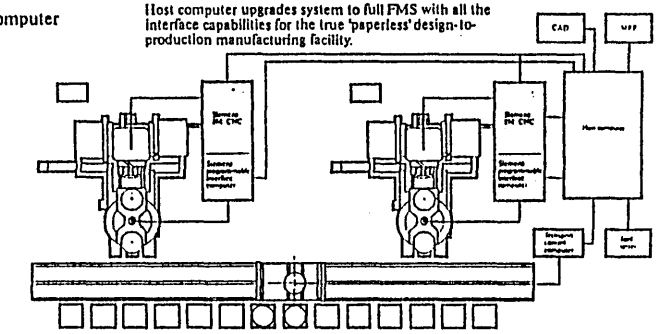


Fig 4. KTM 'step-by-step' FMS

pallet stands linked by an extendable rail guided vehicle transport system. Implementation of this system can be done in stages, with the first stage in production helping to pay for the next. A standard computer control architecture forms the backbone that links all the component machines together.

The current marketplace for flexible manufacture falls into three distinct categories. Large systems, tailor designed, for production with sufficient volume to justify the extra efficiencies gained from the high engineering effort. This may be a flexible flow line type system that minimises the transfer cost between stations when the production process is known. In the case of large product variety or dynamic production demands that do not permit easy optimisation, a system can be built from standard modules that is 'general purpose' and allows step by step expansion. This category of system is attractive as machine shop replacement when the expected machining requirement is known but the exact demand type and pattern is indeterminate, as in general component machining or subcontracting. For production that demands great flexibility and better equipment utilisation than stand alone CNC machines, addition of simple pallet pools and simple sharing mechanisms offers an effective solution.

2.2 CLASSIFICATION

The proliferation of 'flexible manufacturing systems' calls for a more rigorous definition of a FMS.

The Flexible Manufacturing Systems Handbook [13] defines FMS as a "computer-controlled configuration of semi-independent work stations and a material handling system designed to efficiently manufacture more than one part number at low to medium volumes".

The FMS Report [14] uses a functional definition in "A process under control to produce varieties of components or products within its stated capability and to a predetermined schedule" and a technological one in "A technology which will help achieve leaner factories with better response times, lower unit costs and higher quality under an improved level of management and capital control".

Browne, et al [15] defines "A flexible manufacturing system is an integrated, computer-controlled complex of automated material handling devices and numerically controlled (NC) machine tools that can simultaneously process medium-sized volumes of a variety

of part types". This paper further identifies eight types of flexibility of FMS as machine; process; product; routing; volume; expansion; operation and production flexibility. FMS are then classified into four types according to their flexibility into:

Type I : Flexible Machining Cell - consists of one general-purpose CNC machine tool, interfaced with automated material handling which provides raw castings or semi-finished parts from an input buffer for machining, loads and unloads the machine tool, and transports the finished workpiece to an output buffer for eventual removal to its next destination;

Type II : Flexible Machining System - can have real-time, on-line control of part production. It should allow several routes for parts, with small volume production of each, and consists of FMCs of different types of general-purpose, metal removing machine tools;

Type III : Flexible Transfer Line - which for all part types, each operation is assigned to, and performed on, only one machine. This results in a fixed route for each part through the system. The layout is process-driven and hence ordered; and

Type IV : Flexible Transfer Multi-Line - consists of Type III FMSs that are interconnected.

A similar definition and classification is adopted by Warnecke and Steinhilper [16].

A more recent United Nations study [17] definition reflected the advances in the second generation systems and defined : "A flexible manufacturing system is an integrated computer-controlled complex of numerically controlled machine tools, automated material and tool-handling devices and automated measuring and testing equipment that, with a minimum of manual intervention and short change-over time, can process any product belonging to certain specified families of products within its stated capability and to a predetermined schedule". This study re-named a Type I FMS as a flexible manufacturing unit; introduced a DNC linked system of flexible manufacturing units as flexible manufacturing cell and identified a Type II FMS as a flexible manufacturing system.

2.3 ROLES OF DESIGN AND CONTROL

Elements of FMS hardware are organised to exploit the flexibility and controllability of automated equipment and reduce the effects of the corresponding constraints. This is represented by the FMS design problem and the operations control problem. The design problem selects the production part mix, machine selection and layout design

that addresses the aggregate balance of the system to meet production demand. The operations control problem handles the deployment of system equipment to satisfy the immediate production requirement.

The design and control aspects play different roles in fulfilling production requirement, their relative functions in different production types can be shown by cases within Great Britain. Applying traditional production management concepts, manufacturing systems can be organised into the flow, job and batch type systems. These organisations of material flow, with the associated information flow characteristics for production control, are based on the particular production requirement.

A typical flow type FMS is the Rover M16 engine line at Longbridge [18,19][Fig.5] for machining cylinder heads and cam carriers. The machining section consists of nine machining centres and three head indexers to perform the main machining before the parts are assembled and finally bored together with special assembly cells and finish machining cell. The system has tremendous process and part flexibility since material transfer is by AGVs, and with identical machines. Though the machines are tooled to perform specific sets of operations, there are redundancies to allow for routing changes and they can be re-tooled relatively easily. However, Rover decided to use a flexible transfer line for the production of the next engine series, thus the efficiency traded for flexibility is not always acceptable for volume production. The operations characteristics of this type of system are that the use of system flexibility can be planned for. The steady production runs allow system decisions for normal production be performed off-line, the real time scheduling problem becomes relatively simple. Performance can be designed into the system.

A job type system operates on single job entities at a time. In the context of this discussion, a job type FMS organisation is one that tooling, including tools and fixtures, are specific to the particular job. Machines have to be retooled for other jobs and the operation of one job is totally independent from others. A job can be a single entity for operation as in the Anderson Strathclyde FMS [20][Fig.6] or a batch of parts so grouped that they are operated as one entity, as in the British Aerospace FMS in Preston [21,22][Fig.7]. The system can be designed to cope with the spectrum of parts, but the effective utilisation of system depends on good workflow control in operations. During a visit to the Anderson Strathclyde system, the operation was run with the control computer off-line because the designed scheduling system did not reflect the manufacturing problem realistically.

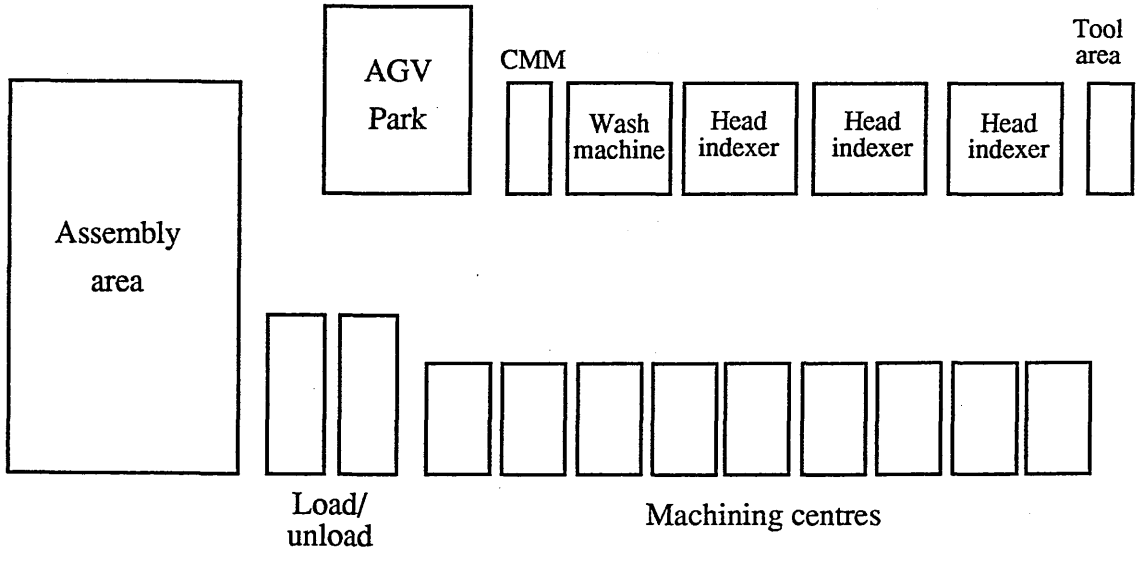


Fig 5. Longbridge FMS layout

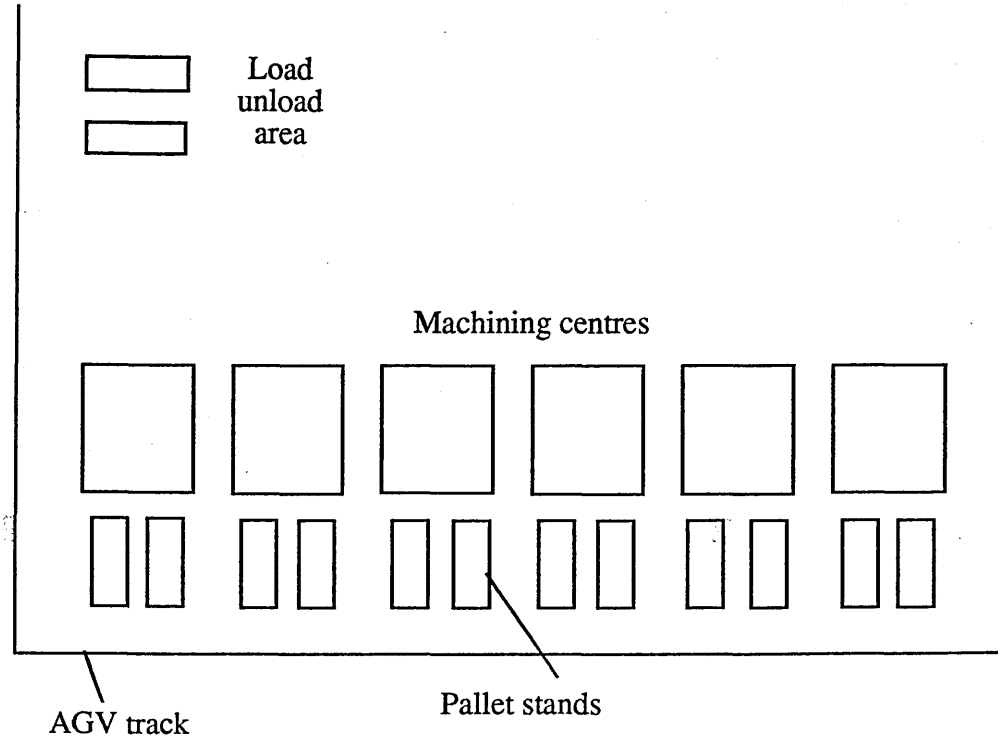


Fig 6. Anderson Strathclyde FMS layout

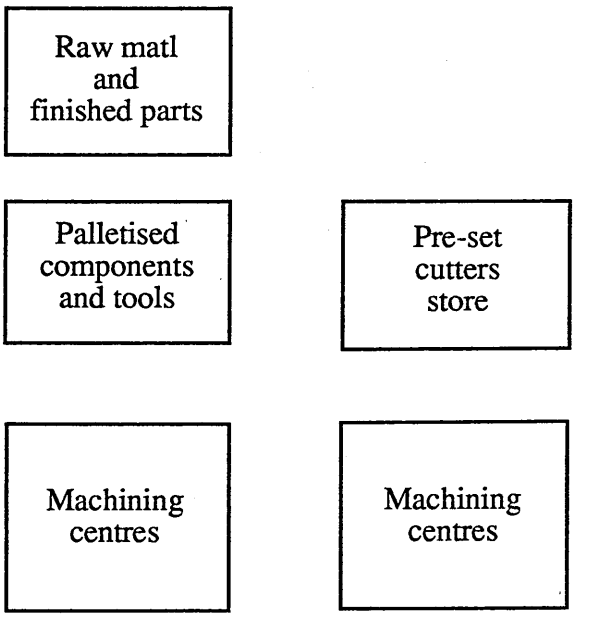


Fig 7. BAe Preston FMS layout

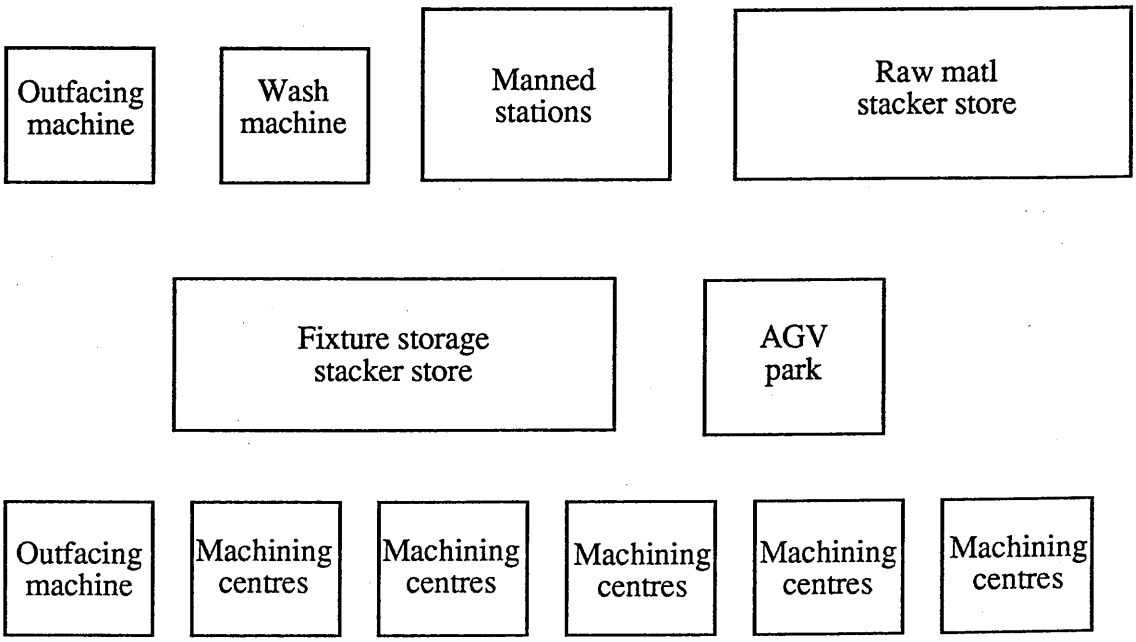


Fig 8. Hattersley Newman Hender FMS layout

Batch type systems simultaneously produce batches of different parts. Parts of a batch enter the system when fixtures are available; tools on machines are used to machine parts from different batches. The inventory of the system consists of parts waiting to be processed and finished parts waiting for the whole batch to be completed in addition to the parts actually inside the system. Though one of the aims of flexible manufacture is to produce batches of one, current production realities dictate otherwise. A very good example of batch systems is the valve FMS at Hattersley Newman Hender, Omskirk [23,24][Fig.8]. This kind of system is most complex to control as there is a mix of dedicated and common tools, a limited number of pallet positions and a mix of batches having different due times and priorities.

The production organisation of the FMSs is not determined by the hardware in the system. The layout and machine selection of the examples are similar across the cases. The distinguishing characteristics are the way the systems are used to meet their respective production requirement. The original FMS design problem that address the machine selection and layout problems has a reducing solution space as equipment suppliers standardise the line of products they support and it is difficult to justify the extra cost for specialised equipment. Though optimisation of layout to production requirement is possible, flexibility is best maintained by using general purpose standard machines that interface with material handling system. The above cited examples all employ 'standard' modules but operate in very different manners. The function of a flexible manufacturing system depends more on the control logic that drives it than the hardware components.

2.4 OPERATION CONTROL OF FLEXIBLE MANUFACTURE

2.4.1 Operation objectives

2.4.1.1 Manufacturing

The effectiveness of any control system depends on the control parameters and objectives. Manufacturing objectives include minimum cost, best quality, shortest leadtime, minimum over-due, maximum equipment utilisation and minimum inventory. Some of these are complementary to one another while some are conflicting. For single product manufacture, high equipment utilisation is directly translated to high production. In multi-product production, conflicts in priorities between parts and overall manufacturing objective are certain; the different parts are competing for the same available proc-

essing equipment. Good utilisation of equipment may require large batch sizes which have an adverse effect on delivery lead time and inventory. Classical production research reduces manufacturing problems to simple objectives which neglects the fineness of balance in real world decision making. Multiple criteria optimisation form an objective function that combines multiple goals in a linear function, however the setting of the goal coefficients are difficult and the time dynamic nature of the relative importance of decision criteria is not considered.

2.4.1.2 Immediate

At a manufacturing cell control level, overall manufacturing objectives are broken down into more immediate system objectives. These system objectives drive the detail job despatch and scheduling control decisions. To achieve the system objective, appropriate decisions have to be made using information on the actual system composition, its behaviour, and current status. The interpretation of this information is affected by the local shop environment. A system producing to high equipment utilisation using 'shortest processing time' rule neglects some long processing time parts when a continuous stream of short processing time parts is flowing into the system. A system that employs a 'look-ahead' rule to pursue production to due date will command machines to wait for impending critical jobs and cause losses in equipment utilisation, with the consequence of forcing other jobs critical. The despatch decision in a shop producing unrelated parts will be different from one producing complete sets of parts for further assembly. In a well coordinated manually controlled shop, these conflicts in goals are resolved with local intelligence in day-to-day production shop management. In an automated production environment, the intelligence for effective operation has to be built into the control system.

2.4.1.3 Dynamic

The decomposition and identification of immediate system objectives is not a static problem. The immediate goal that guides decision making to achieve system objectives changes with time. Consider an automated shop that requires manned loading of parts and runs unmanned shifts whilst the system completes all the parts that are loaded into the system. The system objective is maximum machine utilisation, translated to the immediate goal of minimal idle waiting time of machines. The 'shortest processing time first' despatch rule was found by simulation research [25] to meet this goal. This rule will contradict the objective of the system during the transit to an unmanned shift

— it does not load the unmanned system with sufficient work. The overall system objective will be achieved by a recognition of the the impending change and the adoption of the immediate goal to load as much work to the system before going unmanned. This requires a timely change of despatch rule. The change of immediate goal would not be necessary if the shop does not rely on manned loading or the shop is manned at all times. A different approach will be needed if the requirement is to clear all parts from the system and shut down while still maintaining maximum equipment utilisation.

Conventional scheduling control methodologies do not consider the time dependent characteristics of objectives; numerical optimisation programs are computation intensive and hence expensive [26] in repeated scheduling. Moreover, the extensive study of a particular problem only provides a single solution method. The effectiveness of these methods depends on the current validity of the original assumptions. In complex systems, the solution method is proven to be effective over the long run. However, local situations can develop that temporarily invalidate the methodology. The basis of goal identification is a thorough understanding of the factors that operate in the particular installation at particular times.

2.4.2 Control factors

2.4.2.1 Batching

Production of parts can be in batches or in a complete kit. Batching is the production of a number of the same part. This can also be the result of setup requirements in downstream operations; the economic use of material handling equipment; or the production to store requirements. Batch production allows easy grouping of tools. The processing time and inventory of a batch is the control criteria rather than that of individual parts in the batch.

In the manufacturing of high capital value products, it is often more economic to produce a set of parts to build a single product. There machine setup cost in FMS is low and the whole set of parts kept together reduces the production control overhead. This requires the kitting of parts.

The grouping nature of part production in the form of batches or kits is an important scheduling consideration in the actual production environment that is neglected in scheduling research.

2.4.2.2 Fixtures

The fixed number of buffer stations in a FMS limits the total number of pallets in the system. The number of pallets for each part type is a balance of individual part production rate within this constraint. A further constraint is the number of fixtures available for the part, this number is usually limited as the cost of high precision fixtures is high. Fluid phase fixture [27] promises flexibility in the number of available fixtures, but the technology is not fully proven.

A fixture load of parts on pallet is the basic unit of work in FMS. Short machining time pallets place a high demand on the in-FMS material handling system and reduces the spindle utilisation of machines. For small components with short machining time, it is normal to mount several fixtures on a pallet. In 4 axis machines, cubes are used so that more components can be handled and machined together.

The grouping of components depends on whether batching or kitting is used. In batching, all fixtures in the pallet are for the same component, machining of the whole pallet requires the same set of tools. This eases the tool and production control problem. In kitting, the number of tools to machine the pallet load is usually minimised by exercises to group components using similar tools. However, to allow for circumstances where only some of the parts are needed, the production control logic as well as the NC programs have to be much more complicated.

Another grouping method is progressive fixture. This is used when the production volume of a part is not very high. Fixtures for the different setups of the same part are mounted in the same pallet. The part to be machined will progress through the various setups to completion. Different workflow control logic is needed to handle the different concepts.

2.4.2.3 Tooling

Except for a few special processes, the tools with which a workstation can be equipped at any time is limited. This limit places an uppermost constraint on the number of different processes that the workstation can perform without tool changes. The availability of pre-set tooling facility and central tool store are essential to the operation of FMS. The hardware means to re-supply tools are identified in a matrix form.

	manual change	automatic change
machine running	xxx	xxx
machine stopped	xxx	xxx

The two factors have implications in the machine scheduling and manning policies. The tools can be changed in ones by specially designed robots or in a batch.

Tool availability policy is influenced by the parts produced. For production of parts that share a lot of common tooling, then all machines can be identically tooled and replacement only on usage and breakage. This can be easily planned and controlled. Tool rationalisation and planning [28,29] reduces the requirement for different kinds of tools. The case when all tools are changed when a new part is to be cut is also trivial. The general case of parts sharing common tools but the total number of tools exceed machine magazine capacity is usually true. Tool planning is a crucial constraint in scheduling for FMS.

2.4.2.4 Manning

Human loading and unloading is common is most small FMSs. The system could be manned on 3 shifts and thus continuous operation could be scheduled. The system could be manned in 2 shifts so scheduling has to plan long run time jobs for the unmanned shift to utilise machine time as much as possible. Or the system could be run without unmanned operation then system shutdown has to be planned for everyday, synchronising the end of all jobs near to shutdown time. The same considerations apply to tool preparation as well. Tools have to be planned for unmanned periods. Manning regulates the phases of production in system operation and workflow control logic have to plan for these phase changes.

2.4.2.5 Production changes

Manufacturing systems are built to fulfil production requirements. However, these requirements can change due to changing market scenes and invalidate original estimations. The flexibility and the ability of the system to respond rapidly to changes is a crucial asset rather than a complication of control.

Production requirements changes can be identified in two levels. At a global level, the product for manufacturing may change. This would require changes in process plans, tooling, fixtures and perhaps the way the system is run. Changes in local form may be the changes in production quantity, priority of current production period.

2.4.2.6 Breakdowns

In a highly integrated automated system, any element breaking down affects all other parts of the system. This may be in the form of hardware or software breakdown of related equipment, or consequential effects carried by parts, say a part sent to a tapping station without completing drilling due to a broken drill. A failed workstation also stops or reduces the supply of parts for downstream workstations and may bring the system to a halt.

A manned system can adapt to these random changes easily. An automated system requires all these adaptive logic be programmed into its control or allow for human intervention to handle exceptions.

2.5 CONTROL SYSTEMS IMPLEMENTATION

The implementation of FMS control system is in the various computers of the system.

At individual machine level, control is performed by the machine tool computer numerical control and programmable logic control computers. These computers handle the calculation of tool path and cutting, the necessary sequencing of mechanisms for automatic tool and pallet changes, and probing. They are special purpose industrial computers for machine control. With the advent of more powerful microprocessors, many of these controllers can handle a significant amount of general purpose computing, allowing complex functions like tool monitoring to be built into the machine tools. These computers communicate with the outside world for integration into complete systems. It is the responsibility of machine tool builders to incorporate the necessary functions for the machines to be integrated in a complete system.

The material handling systems are controlled by various means depending on the complexity and requirement of the particular system. In the simplest level, the system can be controlled using the programmable logic controller of one of the machine tools it is serving. Robots have their own controllers which can be linked to the machine tool

logic outputs. In a more complex system or when the processing power of the local elements are not sufficient, the material handling system can be controlled by its dedicated programmable logic controller or minicomputer. With more processing power, the material handling system can be made more intelligent and perform some optimisation functions in its operations rather than just serving the calls of the machine tools.

There is no universal definition of system or cell control functions and different systems suppliers have different approaches. The functions range from the very basic sequencing of transport system to total control of system operations scheduling, control of transport machines and tool management, integrating with plant level production control system. Specialised cell controllers are available in the form of high performance programmable logic controllers, but general purpose computers, especially the PDP and VAX line of minicomputers are very popular. It is also common to have a dual control computer system so that a control computer failure will not disable the whole system.

In a comprehensive system, capacity planning is usually done off-line performing a capacity planning function. This is in the from of a Decision Support function running on a computer with a link to the company MRP system or manual input of data. Interactive systems allowing human operators to change inputs according to trial scheduling results are common. Some of these systems also run on the same system control computer. The KAPLAN system [30] is an example. In simple flexible manufacturing systems, this function may be left to the system supervisor or operator to be done without any specific aids.

Real time job sequencing and despatch function is usually done by the system control computer as a work flow control function. Various heuristics, are used for making the different job control decisions. The possible complexity of control decision logic depends on the processing power of the available computer. Some priority schemes are usually used to allow more urgent jobs to enjoy higher flow rate.

The overall FMS control implementation technique is in the distribution of logic processing. Very fast real time machine tool control is performed by specialised controllers and the more abstract level of production control done with general purpose computers that can be programmed to perform more complicated processing. This distribution of intelligence also means that the hardware specific controls are bound to the machines and can be used as a module for other systems. A hierarchy of processing power integrated with the communication network provides an architecture for flexible design

with minimum expense. Although the computer hardware for control implementation is fairly well defined, the software for control functions does not have an equally established pattern.

2.6 GENERIC CONTROL SYSTEM

There is a growing market trend for small flexible manufacturing systems in the form of machining cells built up of standard components in a step-by-step manner [31,32,33,34,35,36]. They are simple systems consisting of standard machines linked by material handling systems as rail guided vehicles or automatic guided vehicles. Automatic tool replenishment systems are increasingly used in these systems. Being relatively standard systems, they can be supplied and supported easily by machine tool builders and are affordable by users. The benefits of high machine utilisation with automatic work transfer and great product flexibility make them attractive alternatives to unlinked CNC machines. The high utilisation and reliability of systems are reported by Hammer [37]. They do not feature very much in the press but are very evident in machine tool exhibitions, 6EMO, Mach88.

A major economic factor against the use of simple modular FMSs is the high cost of control software. FMS control software had been consistently costly, constituting 15-40% of total system cost [38,39]. Greenwood [40][Fig.9] discussed the significance of 'generic' FMS control systems as a factor likely to influence the development of FMS. Control software is the highest risk component of FMS and the most likely source of problems in commissioning. The importance of a 'generic' FMS control system lies in reducing the cost of one-off development of control software. This allows easier access to flexible manufacture technology.

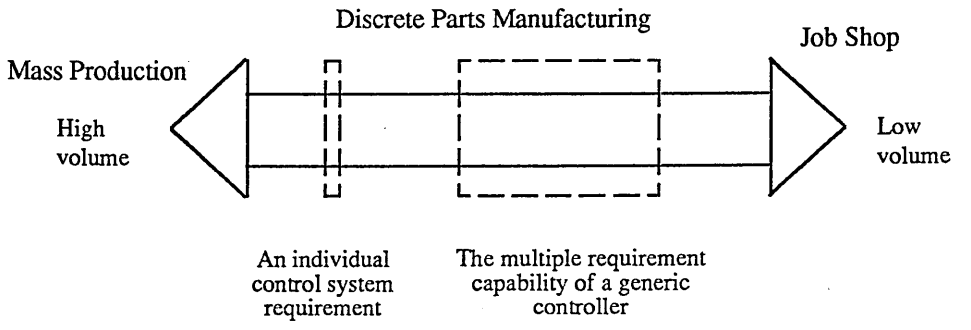


Fig 9. Generic controllers within the manufacturing spectrum[Greenwood,40]

The basic communications and reporting functions of control software can be standardised with the use of modular systems and implemented in low cost control computers. FMS control problem has always been seen at the machine sequencing and communications level, as evident Hudson and Webb [41]. Generic controllers developed to meet this low level automation requirement [42] have no provisions for complex work scheduling for effective production.

However, each application has its characteristics in product range and volume, overall manufacturing structure, company tradition, manning policy and other characteristics. The workflow control and scheduling function has to identify these variations and adapt to these different production environments dynamically to reap the benefits of flexible manufacturing systems. Visits to existing FMS implementations confirmed Greenwood's observation [42] that work scheduling relies heavily on human decisions. The incorporation of site dependent knowledge into system control computers is the key to the next application of next generation flexible manufacturing systems. A generic model and structure to represent the use of decision knowledge is crucial to this development.

CHAPTER 3
SCHEDULING CONTROL RESEARCH REVIEW

3.1 JOB SHOP SCHEDULING

Scheduling problems formed an integral part of the original applications of operations research and management science. The notion of better deployment of resources that result in greater rewards from the same amount of equipment is very attractive. Scheduling is effectively the manipulation of blocks of time to achieve or approach the desired objectives.

King and Spachis [43] divided scheduling problems by their data features into :

- data variability — 'deterministic' if all the data involved are deterministic, otherwise 'stochastic'; and
- data time dependability — 'static' if none of the initial data changes over time, 'dynamic' otherwise.

Conway, Maxwell and Miller [44] classified scheduling problems to their job arrival process, number of machines in the shop, flow pattern in the shop and decision criteria. Of special significance are the flow pattern and the decision criteria.

Flow pattern defines the nature of the shop control problem. The commonly adopted flow pattern representation are:

G general job-shop — where every job may have a different routing through the machines;

F flow-shop — where every job has the same routing through the machines; and

R randomly routed job shop — where job routing is random.

King and Spachis [43] added the type P, permutation where the same job sequence applies on all the machines.

The decision criteria defines the objective of control. Common objectives are:

increase production output;
reduce queuing times;
reduce stocks of finished goods or raw materials;
reduce work in progress;
reduce back orders;

reduce delivery periods and product lead times;
reduce facilities' idle time;
reduce overtime;
reduce delivery delays....

Conway [44] defined a set of regular measures of performance that were values to be minimized that could be expressed as a function of the job completion times. This set included completion times, flow times, lateness and tardiness. Many of these measures are related or equivalent. Conway mathematically proved the relationship between some of these measures.

Optimal finite scheduling for single machine static problems were essentially solved. The problem of two and three machines minimum makespan was solved by Johnson [45] and Jackson [46]. Beyond these, there are no definite solutions to the more general problem. It is agreed that the general n-machines m-jobs scheduling problem has a solution space of $(n!)^m$ and belongs to the class of problems NP-complete [47,48]. That is to say the problem has an answer, but there are no efficient algorithms to find the solution. The computation effort to find the solution increases exponentially with the size of the problem.

King and Spachis [43] categorised solution approaches to the optimal solution of static and deterministic scheduling problem into:

combinatorial switching and restrained enumeration;
branch and bound tree search;
dynamic programming; and
integer programming.

Pervious research using these techniques to solve particular problems are documented [43]. There are limitations to their applicability in real industrial problems with the static and deterministic assumptions on data.

Attempts to ease the solution of scheduling problems include:

exact solutions to relaxed problems;
incomplete search; and
ad hoc decision rules.

The last refers to heuristic decision rules that are applied at the time of decision and will

not be revoked. Computer simulation is the main tool for study of heuristics. Gere [49], Conway, Maxwell and Miller [44], Panwalker and Iskander [50], King and Spachis [43], Graves [51], Blackstone, Phillips and Hogg [52], Kiran and Smith [53], traced the development of this technique. These rules usually rank the choices according to some criteria. These criteria are based upon the processing time, the due date, random effects or a combination of these. An interesting observation is that the "shortest processing time first" dispatch rules and its derivatives consistently outperform others in different kinds of study.

Production shops are also studied by queuing theory or network analysis. Jobs are modelled as customers that visit the machines (servers). Jackson [54] pioneered these studies with an open network model of queues with exponential processing times and Poisson job arrival distribution assumptions to investigate 'first come first serve' and random dispatching rules. This was expanded by Gordon and Newell [55] to closed network models, where the completion of a job is immediately replaced, maintaining a constant number of jobs in the system. Steudel, Pandit and Wu [56] used time series analysis of inventory data to form a queuing network of a job shop with good results. Buzacott and Shanthikumar [57,58] furthered the work in analysing job waiting times and built different models of job shops. Based on the probabilistic distribution of job arrival rates and processing times, queuing network models are useful in analysing aggregate levels of processors, jobs, queues and utilisation in steady state. When the system operations do not deviate too far from queuing model assumptions, this is a good method to gain insights into system characteristics. However, queuing theory models are not applicable for making detail control decisions.

Theoretical work in scheduling has not seen widespread applications in industry. Panwalkar, Dudek and Smith [59] reported small model size and simple decision objectives as part of the causes. King[60] reinforced these arguments and further elaborate on the simplification assumptions.

Newman [61] discussed the difficulties in applying traditional research in real life scheduling and suggested the use of knowledge based systems. Factory scheduling requires predictive planning capability and the flexibility to react intelligently. Mathematical research models are sufficient for planning but too rigid to exploit the flexibility inherent in the shopfloor.

3.2 ARTIFICIAL INTELLIGENCE

Artificial Intelligence research started in 1956 when John McCarthy and other scientists coined this term in Dartmouth College. Development of artificial intelligence was long and treacherous, a very good summary is in Hunt [62]. Details of AI are available in Barr, Feigenbaum and Cohen [63] and Winston [64]. Successes in AI applications emerged in the 1970s. In 1980, Japan launched the $1.5 billion 10-year Fifth Generation Computer Project, threatening to dominate worldwide Information Technology with AI and especially knowledge based systems [65]. Various national AI projects were launched in response, raising awareness of AI and attracting commercialisation of AI products. AI encompasses many fields and subjects [Fig.10], those related to scheduling are briefly discussed.

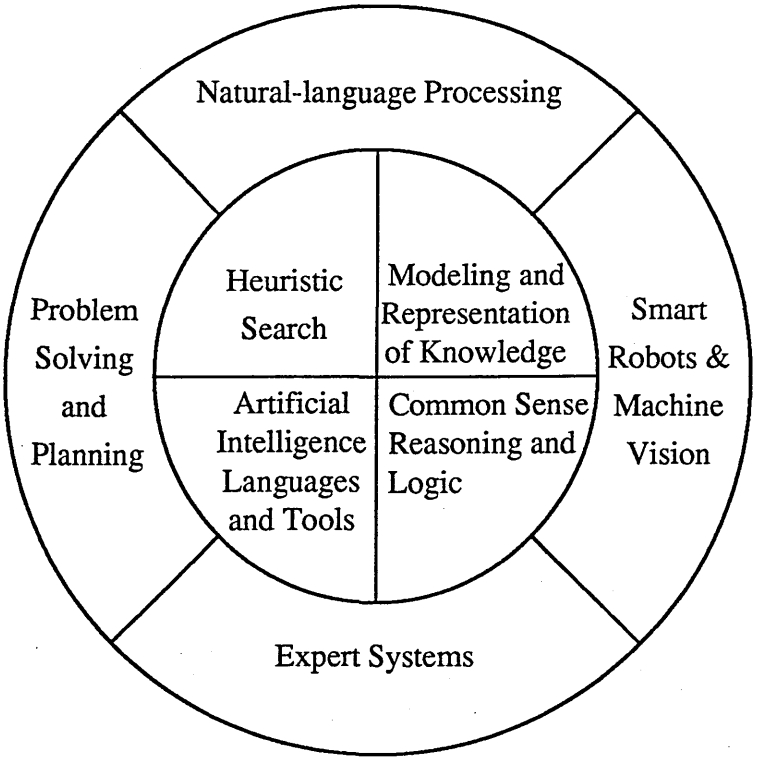


Fig 10. Elements of Artificial Intelligence[Hunt,62]

The early failures of machine translation programs showed the significance of context relevant knowledge for intelligent solutions. This led to the development of knowledge based systems which separate the conventional computer programs in to a knowledge base and a control (also called inference engine) function [Fig.11]. The importance of

the use of knowledge led Nilsson [66] to argue that "artificial intelligence is primarily concerned with propositional languages for knowledge representation and with techniques for manipulating these representations".

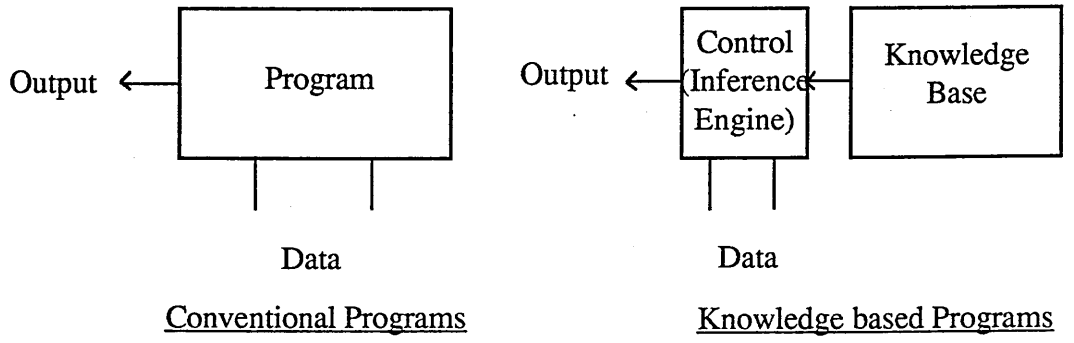


Fig 11. Knowledge based vs Conventional Programs

The representation of knowledge was in semantic networks [67] for the parsing of natural language sentences. Semantic networks are directed graphs, consisting of nodes and labelled edges, used to describe the properties and relations of objects, events, concepts, situations and actions. Minsky [68] developed frames to represent the stereotyped situations. A frame has slots for objects and relations that would be appropriate to the situation. Frames can also include procedural as well as declarative information. The use of default values for slots and the concept of instances and inheritance extend this representation to handle object oriented programming concepts. SRL is one frame based knowledge representation developed by Fox [69] and is the tool for AI development in many projects in Carnegie-Mellon University. Another model for procedural representation is 'production system' [70]. The knowledge of procedures to perform specific tasks are encapsulated in the form of 'if-then' rules. The production system activation mechanism matches the system state to the pre-conditions needed for the procedures to be invoked and control the firing of procedures. Because of their modular representation of knowledge and their easy expansion and modifiability, rule based representation is one of the most popular artificial intelligence knowledge representations and widely used in expert systems [71].

Problem solving methods were devised to use this knowledge. Most problems can be represented as solution trees in which solutions are being searched. Compared with traditional operations research approaches, search algorithms in artificial intelligence employ problem specific knowledge to guide the search more effectively; optimisation

is not mandatory and good feasible solutions are usually accepted. General Problem Solver by Ernst and Newell [72] attempted problem reduction by means-end analysis. State changes are defined by a set of operators, the initial and goal states are compared and operators selected to minimise the difference between the two. The same idea is seen in dynamic programming and breadth first tree search.

Theorem proving attacks the problem of common sense and logic, knowledge and facts are asserted as logic statements and automatic theorem provers are used to deduce the truth from these statements. The main mechanism in theorem proving is resolution which generates the negations of assertions and proves by the contradiction of these negations. A general purpose question and answer system QA3 demonstrated this approach in the domains of robot movements, puzzles and chemistry. Simple propositional logic matures into predicate logic whereby assertions can be made with variables.

Fikes and Nilsson [73] combines the use of theorem proving and search algorithm in STRIPS which performs robot planning. Theorem proving resolution is used to ascertain the truth of facts used by the search, which is done using means-end analysis. This division of problem uses the two techniques to their best advantage and solves more complex problems.

Computation needs in artificial intelligence require tools that represent concepts and facts better than numerical representation of data. John McCarthy developed LISP around 1960 as a list processing language with recursion capability. It is a functional programming language with the interpreter evaluating functions in the form of symbolic expressions. LISP is widely used in AI research in USA and there are special machines to execute LISP. In the course of natural language processing work, Colmerauer and Roussel developed Prolog - PROgramming in LOGic in 1973 [74]. Prolog is a theorem proving system employing Robinson's resolution mechanism on first-order predicates in Horn clause form. Programs are represented as clauses with a single left hand side term and a string of right hand side terms. The clause can be seen as a statement of a goal (left hand side term) and its constituent sub-goals. The depth first search algorithm of Prolog attempts to prove the goal by sequentially matching the right hand side sub-goals to established facts in the program. Backtracking of match to previous search point is done when the attempted matching fails. Prolog allows a problem to be represented by a set of unstructured declarations and simultaneously retain the control of a sequential definition of procedures. Clocksin and Mellish [75] in University of Edinburgh were instrumental in developing efficient implementations and

the promotion of the use of Prolog as the artificial intelligence language in Europe. Prolog gained world prominence after being selected as the core language of the Japanese Fifth Generation Computer Project. Many practical implementations of Prolog as a general programming language are direct consequences of this. The power of Prolog for prototyping concepts and representation of knowledge is reported by Weber and Moodie [76] and evident in many current research reports.

Other programming tools and environments emerged from the commercialisation of artificial intelligence, most of them are developments from previous research work. The more important ones are ART from Inference Corporation, KEE from IntelliCorp, and Knowledgecraft from Carnegie Group. Mettrey [77] gives a good comparison of these systems. Stone [78] plotted the commercial trends in AI tools. The effectiveness of these tools depends on the application, the basic knowledge model driving these tools was developed for particular types of problem domains.

An important work applying artificial intelligence techniques on scheduling research is the ISIS by Fox [79,80]. Fox and others developed the knowledge-based job shop scheduling system ISIS for the Westinghouse Electric Corporation Turbine Component Plant. The knowledge representation power of their approach is more powerful than the numerical problem formulation of conventional operations research. The problem was formulated as a series of contraints to job scheduling, the constraints were classified into five levels and selectively relaxed to achieve an acceptable schedule. ISIS selects a job using a priority algorithm and then works out the earliest start and latest finish times of all operations and passes these as constraints to the detail scheduling level. Detail scheduling begins with a pre-search analysis to determine the scheduling direction and search operators. A beam search is then performed by the search operators to generate the states of the schedule. Further constraints may be generated by the search operators. Alternatives at each state are rated by the constraints and the best N proceed to completion as the final schedule. This is equivalent to a step by step build up of schedule evaluating each step in the process. The use of variations of constraint directed search to scheduling problems was also reported by Bensana [81], Elleby [82], and Sauva [83].

Artificial intelligence computing concepts and tools comes in a time when conventional design of control systems is limited by its pure mathematical approaches to problems. Buchanan [84] presented a survey of working systems and literature current to 1986.

3.3 HIERARCHY AND STRUCTURE OF FMS DECISION MODELS

Different models were used to study flexible manufacturing systems. The structure of models is governed by the problem being studied and directly affects the results of the study.

The approach used in Purdue University by Nof, Barash and Solberg [85] for the study of first generation FMS are applicable to many parts machining systems. The emphasis was on the control of the complex item flow created by the versatility of the automated workstations. Operational control of flexible manufacturing systems were broken down into:

part-mix problem — to balance the production of the system to make the best utilisation. This was further refined to the part-type selection problem and the part-mix ratio problem;

process selection problem — to select amongst alternate processes; and

part flow problem — to sequence the part flow in the system, refined to the initial entry of parts into an empty system, general entry of parts into a loaded system and the allocation of parts to machines within the system.

Stecke [86] continued this work and refined the problems into grouping and loading problems at the aggregate and detail levels. CAN-Q type close queueing network analysis is suggested for aggregate level solution and non-linear mixed integer programs for detail solution. The models for the FMS control problems employ mathematical models that do not represent non-quantitative parameters in decisions.

Suri and Whitney [87] performed a detail analysis of the decision making requirement in flexible manufacture. Based on their experience with the "FLEXPLAN" decision support system for a FMS at Hughes Aircraft Company in El Segundo, they identified the three levels of operation organisation. The first level consists of long-term decisions with time horizons in terms of months and years. This involves establishing policies, production goals, economic goals, and making decisions that have long term effects like part-mix changes and system modification and expansion. The suggested decision tools are part selection program, queuing models and simulation. The second level involves medium term decisions with time horizons of days or weeks. Typical tasks are dividing production into batches, maximizing machine utilisation and responding to disturbances in production plan-material availability. Decision tools suggested are batching and balancing programs and simulation. These decisions are typically made by the FMS line manager. The third level involves short term decisions with

horizons of minutes and hours. They include work order scheduling and dispatching, tool management and reaction to system failure. These decisions are made by the FMS control computers unless in exceptions, when the FMS line supervisor will take over. This is a commonly accepted structure of decisions in FMS.

These decisions are interdependent. Decisions are structured to hierarchies to ensure their interactions are defined and complementary. The Automated Manufacturing Research Facility(AMRF) of the US National Bureau of Standards is a testbed in FMS control concepts. Jones and Mclean [88] reported the five level AMRF hierarchical control structure of facility, shop, cell, workstation and equipment. The facility level corresponds with the long term level in the Suri and Whitney model. The shop level corresponds to the second level decisions. Apart from batching using group technology, this level also controls the activation and de-activation of 'virtual manufacturing cells'. These are logical groupings of manufacturing resources to work on particular families of parts, and are formed and disbanded dynamically to meet production requirement. This corresponds to the third level decisions. The lowest levels of workstations and equipment are automated machine units that carry out commanded tasks. O'Grady, Bao and Lee [89] compared this hierarchical structure with the Advanced Factory Management Systems structure of Computer Aided Manufacturing International Inc. and considered the issues in intelligent cell control. The Advanced Factory Management Systems structure consists of the factory control system level, the job shop level, the work centre level and the unit/resource level. It does not have a cell level, the Suri and Whitney third level decisions are performed in the job shop level.

Apart from these hierarchical structures of control, Duffie, et al [90] proposed a non-hierarchical structure based on networks. The proposed system handles part scheduling only, parts "make requests" for the next processing machines when they are finished at a station. A communication network connecting the machines broadcast these requests until a machine can receive the job. This simple logic allows a small software to be installed in each machine and dispenses with a central FMS supervisory computer. Lewis, Barash and Solberg [91] had suggested a similar data flow architecture. Distributed systems are more resilient to failures, but the need to coordinate and plan forward cannot be easily addressed.

Not all the model decisions are necessary. Flexible manufacturing systems organised for flow type manufacture do not need to consider process selection or alternate routing problems. Other systems may have configurations where batching is not necessary.

3.4 QUEUEING MODELS

The first mathematical model on flexible manufacturing system (called computerized manufacturing systems then) was reported by Solberg [92]. The FMS studied was the Sundstrand Omnicontrol system at the Caterpillar Tractor Company in Peoria, Illinois. The major problem was in the determination of the material handling system design, the number of pallets and transport carts to ease the congestions between machines. FMS was treated as a closed network of queues and the production rate, average production time and the various station utilisation were calculated. This model was based on the Jackson network of queues model with a static N jobs in the shop and involved several restrictive assumptions:

all stations with FCFS queue discipline, service distributions were exponential;
all jobs had the same arrival rate;
unlimited queuing space at each station, blocking never occurs; and
no machine breakdowns.

The solution of the network was coded in the CAN-Q package and served as the basis for several FMS research projects in Purdue University with Stecke. CAN-Q was used to study the machine loading and balancing problems for FMS. The problem of assigning jobs to different machines was studied [93]. It was found that the system production rate was best when the loading level of all machines were even. Pooling of machines into groups improved productivity because of the reduction of waiting times in a single server queue [94]. The optimum grouping of machines and their loading, however, should not be balanced; a suitable on-line control strategy gave better performance with an uneven grouping. These provided insights into the operations characteristics of FMS.

Suri and Hildebrant [95] developed the MVAQ system that incorporated multiple part classes functionality into the model and solved the network of queues problem in a more efficient manner. This allowed the modelling of systems producing a variety of parts simultaneously. Hildebrant [96] also studied the effect of machine breakdown to FMS scheduling.

Other modelling efforts were undertaken by Buzacott [97] and Shanthikumar [98], Yao [99,100] who developed generic models for different kinds of material handling systems and queuing discipline. His modelling efforts extended to open queuing networks to represent the limited buffer space of stations and the effect of blocking.

Analytical models are useful in providing a rough cut solution to FMS design problems. The production rate and utilisation of equipment and flow time of jobs can be determined easily. However, the assumptions involved are very restrictive and do not take into account the scope of performance improvement at the dispatch level. The single steady-state objective of solution is also contrary to normal production dynamics. The main advantage of this solution technique is the limited effort required for the results. They are useful in predictive applications on gross system characteristics rather than in the detail prescriptive control of a system.

3.5 SIMULATION AND HEURISTICS

Flexible manufacturing systems can be seen as an automated production shop, the application of job shop scheduling techniques to the study of FMS includes the popular heuristics and simulation. FMS is very amenable to simulation as most of the data in FMS are deterministic, the processing time of a CNC machine is usually fixed by the program.

Early simulation works were developed into FMS simulators for different kinds of FMS. They included GCMS from Purdue [101] and MAST from CMS Research [102]. Simulation became an integral part in all FMS planning and design [103,104]. Graphical animation in simulation was used by ElMaraghy [105]. Major simulation work was done by Rathmill [106] for the SCAMP FMS and Carrie [107,108] for the Anderson Strathclyde system. All major FMS builders and consultants use simulation to validate design and justify system layout. Simulation systems with graphical animation is the adopted method as it eases the debugging and validation of the model, it also serves to present the system to management and educate operations personnel. Commercial simulation packages like SEE-WHY, OPTIC, SLAM II, SIMSCRIPT, HOCUS, etc are used as well as systems developed from raw programming languages. Detail simulation is a powerful tool to understand the operations of any system. However, the progressive refinements of decision variables can take many simulation iterations. Suri and Cao [109] developed perturbation analysis that greatly reduce the efforts in gradient climbing. Traditional simulation tools are supplanted by new tools based on artificial intelligence techniques, like SIMKIT developed from KEE and Simulation Craft from KnowledgeCraft. Specific techniques in natural knowledge front ends, expert system model generators [110] and debuggers are appended to current simulation tools.

The application of simulation to FMS was mainly in the study of the best hardware

configuration and layout of particular systems [111]. Stecke and Solberg [112] used simulation to overcome the limitations of analytical techniques and studied the loading and control problems of the Caterpillar FMS. This classic paper defined 'loading' as the assignment of operations to machines and done prior to the real time despatching of work. A control scheme was used to define the sequence of decision making. The decision points were time when the cart could make a movement. Idle machines were selected according to their workload and feasible waiting jobs were then selected according to the dispatch rules. Five loading strategies and sixteen priority dispatching rules were used to investigate the effect of these combinations. The research provided important insights into factors, interactions, problems and control of FMS.

Chan and Pak [113] developed a model of a FMS controller that can be used to evaluate different alternative route control, material transport facility control, machine failure control and workpiece loading control strategies. The model was used to develop heuristics for scheduling dynamically loaded FMSs [114]. Their 'alternate operation' heuristic evaluated the selection of jobs according to job slack and whether the selected job will force other jobs critical. An 'alternate operation + look back' heuristic used look ahead to check if a critical job would come to the selected machine before the selected job could finish. The major emphasis in this work was in the completion of jobs within due dates.

Most of these studies used traditional scheduling rules in job shop scheduling, a review of which was published by Choi and Malstrom [115]. They developed a physical simulator using model hardware to perform simulation [116]. Model components and microcomputers were used to physically model a FMS. The model was very realistic, encompassing all the details of a real system. This approach, however, is expensive to modify and expand.

Weston et al. [117] reports the FMS emulation system of Loughborough University of Technology. Multiple microprocessors are used in a system that represents the different stations of the FMS. This is less expensive than the physical model simulation and more flexible.

Because of the details involved in simulation, the actual decision decomposition process of the model is also reflected. The scheduling decisions in FMS are commonly separated into the part selection problem, the machine selection problem and the routing selection problem. The decomposition of the problem reflects the characteristics of the

problem studied.

The production data set used in simulation affects the results of the model. Part routings and processing times interact with the scheduling mechanics of the model. Nelder [118] studied the interactions between the skewness of operations times, routing flexibility, loading of machines and scheduling rules. The interactions are statistical significant, though the exact causal relationships were not quantified.

3.6 KNOWLEDGE BASED SCHEDULING

Knowledge based approaches differ in the scope of problem. Some researchers use planning models to attack the static deterministic scheduling problem while others apply the knowledge based approach to a specific scheduling sub-problem.

An early overview of artificial intelligence in manufacturing was by Bullers, Nof and Whinston [119]; they used predicate logic and theorem proving to represent aspects of manufacturing control, specifically assignment of jobs to machines. Young and Rossi [120] gives a more recent comprehensive look at the application of knowledge based control of flexible manufacturing systems.

Shaw and Whinston [121] adopted a plan generation system based on the STRIPS with the inclusion of the time concept. Job schedules were represented as plans and partial plans in the data base, together with the goals and the world model of current situations. The knowledge rules, the operators that describe the transformation process and the inference rules were located in the knowledge base. The inference engine directed the plan generation process. A linearly sequenced plan was generated for each task, these plans were then check for conflicts in resource requirement. The precedence relationship between conflicting plans was then resolved dynamically by a procedure that commit the least resources as possible. Jobs that could not be scheduled with the primary resources would then be scheduled with alternate resources. Dynamic scheduling could be achieved by using the current world model and schedule in new jobs. This system was written in LISP and runs on a VAX 11/780 computer. Shaw [122] reported the computational performance of the Non-linear Planning Algorithm system with more powerful heuristics implemented with Common Lisp on a Texas Instrument Explorer computer.

Subramanyam and Askin [123] applied the production system representation of rules.

Prolog was used to build an expert system to study a flexible manufacturing system that consists of two head indexers and six machining centres linked by a tow cart system. Five part types with predetermined routing were scheduled. The "part selection from the input queue at a machine" was addressed. The decision was to select a job selection rule that corresponds to the system states(heavily, moderately or lightly loaded), machine states(overloaded, moderately loaded or underloaded) and job states(critically late, moderately late or normal). Different rules were defined using utilisation and queue length to determine loading. The important criteria at the state was deduced from the various states. This is an interesting system that conforms to the decision hierarchy of conventional systems but uses a much more powerful analysis technique to make the necessary decisions. The states of system, machines and jobs are requested from the user through a dialogue and the expert system recommends a rule after the dialogue.

Kusiak [124] presented a two phase scheduling algorithm that schedules jobs according to a sequence of priority rules; these included job type, number of parts in product, number of successors and processing time. The readiness of machine, fixture, pallet, tools, tool magazine and material handling equipment were then checked. If any equipment in the checklist was not ready, the corresponding alternative equipment decision table was consulted for alternative action. The job sequence was then generated according to the appropriate operations precedence and equipment availability. This algorithm takes into comprehensive considerations all the equipment requirements of FMS.

A knowledge based routing system was developed by Ben-Arieh [125] using C-PROLOG on a VAX 11/780. The system has two levels of knowledge. The first level decides the immediate solution to the routing problem, the second level observes the behaviour of the system and identifies favourable groups of rules. The level one system consists of the dynamic database that contains the real-time information which varies with time (queue size, machine states); the static database which is time-independent (product structure, processing times); the behavioural knowledge that contains the rules describing the behaviour of the system, priority in queues and database query system; the algorithmic knowledge that calculates the various alternatives and evaluates them; and a simulation driver that contains the event file and simulation clock. The manufacturing system studied was composed of five cells which could perform all operations but at different efficiencies. The products were 14 parts, nine of which were used to assemble the final product. The objective was to maximise the production of assembled parts. The scheduling algorithm used complicated heuristics to breakdown assembly

structure and work out the due time of all components. Dispatching of jobs in queue used these due times and attach high penalty to lateness. Heuristics were used to estimate the component completion time for this decision. This algorithm outperformed static priority rules for the specific parts and assembly production and compares favourably with a interactive human scheduler.

Alexander [126] suggested a framework of an expert system for the selection of scheduling rules, a portion of which was implemented using development tool MI on an IBM PC. Unspecified rules were used for the selection of rules according to the scheduling objectives. For multi-criteria scheduling, individual rules were selected for the different objectives and then combined into the final rule using weights. The resultant rule was then fed to simulation and fine tuned to the unique requirement of a particular shop.

Steffen [127] presented other work in artificial intelligence based scheduling systems.

3.7 INTELLIGENT CONTROL SYSTEMS

Apart from specific algorithms, different authors have proposed control systems models for intelligent FMS control.

Wu and Wysk [128] reported the Multi-Pass Expert Control System(MPECS). MPECS included :

an expert system to generate potential scheduling alternatives based on real-time shop information and scheduling knowledge;

a simulation model to allow the system to evaluate alternative schedules based on the system performance;

a decision structure that will update performance rules based on "simulation-system" experience; and

a mechanism to affect the control on a variety of flexible machining cells.

Upon receiving an job order, the expert system generates alternative dispatching rules and scheduling heuristics for the simulation model to evaluate. The best rule is than employed for execution. The performance was better than that of the single scheduling rule and heuristics that switch scheduling rules without reasoning based on dynamic change in the system. The use of simulation for evaluation is time consuming and the effectiveness is affected by the simulation time window.

Cross and Ravi Kumar [129] proposed a very comprehensive model for Intelligent Control System. The Input Prediction Module forecasts the value of all relevant variables. The System Analysis Module provides a detailed description of the system state trajectory, an initial system state and a specific set of policy parameters. The Policy Nomination Model then determines which policy alternatives are to be studied within a decision interval. The Policy Selection Model selects a specific set of policy parameter values to be used within the next planning horizon. This complicated model contains very detail mechanisms to decide on the current control policy and incorporate the role of control system design into the control system architecture. There were no details on the implementation results reported.

The Production Logistics and Timings Organizer(PLATO-Z) of O'Grady and Lee [130] interprets shop level orders and control equipment in cells. It employs a hybrid blackboard and actor framework. The knowledge source used by the blackboards are selected dynamically to reduce solution space and approaches the actor model in the extreme case. Four blackboards were used to handle respectively the scheduling, operations dispatching, monitoring and error handling functions. Each blackboard has a number of knowledge sources that guides decision making, the knowledge is represented in the forms of production rules, flavours and procedures. The blackboards communicate with one another to maintain the current status and reschedule correspondingly. The system was implemented in Lisp on a Symbolics 3645 computer.

The MEDEMA (MAnufacturing DEcision MAking) decision framework by Chryssolouris [131] treats the assignment of production resources to production tasks as a multiple-criteria decision-making problem. MEDEMA uses an eXperimental Determination of ALternatives(X-DAL) module to generate alternative task-resource alternatives; the X-DETA(eXperimental DETetmination of Attributes) module determines the decision criteria; the X-EVA(eXperimental EValuation of Alternatives) module evaluates the alternatives according to the decision criteria. Different criteria can be dynamically selected and used to evaluate the alternatives. The correspondence of MADEMA decision rules and the conventional Shortest Processing Time rule under certain decision situations is very well discussed.

The Production Activity Cycle(PAC) model by Browne [132] aims to control cell level systems with a small time horizon. Incoming orders are interpreted by a scheduler to form the production schedule, rule based techniques are being used to perform scheduling. The schedule is executed by the dispatcher. A monitor keeps track of physical pro-

duction and sends feedbacks to the scheduler and dispatcher for update of schedule if necessary.

All the models recognise the dynamic nature of system control and employ varying criteria and rules in decision making. Feedback is a feature in some of the systems. All the systems assume the need for detail schedule generation at a cell production level and then use simulation or dispatch and monitor to adjust the schedule as production progresses. Given the short time horizon and the dynamic nature of cell production, a detail schedule that needs constant revision may not be an efficient way of decision making. A model that concentrates on the decision making for cell control may shed some light on the control needs of small systems.

CHAPTER 4
PILOT RESEARCH MODEL

4.1 OBJECTIVES

The pilot model developed by the author is the investigatory vehicle for the research into intelligent decision making in flexible manufacturing systems control. Intelligent control is attempted by merging the wealth of classical scheduling research with installation particular knowledge using artificial intelligence techniques. The approach is to establish a framework for FMS control decisions that delineates individual decisions and their relationship. This allows decision solutions to be problem specific within an overall framework that is based on solid research.

The objectives of the pilot model are :

1) form a simulation environment for the development of a more advanced control system;
2) test the suitability of Prolog as a language to develop the control system;
3) gain experience in building a FMS control system; and
4) evaluate the power of conventional heuristics in the control of FMS.

4.2 SIMULATION FOR REAL TIME CONTROL

Simulation is the tool to perform 'what-if' studies, which is becoming a major part of decision support. Most of the work done in simulation has studied the effect of a particular decision evaluated to a certain objective. In real time FMS scheduling, decisions have to be made continuously under changing situations, it is important to be able to see the effects of each decision. The close inter-relationship between the various decisions means the effects of decisions are in two levels. A real level when the actions caused by the decision have actually taken place and a planned level when the decisions made affect the course of other related decision.

To study the decisions in a real time control system, the model must have sufficient detail to depict all the relevant factors for decision making. Moreover, these data must be available in 'real time' so that the decision making process can be monitored. It is beneficial to have dynamic colour graphics display of the current state of the system [133], to show the 'real' effect of the scheduling decisions. The effects on other decisions can be interrogated from the queues of planned actions and detail states tables.

The graphics display forms a picture of the system that focuses the attention of the user much better than text tables and print-outs. When the user detects events or sequences of activities of interest, he can suspend the simulation and interrogate for details of different states and plans. Facilities to save the state of the whole simulation environment and to restore the environment to these 'snapshots' are essential for a step-by-step development process.

If the user interface is sufficiently realistic, users of the system can learn from the decisions made and improve upon them. This forms a testbed for developing heuristics and strategies for control. Current commercial simulation systems are designed for study of aggregate parameters and do not have the flexibility for the variation of decision making and detail data. As the theme of this exercise is to study the control of the system, the decision rules and knowledge are of paramount interest. The new artificial intelligence techniques in knowledge representation provide a promising tool in the study of decision. The programming language Prolog was selected to build a simulation environment for the study of the FMS control problem.

4.3 KNOWLEDGE BASED MODEL

Conventional Flexible Manufacturing Systems suffer from an inflexible control structure. The high level work flow control problem is usually neglected. There is a requirement for software systems to control small modular FMSs in different applications. These control systems must grow with the machinery hardware and utilise the additional resources effectively. The work flow control of these systems must be flexible enough to suit the production environment of different installations.

The knowledge based system model is a very suitable framework for representing intelligent control systems. A basic knowledge based model [Fig.12] consists of the knowledge base, the database and the inference engine. The knowledge base contains the knowledge needed for decision making in the system. The database holds data to be manipulated. It is usually refined to a static database which holds static data defining the system and a dynamic database which acts as the scratchpad for the system to hold intermediate data. The inference engine contains the logic that drives the system using the rules from the knowledge base and data from the database. The major characteristic of knowledge based systems is the separation of a knowledge base from the control. This highlights the significance of problem specific knowledge. For the solution of similar problems, the knowledge base for solving particular problems can be replaced

with another, using the same inference engine. This is the concept of expert system shells.

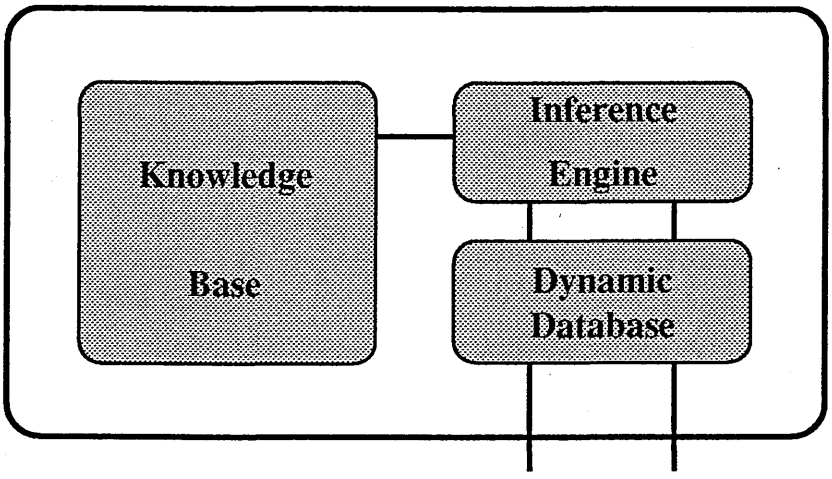


Fig 12. Knowledge based model

On a system analysis aspect, knowledge based models highlight the importance of decision making and raise the problem oriented decision making above the detail mechanics of computer programming. System design can concentrate on how to make better decisions rather than being bogged down in detailed coding. The high level knowledge representation schemes developed for knowledge based systems also facilitate the representation of the problems involved.

In mapping the knowledge based model to the requirements of intelligent FMS control, the low level communication and machine control functions are well developed and can be standardised as the common core of all control systems. The driving core of the control system forms the inference engine; the details of these low level controls are hidden from the work flow control system. This core governs the running of the system, identifies the decision points and then makes use of the rules in the knowledge base to perform decisions. The knowledge base is different for different flexible manufacturing systems. Each is tuned to the particular production requirement. It is envisaged that the major part of the knowledge base will be based on job shop control research and is common, but the particular requirement of individual systems can be incorporated with minimum difficulties. This universal control model [Fig.13] is applicable for all flexible manufacturing systems. This is effectively an expert system shell for decision making in flexible manufacturing systems; it controls machines rather than providing diagnosis or other peripheral functions.

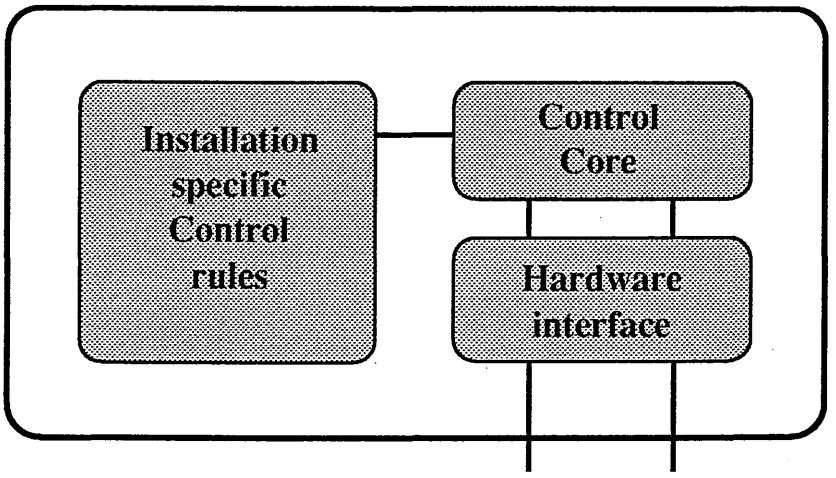


Fig 13. Universal FMS control model

In this study, the control system is detached from the machinery hardware it controls. The hardware model is replaced by the simulation environment which emulates the low level control function and response of the real world system.

4.4 MODEL OF MODULAR SMALL FMS CELL

4.4.1 FMS model hardware

The flexible manufacturing system hardware configuration selected for study is a small flexible manufacturing system that can be built in a 'step-by-step' manner [Fig.1]. It consists of four identical machining centres linked by a rail guided vehicle. All machining centres have automatic tool change capability and tool magazines of reasonable size. Each machining centre has a pallet buffer to hold pallets waiting for machining or finished work to be transferred to the next stage. The system has fifteen pallet buffers which is available to buffer all machines in the system. The rail cart travels along a simple to-and-fro line. The system is controlled by a central host computer. The work envelope of the machining centres are in the 500mm - 750mm cube range and tool magazines hold 60 - 80 tools. This class of small FMS under consideration is particularly suitable for small/medium machine shops and subcontractors.

4.4.2 System component characteristics

4.4.2.1 Workstations

The workstations are CNC machines and load/unload stations. The CNC machines are equipped with automatic tool change facilities and a single buffer station. The operation of the machine is under full CNC control. The CNC programs are either stored in machine control memory or can be loaded through DNC with no time loss. Under normal operations, the system control computer will inform the machine of the pallet number and NC program number for the next job to process. The machine then looks for the pallet on the machine table and the buffer station, moves the pallet into the machine table and starts operations, executing the CNC program. Tool life usage is monitored and deducted from the tool file. Upon normal completion, the machine notifies the system control computer and repeats the cycle. If no work is available, the machine will swap the worked pallet to the buffer and wait for a new job.

The load/unload stations do not have a buffer and tools needed are hand tools that are always available.

The status of the workstations are monitored by the system control computer, faults will be flagged and the operations in process is aborted when faults are detected.

4.4.2.2 Buffers

The buffers in the system serve as a common bank to hold pallets. They possess no intelligence and can only reflect their states. They can be declared fault or down for maintenance.

4.4.2.3 Material handling system

The rail guided vehicle used has two pallet stations, thus it can collect a pallet and deliver another to a workstation or buffer at the same time. This mode of operation does not use the vehicle to carry two pallets at the same time, the two stations are for transfers only. This simplifies the control logic.

The transport system is controlled by a computer under the direction of the system control computer.

4.4.2.4 System control computer

The system control computer is a general purpose computer that communicates with all

system components and updates the system status in various databases. It performs all the necessary tool life management and reporting function. The scheduling control of the workstations and the material handling system can use all the available current status information in the system. The link with the higher level production control system provides the production targets as well as indicates the expected loading of the workstations and the pallets.

4.4.3 System assumptions

The characteristics of the flexible manufacturing system studied are more rigorously defined by the following system assumptions:

Physical Assumptions :

- No machine may process more than one operation at a time. Obvious due to physical impossibility.

- No pallet-fixture may be processed by more than one machine at a time. Obvious due to physical impossibility.

System hardware related assumptions :

- All machines in system are identical with identical characteristics and performance.

- There is a local buffer at each station.

- CNC programs include pallet change and tool change commands. They are proved and correct.

- A finite process time is assumed which includes set-up time --- with NC programs and automatic transfer mechanism, this assumption is valid. Provision for variable processing time due to torque monitoring/adaptive feed control can be incorporated later.

- The time intervals for processing are independent of the order in which the operations are performed — valid as above.

- Simple start/stop's are all that is needed to drive the machines.

- Transport times are deterministic from fixed stop position to fixed stop position.

- Pallet change time between cart and buffer is fixed.

- Sets of fixtures for parts are prepared and ready. All fixtures enter and leave the system through the load/unload station.

- The time for fixture introduction and removal from the system is fixed and known.

Production environment assumptions :

- MRP type system work targets are available — this gives a pool of orders to schedule.

- Due dates are known with variance allowed.

- Workplans exist for all parts to be machined. These detail the resource requirements in machining the parts.

- Raw material arrives in bins/batches. Arrival times can be random or in a certain pattern.

- Tool material to make up tools is assumed to be always available. This could be ensured by MRP analysis in advance.

- Machine breakdowns will occur in a pre-determined schedule as well as randomly. This allows for preventive maintenance and sudden breakdowns.

- Job routing is flexible, specified only by the pallet/fixture group compatibility between component fixture pallet and machine and machines' working envelope. Actual job possibility depends on the tooling of the machine and job operation requirements.

4.5 CHAMELEON - A PROLOG FMS ENVIRONMENT

4.5.1 Prolog simulation

Traditional numerical programming languages used in simulation are prescriptive rather than descriptive. They specify the sequence of processing in a procedural manner and

their data structure does not represent objects and symbols well. In simulation, the model is usually built up with a definition of the objects involved and their characteristics as to what will happen if some other events happens, mainly in the form of multiple 'if-then-else' definitions. The sequence of events are non-deterministic and program execution depends on the dynamics of the interaction of the model elements. A 'production system' concept consisting of 'if-then' rules is more suitable for simulation. Numerical languages are also weak in data representation and representing knowledge in which the newer generation symbolic and declarative languages excel. There is an increasing trend to use non-procedural language for simulation work [134].

The main characteristic of Prolog is its declarative format. All clauses (or statements or predicates) in Prolog declare some facts or rules, relationships between facts. Prolog is also a goal searching theorem prover. Given the facts and the rules, it seeks to verify the goal set. Prolog can be used to model a simulation in an expressive manner as the state of a machine for example is represented by

```
'state(machine,idle)'.
```

The condition for starting a transport function is :

```
condition(event(_,_,cart,Cart_x,[deliver,start],[[Dest,Dest_x],Fix_no,_,_],_)):-
    !,
    wks_state([cart,Cart_x],idle),
    wks_operational([Dest,Dest_x]),
    fixture_dock([Dock_type,Dock_no],Fix_no),
    wks_operational([Dock_type,Dock_no]).
```

This is translated as : the condition for cart Cart_x to start delivery of fixture Fix_no to destination of type Dest_type and number Dest_x is true or satisfied if Cart_x is idle and destination is operational and the location which Fix_no now stands is also operational. Prolog takes character strings as natural data structure and they are used to give the best expression of the model. Early implementations of Prolog could not handle numerical computation easily, but more recent versions provide good floating point mathematics and graphics capabilities.

Computer codes ultimately become electronic on-off signals in a processor. Prolog pays for its expressiveness in processing speed since more codes have to be interpreted to do the equivalent amount of work. Prolog and other numerical programming languages are different means of performing the same function. Prolog has the edge in communicating with the modeller while numerical languages hold their own when

numerical computation or very long simulated system run times are needed. The rapid advance in computer technology providing cheaper and faster computing power emphasise the need to save programming effort. The proportion of computing system cost continues to shift to the software side, the labour intensive modelling and programming efforts are the real critical part of simulation cost.

Futo [135] developed T-Prolog with a simulation extension to Prolog as a simulation language, derivative TS-Prolog [136] was developed for continuous simulation. PROSS in based upon an implementation of GPSS within Prolog, O'Keefe [137] demonstrated its use in a model of a barber's shop. To achieve interactive simulation of the real time control that gives the freedom of access to low level functions in prototyping, a special environment is written in Prolog from scratch.

4.5.2 Chameleon simulator

4.5.2.1 Chameleon implementation

Pilot Chameleon [138] was written in Enhanced Arity/Prolog Interpreter Version 4.0. A number of predicates had been added to the commercially available Arity/Prolog Interpreter to facilitate simulator writing. Certain graphics routines were written in C and compiled into the interpreter for faster response. The hardware used was an IBM PC/AT personal computer with 640K memory and 20Mb hard disk storage. The system runs with PCDOS or MSDOS Version 3.00 or higher. The computer was set up with the display conforming to ANSI Standard X3.64 for colour graphics display. For performance enhancement, one Megabyte of memory was used as virtual disk.

The system was upgraded to run with Arity/Prolog Version 5.0 which supports much better memory management. An IBM PS/2 Model 80-111 was used to run the system. The overall system performance was improved by four times with this workstation class machine.

Prolog programs are text files so it is straightforward to port the system to other Prolog environments. The few predicates special to Arity can be written in the destination environment with minimal customising workload.

4.5.2.2 User interactive control

A goal of this work is to investigate short term system responses to FMS control, the model is designed to perform event by event tracing and interactive control. The manufacturing environment is graphically represented on the screen and colour used to represent the states of all components. These are driven by the model as simulation proceeds. The user can interrupt the system any time during the simulation run or work in a step by step mode. Upon interruption, the model will display a menu of user functions [Fig.14]. This suite of functions allow the user to view all the states of the system, the queues and the decisions made, as well as the commanding action of the system. The user can create actions at any time by specifying the desired events and the times the events occur. This mode allows the user to drive the system, making all the logic decisions rather than using the logic modules pre-built in the system. If these 'canned' functions cannot satisfy the user, he can go to the Prolog interpreter level and

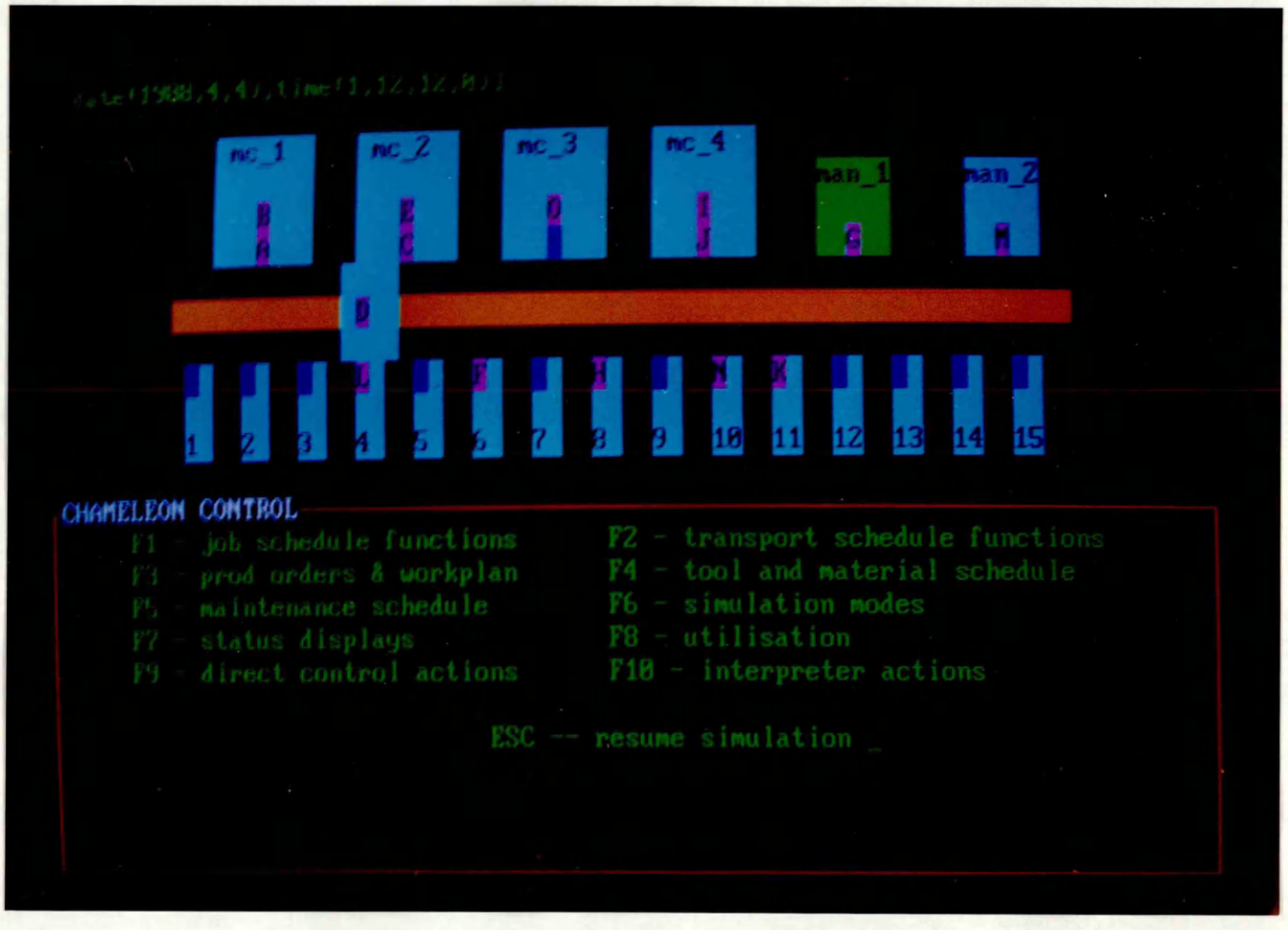


Fig 14. Chameleon screen - user interaction

issue Prolog commands to effect control. This of course requires knowledge of Prolog and the model. Frequently used functions can then be added to the suite of canned functions. This illustrates the incremental nature of Prolog.

All the display modes can be selectively switched on and off any time during simulation runs. To allow the user to experiment with the development of decisions, whole simulation states can be saved and restored.

4.5.2.3 Log and statistics

A log of all the events executed is kept on external files for post-run analysis. Event logs are structured so they can be interpreted by computer, an expert system written in Prolog or by the user [Fig.15]. This log consists of records of all events executed. In each event log, the first line consists of the time of action, they are the system time code, the system date and time and the actual real world time. They allow the study of system progress and the actual execution performance of the simulation system. The event executed is listed next. The last two terms in the log, global stack and garbage collection are parameters of the Arity Prolog language and used for control of system development.

Utilisation and other statistics can also be kept. These are formatted to be acceptable by common spreadsheet programs for analysis and graph plotting. The utilisation log [Fig.16a] presents the average utilisation of system resources in column form referencing the variable statistics collection time base. Time series analysis or other detailed utilisation analysis can be generated with any common spreadsheet programs. A sample analysis of the variations of utilisation is preformed and the trend plotted [Fig.17].

The production progress report [Fig.16b] records the completion of each order and allows the relative part flow rate for each order be checked. The format used is similar to utilisation logs for easy analysis.

All the statistics and logs can be selectively switched on and off any time during simulation runs.

```
$Log file to determine stack usage with unblock inoperation 9 Oct$
81772126.00000002
date(1987,10,9)
time(7,26,18,0)
time(15,17,7,5)
event(date(1987,10,9),time(7,26,18,0),user,user,interface,[],81772126.00000002)
global stack -size(19076,39868.0)
garbage collection -invoked(0,0,0)

81772137.00000002
date(1987,10,9)
time(7,26,51,0)
time(15,17,9,13)
event(date(1987,10,9),time(7,26,51,0),cart,1,[deliver,finish],
[[buffer,10],'100011',before,81772126.00000002],81772137.00000002)
global stack -size(30752,39868.0)
garbage collection -invoked(0,0,0)

81772162.00000002
date(1987,10,9)
time(7,28,6,0)
time(15,17,12,4)
event(date(1987,10,9),time(7,28,6,0),man,2,[load,finish],
['100001','PO_1',0,wp_3_0001,1,10,1],81772162.00000002)
global stack -size(30500,39868.0)
garbage collection -invoked(0,0,0)

81772162.00000002
date(1987,10,9)
time(7,28,6,0)
time(15,17,18,36)
event(date(1987,10,9),time(7,28,6,0),cart,1,[deliver,start],
[[man,2],'100005',before,81772162.00000002],81772162.00000002)
global stack -size(28400,42780.0)
garbage collection -invoked(0,0,0)

81772173.00000002
date(1987,10,9)
time(7,28,39,0)
time(15,17,20,45)
event(date(1987,10,9),time(7,28,39,0),cart,1,[deliver,half],
[[man,2],'100005',before,81772162.00000002],81772173.00000002)
global stack -size(33060.0,42780.0)
garbage collection -invoked(0,0,0)

81772185.00000002
date(1987,10,9)
time(7,29,15,0)
time(15,17,22,31)
event(date(1987,10,9),time(7,29,15,0),cart,1,[deliver,finish],
[[man,2],'100005',before,81772162.00000002],81772185.00000002)
global stack -size(31140,42780.0)
garbage collection -invoked(0,0,0)
```

Fig 15. Chameleon event log

$Utili file start 19/1 pm$

'Mc_1'	'Mc_2'	'Mc_3'	'Mc_4'	'Man_1'	'Man_2'	'Cart'	'Time_base'
0.855	0.935	0.886	0.949	0.813	0.735	0.245	4800.0
0.988	0.988	0.992	0.983	0.896	0.908	0.285	4800.0
0.988	0.99	0.992	0.994	0.71	0.612	0.183	4800.0
0.985	0.99	0.99	0.99	0.868	0.828	0.257	4800.0
0.988	0.988	0.994	0.992	0.842	0.716	0.235	4800.0
0.985	0.99	0.99	0.992	0.802	0.722	0.237	4800.0
0.988	0.99	0.992	0.985	0.862	0.833	0.26	4800.0
0.985	0.99	0.992	0.998	0.652	0.729	0.192	4800.0
0.972	0.988	0.992	0.983	0.876	0.917	0.29	4800.0
0.985	0.99	0.99	0.996	0.72	0.601	0.202	4800.0
0.955	0.99	0.992	0.988	0.9	0.896	0.279	4800.0
0.988	0.988	0.992	0.992	0.719	0.836	0.218	4800.0
0.985	0.989	0.992	0.994	0.774	0.692	0.215	4800.0
0.988	0.99	0.992	0.985	0.871	0.863	0.269	4800.0
0.921	0.99	0.99	0.998	0.654	0.677	0.19	4800.0
0.985	0.988	0.992	0.985	0.863	0.871	0.28	4800.0
0.988	0.99	0.992	0.992	0.705	0.683	0.209	4800.0
0.985	0.99	0.99	0.992	0.857	0.859	0.266	4800.0
0.986	0.99	0.994	0.988	0.804	0.697	0.233	4800.0
0.989	0.988	0.99	0.996	0.791	0.752	0.225	4800.0
0.985	0.99	0.992	0.985	0.853	0.855	0.271	4800.0
0.985	0.99	0.992	0.998	0.685	0.654	0.183	4800.0
0.988	0.988	0.992	0.983	0.907	0.893	0.302	4800.0
0.985	0.99	0.99	0.994	0.787	0.69	0.211	4800.0
0.91	0.99	0.992	0.99	0.831	0.775	0.248	4800.0
0.988	0.99	0.992	0.992	0.801	0.73	0.235	4800.0
0.983	0.988	0.992	0.992	0.779	0.684	0.223	4800.0
0.763	0.99	0.992	0.985	0.877	0.783	0.259	4800.0

Fig 16(a). Sample utilisation reports

$Prod file started 19/1 pm$

'PO_1'	'PO_2'	'PO_3'	'PO_4'	'PO_5'	'PO_6'	'PO_7'	'PO_8'	'PO_9'	'Time_base'
4	3	2	0	0	0	0	0	1	4800.0
9	8	4	2	1	2	1	1	2	4800.0
14	13	6	2	2	2	1	2	3	4800.0
20	17	8	4	2	4	2	3	4	4800.0
25	22	10	5	3	5	3	4	5	4800.0
30	27	12	6	3	6	3	6	6	4800.0
36	31	14	8	4	8	4	6	7	4800.0
41	36	15	8	4	8	4	8	9	4800.0
46	41	18	10	5	10	5	8	10	4800.0
51	46	19	10	6	10	5	10	11	4800.0
57	51	22	12	6	12	6	10	12	4800.0
62	56	24	13	7	13	6	12	13	4800.0
68	61	25	14	7	14	6	14	14	4800.0
72	66	28	16	8	16	7	14	15	4800.0
78	70	30	16	8	16	7	16	16	4800.0
83	75	32	17	9	18	8	17	17	4800.0
89	80	33	17	10	19	8	18	18	4800.0
94	84	35	19	10	20	9	20	20	4800.0
100	89	37	20	11	21	10	20	21	4800.0
105	93	39	21	11	22	10	22	22	4800.0
110	99	41	23	12	24	11	22	23	4800.0
115	103	43	23	12	24	11	24	24	4800.0
121	108	46	25	13	26	12	24	25	4800.0
127	113	47	25	14	27	12	26	26	4800.0
132	117	50	27	14	28	13	27	27	4800.0
137	122	51	28	15	29	13	28	28	4800.0
142	126	53	29	15	30	14	30	29	4800.0
146	131	56	30	16	32	15	30	30	4800.0
152	136	58	31	16	32	15	32	31	4800.0
157	140	60	33	17	34	16	33	32	4800.0

Fig 16(b). Sample production progress reports

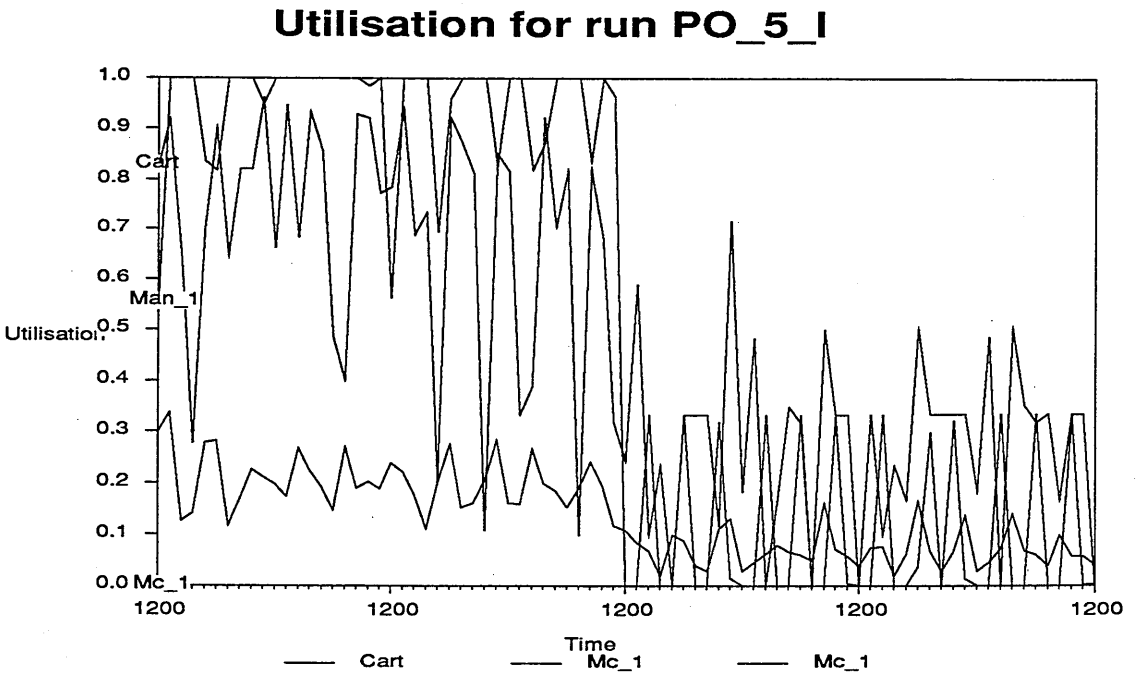


Fig 17. Analysis of utilisation log

4.5.2.4 Setting of simulation experiments

The three levels in the control of simulation in Chameleon correspond to the levels of control in operating a flexible manufacturing system. The top level is in the definition of parts production in the system. This defines all the parts that are to be produced, their routings and tooling requirements. The second level defines the fixture and tool assignment that defines the production configuration of the system for that specific time. The third level defines the actual despatch of the job. In Chameleon, the strategy in the third level is defined by loading the appropriate control modules into the system.

The top level production setting of the system is defined by the components the system makes, the components routing and the tools and fixtures available in the system. The simulation environment is designed to represent a small production with a wide spectrum of parts. A total of 22 component workplans are defined. All components go through the load/unload stations to enter and leave the system. The machining process routings are of three kinds:

a simple routing that requires operation on only one machining process;

a routing that requires two machining processes that have to be performed in a strict sequence; and

a two machining processes routing, the processes can be performed in any order.

Processing times for the load/unload operations are set at 10 to 15 minutes. Processing times for the machining operations range from 5 to 240 minutes, with the majority between 35 and 75 minutes. These operation times correspond to typical machining times of machining centres of the simulated kind. For two process routings, different skewness of operation times are used. There is a spread of operation times from front loaded to back loaded. Typical components without off system processes can be machined in two set-ups with four axis machining centres. Details of the work plans are in Appendix A.

The middle level assignment of processes to machines is determined by the machine tool assignments at production time. It is perfectly possible for tool assignments of a machine to perform the two operations in a routing, combining the two processes into a longer process. This emulation of the actual production process allows experiments with the loading problem, Stecke [112].

45 fixtures are defined for mounting the components in the system. Each process can use from one to three fixtures. The number of fixtures in the system is limited by possible pallet positions to 23, thus by controlling the fixture assignment in the system, different production ratios can be experimented.

The original system definition included 400 possible tools for the different processes. However, the power of the computer did not allow this degree of detail to be achieved. Tool setting is represented by the current ability of the machines to perform certain processes.

The production of the system is commanded by production orders to simulate the batch production environment. The order quantity can vary from one to infinity. This sets the production mix and requirement for a certain time. Together with the tool and fixture assignment, these three form the configuration of any experiment set.

Pallet and tool assignments, and production order quantity and priority can be changed during a simulation run if desired.

4.5.3 Prototype simulator structure

4.5.3.1 Kernel

The Chameleon simulator has a small kernel holding the concept of time based discrete event simulation [Fig.18]. The classic event driven logic for discrete event simulation is modified with queue polling to achieve a hybrid model that facilitates the integration of different functions in a simulation engine. Events are maintained in different queues which are polled for next event when the clock needs to be advanced. The model data, entities and events, are external to the kernel [Fig.19]. The kernel is essentially a server to advance the time and conditions of the model and regulate the occurrence of events and their associated changes in model states in the model. This kernel uses queues as channels of communication with any number of decision logic units in the model. These logic units manipulate the various events that can happen in the model. In using queues as a communication medium, the task of the decision (or control) logic is to maintain a queue of events for the simulator to execute. The kernel has the means to

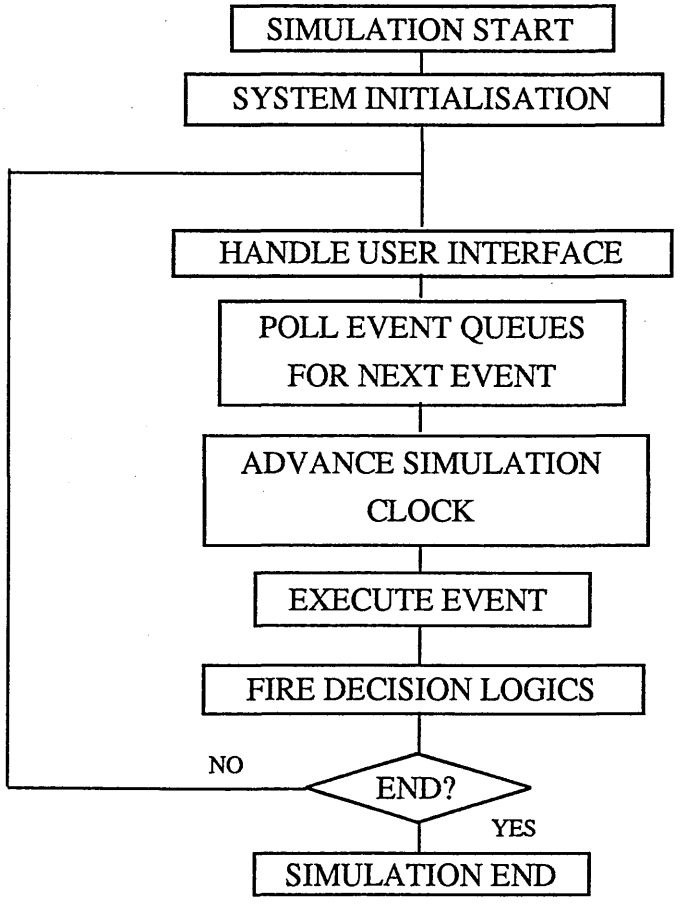


Fig 18. Chameleon simulation logic

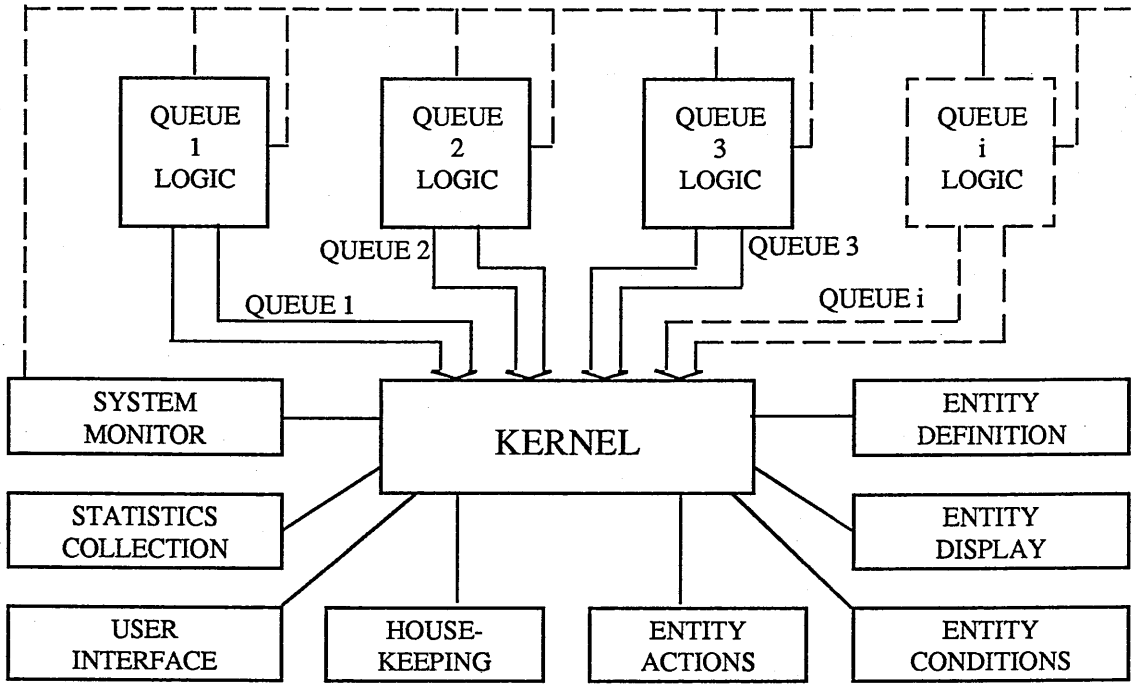


Fig 19. Pilot Chameleon structure

pass data and invoke logic to perform decisions at appropriate times. Regulation of the firing of decision logic economises on computing power and ensures model integrity. There are other housekeeping functions to attend to, such as report generation and event logs, these are also handled by the kernel.

4.5.3.2 Objects and entities

All entities(objects) are identified by their class and number. Their characteristics are defined by their attributes, part of which are their various possible states. Entity behaviour is defined by the events that change these various states and the corresponding event conditions and actions. The unified class and number identification allows common codes to be written to handle all entities in an uniform manner, achieving a near object oriented programming style.

This can be illustrated by examining the workstations; these are the machining centres, load/unload stations, transport cart and the buffer stores. They are the processing centres of the system and are represented by a four term structure :

```
wks( [workstation_class,workstation_number],
     workstation_description,
```

```
        number_of_tools,
        pallet_group      ).
```

The class and number uniquely identifies the workstation, the description is for reporting purposes, and the tools and pallet information are attributes for the control logic of the system to use.

4.5.3.3 Time representation

Time is fully represented in year, month, date, hour, minute and second. This full format is transformed to an internal time representation using a floating point number with each unit representing three seconds. The time base used in this implementation can handle 80 years before this floating point time representation overflows.

4.5.3.4 Events - queue and execution

All events in the simulator have a unified five element structure. They are expressed as a Prolog structure :-

```
event(      [date,time],
            [object class,object number],
            event name,
            parameter list,
            time key
      ).
```

The date, time and time key specifies the time point the event is scheduled to take place. The date and time representation are derived from the simulation time key purely to ease comprehension of results. The time key is used for sorting the event in the event queue and actual simulation. The object class and number defines the entity to which the event is related. Event name identifies the event. The parameter list holds the parameters for the event's action. Using a list, the number and structure of parameters are totally flexible.

Associated with each event are its conditions and actions. Conditions define the object and/or system states necessary for the event to take place. Actions are the consequences of the event taking place. This involves changes in object and/or system states resulting from the execution of this event.

A number of event queues are maintained. They are established according to the functions to which events are related. The present model has queues for the Job, Transport, Maintenance and Report functions. Although this increases the overhead in looking for the next event to execute, it allows a more logical grouping of events. Inspection of the queues provides useful information in analysing system execution [Fig.20]. The logic

Fig 20. Chameleon screen - event queues

control module built for each queue can manipulate the queue events to achieve real time re-planning.

The execution of events is shown in [Fig.21]. All the queues are polled by the kernel to find the next event that has its conditions satisfied. A pending events set is formed from the eligible events at the next firing time. The sequence of operations for execution ensures all activities are completed before new ones are started, achieving a three phase approach to simulation. Further tie-breaking is by a standing order of queues. It

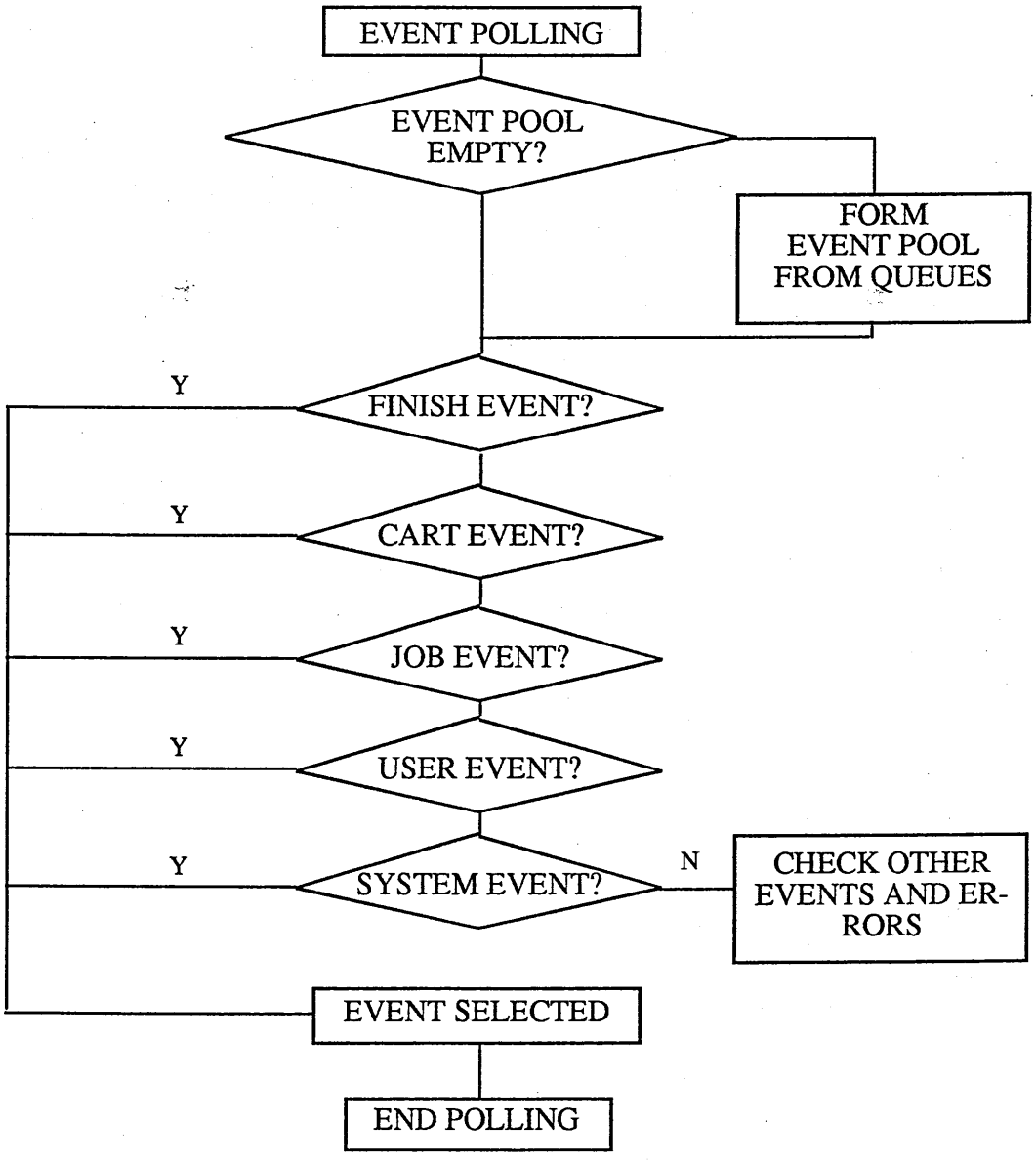


Fig 21. Event queue polling logic

is known that the tie-breaking mechanism affects the execution of the model. However, these effects are very model dependent and no general solutions may be found. More complicated tie-breaking mechanism could be built into the event checking routine if necessary. The simulation clock is advanced according to the next event time. The event is executed and reported. The necessary statistics and housekeeping work done and the system continues until the end of the simulation period.

The polling of event queues sorted in time order and checking the conditions for event ensures that events will be executed to schedule as soon as possible. The control logic

can diagnosis the cause of a scheduled event not being executed in time to achieve re-scheduling.

4.5.3.5 Decision logic

The pilot Chameleon models the different decision logic on a peer level, each logic module is independent and obtains information from other modules through firing other decision modules. This is similar to the blackboard approach used in HEARSAY-II [139].

Associated with each event queue is the decision logic module for the particular function. This decision logic generates and maintains events on the queues.

The logic can be fired by events that are executed. Event execution leaves messages in the system for the system monitor to interpret and invoke the appropriate logic to handle these requests. This extra step eliminates the possibility of making conflicting decisions. The system performs the conflict resolution function of artificial intelligence type production systems.

Logic can also be fired by other logic. This allows coordinated decision making in the system. This design allows one to functionally divide the whole control task into different aspects. The prime advantage of this approach is the incremental development of the model; building the system one function at a time and easy modification to tailor for different physical manufacturing systems.

Logic modules have their own communication queues. The firing party(ies) place messages in the communication queue of the fired logic and invoke it. The fired logic then examines the queue and generates, removes or re-schedules events in the event queues. They may fire other logic modules in the process to ensure their decision can be executed. This allows more sophisticated decision making than most other simulation systems, though at the expense of execution speed. As the processing at this point is situation dependent, it is not possible to predict computer execution speed and the simulation times are not constant. The modeller has the option to built more detailed logic and accept slower simulation run times.

The logic of the model is designed to be modular, and can be replaced with alternative modules. Five logic modules are implemented on the pilot model for job scheduling

using shortest processing time, longest processing time, longest remaining processing time, shortest remaining processing time and a pseudo-random logic. Two logic modules for transport scheduling are in use. They can be interchanged during simulation runs. Prolog has far greater representation power than these simple rules which will be exploited in later refinements.

4.6 RESULTS AND CONCLUSIONS

4.6.1 Production data set

For the pilot investigation, a production requirement of 50 units for components 0001 to 0009 was set. This selection has six single machining process components, three components with two sequential machining processes. A production of predominantly simple parts with a mix of processing times from very short(component 0006) to very long(component 0005) is used to investigate the discriminating power of classical scheduling rules. Fixtures 100001 to 100015 are assigned to pallets, a different number of pallets are assigned to components of similar complexity. The machines do not have duplicated tooling, but the load/unload stations are common for all load and unload processes. Details of this configuration is in Table 1, workplan information is in Appendix A.

Machine 4 is the bottleneck machine with nearly six times the work load of the lightest loaded Machine 1. It is loaded with both the longest and the shortest operation. Components 0001, 0002 and 0003 have near dedicated machines and multiple pallets. Component 0009 has only one pallet but processed in lightly loaded machines.

4.6.2 Results

A first set of experiments was done with the machines selecting a job from the set of available pallets in the buffer bank and that of the machines and load/unload stations. A request for transport is then sent to the cart which executes them in a first in first out order. This was repeated for five job selection rules:

scan first job available from buffer bank;
shortest next processing time first;
longest next processing time first;
shortest remaining processing time on fixture first; and
longest remaining processing time on fixture first.

TABLE 1

CONFIGURATION OF PILOT EXPERIMENT

Production orders	PO_1	PO_2	PO_3	PO_4	PO_5	PO_6	PO_7	PO_8	PO_9
Components	0001	0002	0003	0004	0005	0006	0007	0008	0009
Pallets	AB	CDE	F	GH	IJ	K	L	MN	O
Operation times (mins)	10	15	10	10	10	10	10	10	10
	40	45	75	20	240	10	45	45	20
	10	10	10	10	10	10	30	30	50
							10	10	10
Tooling assignment	MC_1	MC_2	MC_3	MC_4	MC_4	MC_4	MC_2 MC_4	MC_4 MC_3	MC_1 MC_3
Total operation time (min)	60	70	95	40	260	30	95	95	90
Minimum work time for each order (hr) = Total operation times / number of pallets assigned	25	19.4	39.6	33.3	108.3	25	79.2	39.6	75

Assigned machines work load(hr)

MC_1	50
MC_2	75
MC_3	129.2
MC_4	287.5

Starting pallet locations

Pallet	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Buffer startion	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Starting cart location : Buffer 9

A second set of experiments was conducted with a different decision strategy. The decision is initiated when the cart is free. The list of workstations from machine 1 to 4 and then load/unload station 1 and 2 and scanned in this order for a workstation that needs work. For this workstation, an available fixture is selected using the above five job selection rules. This fixture is immediately moved to the designated machine. Thus a machine selection problem is added to replace the transport request. The job selection decision is immediately executed and no decisions are made before they can be executed. This represents the situation when the cart is the bottleneck of the production process.

The production results of the experiments are presented in Table 2. The time from the start of the system when the first component for an order is completed, and the time when the last component is completed is tabulated. This represents the relative production rates the system control logic achieved for the particular production set.

Components 0001, 0002 and 0003 were completed soonest as they form essentially independent production sub-systems. The availability of the extra pallet in component 0002 did not confer significant advantage as there was only one processing machine. The better discriminating power in considering the total work content was demonstrated by components 0004, 0006 and 0005. The shortest and longest remaining processing time rules push their relative work load more effectively because the imminent processing time rules could not discern these operations when performing selection to the load/unload stations, which was a governing process in the routing. The decision timing when the cart is free performed better than that of workstations free, completing most of the job runs sooner. This is attributed to the making of decisions only when the cart is able to effect immediately. The small advantage margin indicated that the cart was not a consistent bottleneck.

Table 3 shows the times when the different workstations were first loaded with work and the times when the buffers on the machines began to be loaded with work waiting for processing. This reflects the ability of the control logic to utilise the immediate availability of resources and thus short term response. The values for filling buffer is not a very good indicator because the machines were finishing parts as the cart was delivering new ones, thus short processing time machines take longer to fill.

The shortest processing time rules performed better with a faster job turnaround. However, no rules consciously attempt to fill the machines up quickly and the result was

TABLE 2

PRODUCTION RESULTS FOR PILOT EXPERIMENT

This table summarises the elapsed time (hour) before:

first component is completed (upper value); and

all components completed (lower value).

	Order no.								
Despatch Rule	PO_1	PO_2	PO_3	PO_4	PO_5	PO_6	PO_7	PO_8	PO_9
EBUF	2	3	3	3	6	15	216	218	5
	38	39	83	245	215	271	*317	277	124
ESPT	3	3	3	3	8	3	4	4	2
	40	44	98	155	286	155	*291	146	134
ELPT	3	2	4	222	7	223	55	3	2
	38	39	84	286	220	286	*309	252	129
ESRPT	2	3	41	1	45	1	46	40	4
	47	39	145	58	297	39	*317	187	143
ELRPT	3	3	3	220	5	221	53	18	5
	38	40	82	287	218	287	*309	251	128
FBUF	2	3	3	3	7	219	219	222	5
	37	39	73	215	219	283	*315	276	133
FSPT	3	3	4	3	8	3	4	3	2
	41	44	101	139	*289	139	225	153	135
FLPT	3	2	4	206	7	206	205	3	2
	39	39	75	284	203	284	*311	256	134
FSRPT	2	3	41	1	44	1	45	41	40
	43	39	146	38	301	38	*302	189	173
FLRPT	3	3	3	204	5	204	203	202	5
	38	40	75	284	201	284	*309	256	132

* production completion time

E : dispatch decision when machine buffers are empty

F : dispatch decision when cart is free

BUF : scan along buffer stand

SPT : shortest imminent processing time first

LPT : longest imminent processing time first

SRPT : shortest remaining processing time first

LRPT : longest remaining processing time first

TABLE 3

START UP TIMES FOR PILOT EXPERIMENT

This table summarises workstation start up times :

time when machines first start on the first job (upper value) ; and

time when machine buffer is filled with a waiting job (lower value).

	MC_1	MC_2	MC_3	MC_4	Total(min)	MAN_1	MAN_2
EBUF	12'33	30'3	43'45	58'36	145.0	1'18	2'39
	15'45	33'6	56'48	1h7'54	173.6		
ESPT	12'27	27'15	33'45	14'51	88.3	1'12	2'18
	1h15'21	1h31'36	59'9	25'24	251.5		
ELPT	31'9	17'27	52'24	43'33	144.6	1'12	2'30
	1h54'42	20'27	1h29'45	46'30	271.4		
ESRPT	24'51	44'9	2h6'30	12'18	207.8	1'6	2'18
	28'3	1h7'45	1d15h4'27	15'33	2455.8		
ELRPT	50'45	35'48	38'12	12'18	137.1	1'6	2'18
	1h32'48	1h5'9	48'48	15'9	221.9		
FBUF	12'33	30'3	43'45	59'12	145.6	1'18	2'39
	15'45	33'6	1h1'27	1h7'57	178.3		
FSPT	12'27	26'3	33'42	13'45	86.0	1'12	2'18
	1h15'30	1h31'45	1h0'48	29'6	257.2		
FLPT	31'9	17'27	52'24	43'36	144.6	1'12	2'30
	1h55'0	20'27	1h29'45	46'30	271.7		
FSRPT	24'51	44'45	14h5'18	12'18	927.2	1'6	2'18
	28'3	1h5'15	14h35'9	15'33	984		
FLRPT	50'48	35'48	37'6	12'18	136	1'6	2'18
	1h32'48	1h6'24	53'42	3h18'9	411.1		

E : dispatch decision when machine buffers are empty

F : dispatch decision when cart is free

BUF : scan along buffer stand

SPT : shortest imminent processing time first

LPT : longest imminent processing time first

SRPT : shortest remaining processing time first

LRPT : longest remaining processing time first

related to the processing times of routings and location of pallets.

4.6.3 Findings

4.6.3.1 Simulation environment

The Chameleon simulation environment is well designed for the study of flexible manufacture control. The core of Chameleon comprises only the Prolog predicates that handle simulation times and queues, service functions that perform input/outputs, housekeeping functions to keep the system tidy and efficient, and the kernel. Using the very flexible data structure of Prolog, the definition of entities and events is not restricted in either complexity or number. The pilot simulation model is developed containing all the relevant features of flexible manufacturing system control but smaller in scale and/or complexity. This validation model can then be expanded stage by stage. As Prolog is not a sequential language, the addition of rules to the system can enrich the knowledge of the model, giving it greater capability without the need to discard or modify any of the rules previously in the system and known to be correct.

The degree of detail in the pilot model matches that of the production environment and gives the latitude needed to practice the variety of scheduling decision making in the control of flexible manufacturing systems. The various real time and off line user interface and reporting functions can satisfy the requirement for detailed analysis of decision making. The communication between logic using peer-to-peer queues is over-restrictive, it is very computation expensive to duplicate logic structures for all the decision domains.

Refinement of Chameleon to incorporate more advanced decision making concepts provides a most useful indication of the future in flexible manufacturing systems control.

4.6.3.2 Prolog for simulation

Prolog is very suitable for the simulation of control decisions. The Horn clause structure represents conditional statements in simulation elegantly. The list data structure can represent sets which can be manipulated conveniently. The non-declarative functions can be easily off-loaded to procedural C routines with the Prolog implementation used. In the procedural interpretation, a Prolog clause is seen as a sequence of function

calls, very much like C. Within each predicate call, the result can be generated by another procedural form or by the declarative form that allows multiple paths of solution according to situation.

The use of a personal computer based Prolog language implementation for this research has its virtues in being free from the restrictions of sharing facilities with other big computer users. The present generation of personal computers gives very respectable performance compared with a heavily loaded mainframe computer. The choice is further biased when colour graphics display and thus much better user interface is available at no extra cost in personal computers. The trade-off is in smaller memory size and raw processing power. The memory limit of 640K imposed by the PCDOS operating system means large models have to run in virtual memory mode, with pages swapping between memory and hard disk. This grinds to a crawling speed as the model is built up and frequent memory swapping is required. One megabyte of extra memory is used as virtual disk to alleviate this problem and the model is not expected to exceed this size. The performance problem is 'solved' with the migration of the system to a workstation class IBM PS/2 Model 80, giving four times more processing speed.

The PC based Arity/Prolog implementation has a run time limit on stack memory that holds the history of execution for backtracking. This limits the complexity of the rules that can be used. Since a queue structure is used for execution, complex operations are broken down to smaller manageable units and fed sequentially through a high priority queue for processing. From the above discussion, it is evident that processing speed is not a strong point in the present system. This slower speed is accepted in exchange for the greater complexity of rules possible and the friendly development phase.

4.6.3.3 Decision making

Decision making in flexible manufacturing control is more complicated than strict forward application of decision rules. Decomposition of the decisions to form the solution methodology depends on the environment the solution methodology is going to work. In the case of FMS control, knowledge about the overall production requirement is equally useful as the local knowledge of the impending processes. This can be stated as overall system environment as a result of the configuration state. Links further up to a MRP system can help with decisions with longer term effects.

The structure of decisions affects the performance of the system as it determines the use

of suitable and timely information. A decision rule is not useful if the situation does not provide alternatives to choose from, or the decision results are made so far in advance that the criteria or the data of decision is no longer valid.

The embryo control systems using simple scheduling rules BUFFER SCAN, SPT, LPT, SWKR, and LWKR were not satisfactory. Visual tracing of the scheduling decisions using the graphics display showed immediate inadequacies. The load/unload stations are bottlenecks during system start up. The processing time selection rules applied at these stations do not consider the processing time requirement for the machining operations afterwards, creating the idle times as shown in the results. There is no control over the flow of critical work as this is not recognised in processing time rules. In production phases when the load for the cart is heavy, the job selection decisions commit jobs that are no longer optimal to the rule when the cart can deliver it. Delays of the system occur when the transport system is overwhelmed. It becomes quickly apparent that a different approach to making decisions is necessary according to the dynamics of the production. Critical resources and thus the goal and effect of the decision rules vary, so do the criticality and priority of parts. Scheduling in real time requires the organisation of decisions according to the information available and the range and effects of those decisions.

A high degree of human intervention may be required [140] in efficient control decision making. The sharing of automated and human decisions varies with system complexity and operations methods. Areas where control decisions could be automated should be investigated and implemented, leaving the human element to handle exceptional circumstances to best effect[141,142].

An important first step is to define a flexible framework for organising decisions to meet the dynamic environment.

CHAPTER 5
GENERIC CONTROL ARCHITECTURE

5.1 CONTROL

In a treatise on control, Tocher [143] stated the necessary conditions for a precise control problem as:

1) There must be a specified set of times at which a choice of action is possible.

2) At each such time, there must be a specified set of actions from which to choose.

3) A model must exist which can predict the future history of the system under every possible choice.

4) There must be a criterion or objective on which the choice of action is based by a comparison of predicted behaviour of the system with the objective.

These are the four requirements to answer the when, what and how in designing control decisions. Current academic models concentrate on the definition of the control problem as a single decision making process and the methods of optimising the decisions[94,95,98]. In real control systems, decisions are made continuously to react to system behaviour [143][Fig.22].

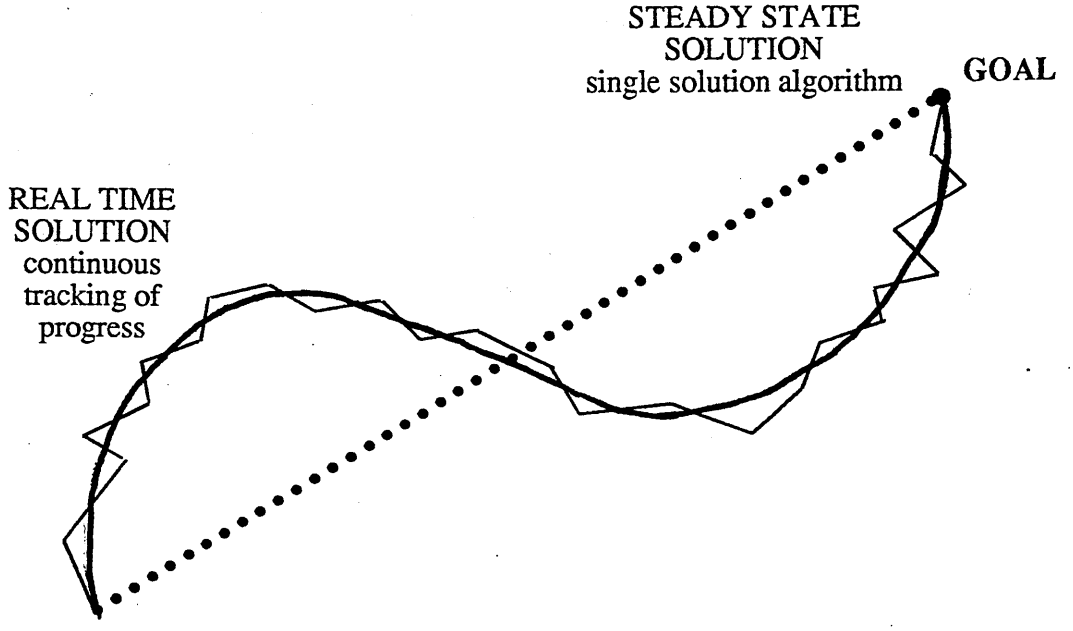


Fig 22. Real time solution vs Steady state solution

Intelligent control models employ objective tracking and criteria determination to suit the dynamic environment. Dispatcher and monitor are used to re-adjust the planned schedule[129,130,132] but do not address the detail decision timings. Chryssolouris [131] discussed the effects of the timings of decisions but MADEMA was not designed to handle variable action times.

In the real time control of FMS, the possible decisions and decision times are determined by the system operations model. The possible choices of action is limited by the tool and pallet constraints. Tool and pallet configuration is better performed off-line because of the operations logistics involved. This leaves the core real time control functions of a FMS as dispatching of parts and material handling device. Automatic learning is superfluous for the small set of action choices and short time horizon; which is also not realistic within the cost constraints of a generic control system. A good model to predict system behaviour requires local knowledge of system and heuristics from job shop scheduling research. Knowledge based representation provides a means of capturing the qualitative and quantitative characteristics of the model. The same applies to the definition of the time dynamic objectives of the system.

A model for generic control of flexible manufacturing systems has an architecture that accommodates all the decisions and provides all cell dynamic information for a suitable intelligent scheduling control system. It is a framework to organise the what, when and how in scheduling decision making.

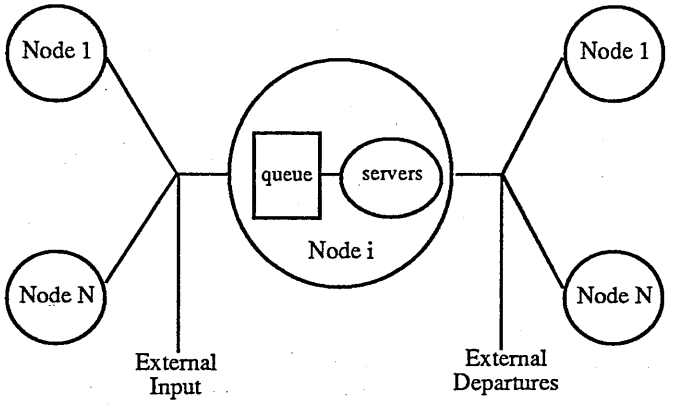
5.2 FORMAL SCHEDULING DECISIONS STRUCTURE

5.2.1 Generic resource decision model

5.2.1.1 Extended network of queue model

A scheduling system is designed to coordinate the resources within the system to perform the stated tasks. A system model of all the resources is necessary for the complete specification of the decisions requirement.

The 'server-customer' model of queuing theory forms a sound basis for the analysis of decision making design. In a Jackson network of queue model [144][Fig.23], each processor has a queue of tasks to process and the characteristics of the queue are assumed and modelled. Solberg [92] refined this model for FMS control [Fig.24] by



Node i in equilibrium

Fig 23. Jackson network of queue model

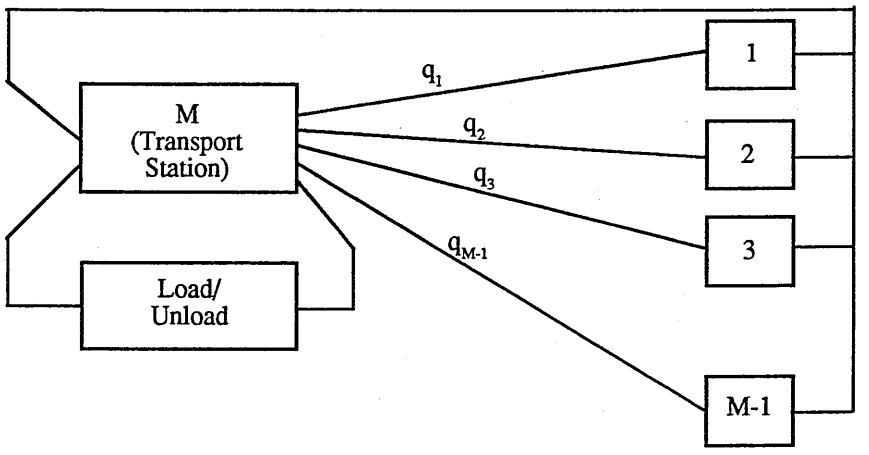


Fig 24. Solberg network of queue model

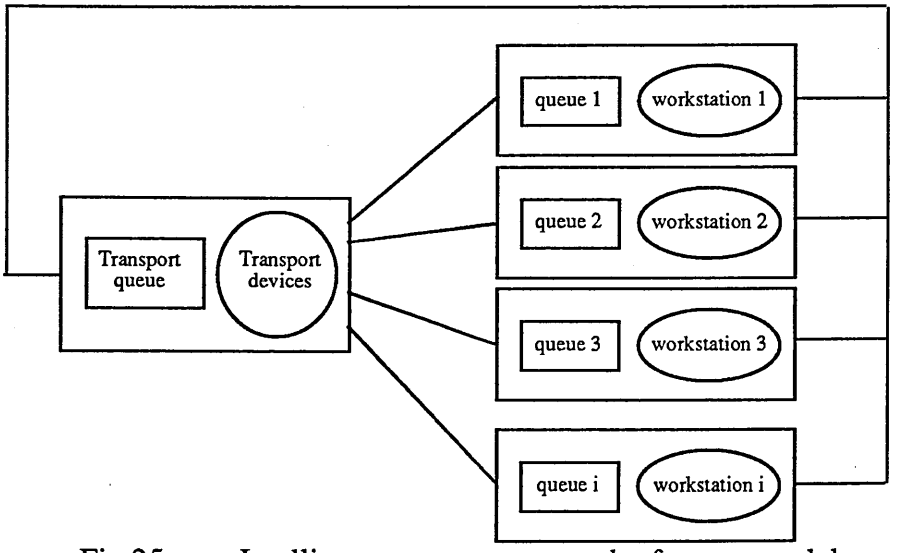


Fig 25. Intelligent processor network of queue model

adding the material handling system as an essential consequence of all processes with stochastic processing times. In detail scheduling decision making design, the same network of queue model is used with the formation of the work queue as the design problem. Rather than using the solution of model to give aggregate steady state production estimates, the model is used to identify the necessary decisions in FMS control and the structure to link these decisions. Each processor in the model assumes its individual characteristics which makes a general mathematical solution impossible.

Given an arbitrary processor representing a workstation [Fig.25], there is a set of tasks or jobs waiting to be processed. Tasks that join this set remain until it is processed. The physical machinery layout affects the characteristics of this set. If no buffer positions are linked to the machine, the physical size is zero; or there may be a fixed buffer between processors, setting the size of the set. If the buffer can only be accessed in a 'first in first out' manner, say a conveyor, the waiting set becomes a fixed queue. The buffer may be accessed in a random manner, as a pallet carousel, then this set is a true random set. Given a flexible material handling system, like a random access buffer bank, the formation of the waiting set is purely conceptual. Tasks are assigned to a processor as control information but not physically queued at the processor. This waiting set will be called the queue according to queuing theory convention, though it is a randomly accessed set rather then a sequentially accessed queue.

This queue has to be filled to feed tasks to the processor. This can be defined as the queue formation problem. The queue formation problem is defined by two sub-problems. The task selection problem and the timing of task selection, essentially, the how and when of performing queue formation.

5.2.1.2 Queue membership

The task selection problem is equivalent to the job despatch problem in classical job shop scheduling research. Panwalker [50] listed more than 150 rules for this problem. Without loss of generality, task selection rules are classified into six categories according to their emphasis:

- processor : rules that emphasise processors utilisation, mainly reflected in using forms of task processing time as decision criteria;
- task : rules that emphasise task completion time, characterised by using due dates and processing times as decision criteria;
- task group : rules that emphasize the completion of a group of tasks, as in assem-

bly, the product structure is the reference for these rules;

- delivery : rules that concern the timely delivery of tasks to the processor, applicable in situations where the processor has to be supplied with tasks from an external source;
- random : rules that do not utilise any information on the processor or tasks; and
- switching : intelligent algorithms that switch emphasis according to situation and apply the corresponding rules.

These categories are not mutually exclusive, the same rules may be applicable to more than one emphasis. The function of rules has been studied by more than thirty years of simulation. Hybrid rules combining the power of rules in different categories had also been used as compromises; the performance of these are very sensitive to the production environment. There is no definite guideline for rule selection.

In real time situation, the set of tasks available for selection is limited by the tool and fixture constraints. Configuration planning can reduce the effectiveness of rules by presenting only a small number of tasks at decision points.

5.2.1.3 Queue formation times

The timing to initiate task selection is crucial to the efficiency of the decision, for membership of queues are irrevocable and empty queues cause under-utilisation of the processor. The potential timings are:

processor starts and stops;
queue size changed or emptied;
extra tasks available;
delivery device starts and stops;
external initiation; and
fixed period repetition.

The control of the timing should reflect the real time situation. If the queue is emptied, the appropriate timing is when extra tasks are available. When the queue is full, new tasks can only be accepted when the processor has consumed a task from the queue, signalled by either the processor starting on a new task or a change in queue size. If the delivery device is a bottleneck, the delivery device stop is a better decision point, as there is no point in making assignments that cannot be effected until much later.

The timing of decisions at the extra tasks available point allows the implementation of the job view of scheduling. From this point, different processor queues can be evaluated for the readied job to join.

5.2.2 Integrating queues

Apart from decision integrity, the timing of the decisions have major implications in the computing requirement. Computation procedures for task selection can be lengthy, the correct initiation of procedure can reduce reaction time of computers tremendously.

The design of a complete scheduling system is the specification of the queue formation problems for all the processors in the system. Traditional research employs only one solution strategy for all processors. The classical Jackson network of queue model is characterised by sequential first in first out queue selection at the point when a job is available [144]. Alternatives in processor selection are not allowed. Human shop scheduling is very different.

The contention problems in integrating the queues are when decisions are needed at the same time. The two contentions are processor selection and job selection. Processor selection is needed if the timing of decisions for several processors are coincident. The resolution of precedence reflects the relative importance of the different processors. This resolution problem is distinct from the machine selection problem where jobs select prospective machines. The job selection contention problem is applicable to the job view of scheduling, when multiple jobs are searching for queues to join. The sequencing of jobs is of necessity related to the priorities of them. These two contention problems are inter-related, any a priori sequence in resolving them will result in decision orders that do not take all information into account.

Not all processors need a separate control decision logic, in systems where workstations are similar and the production requirement is stable, differences in requirement for the system can be small. Workstations can be grouped to share the same set of logic, thus reducing the software development effort.

5.3 UNIVERSAL CONTROL ARCHITECTURE

5.3.1 De-coupling control from scheduling

Current practice is to incorporate decision strategies into the control systems to produce a compact software system. The software produced with this approach has an architecture that forms a fixed decision framework. Most 'intelligent' control systems built to this approach can only select particular rules from a rule set rather than adapt to the dynamic environment. The systems designed have a fixed sequence of decisions and the architecture is specific to the particular system installation.

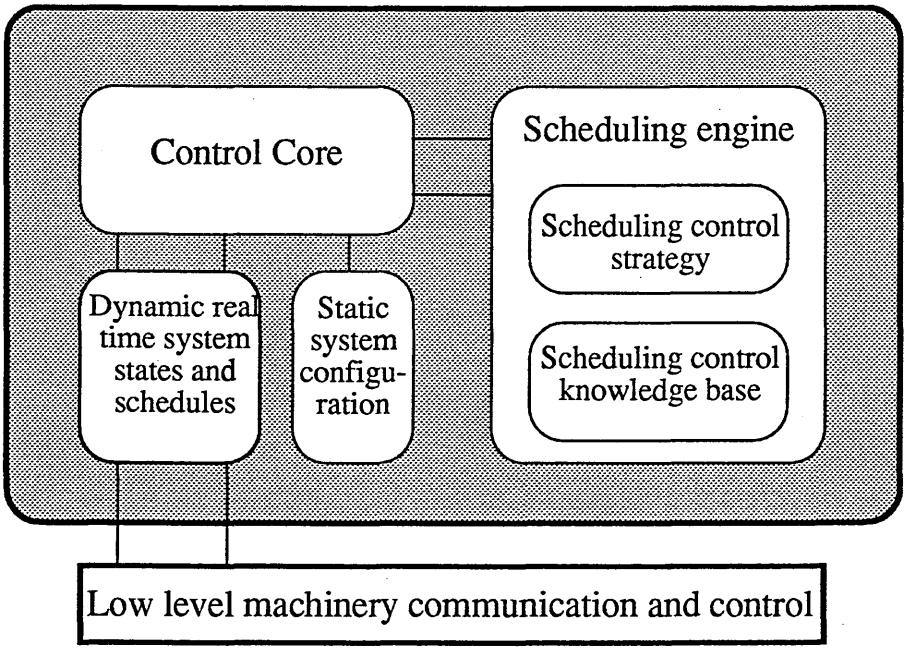


Fig 26. Universal architecture for generic control

A universal architecture [145][Fig.26] for generic control is designed that separates information processing from control strategies. The core of the system maintains current information on the states of the system. This minimal control core activates the scheduling engine to perform all necessary decisions. The activation of the scheduling engine depends on the type of decisions required.

5.3.2 Control core

The control core acts as the intelligent interface that isolates the scheduling systems from the machinery installation. There is an active and a passive element to this interface. The active element provides requests for decisions to the scheduling engine. The passive element provides data for the scheduling system to perform decisions and translates the decisions to schedules for system execution. The databases have to be maintained in real time and their integrity ensured.

The control core has to service the scheduling engine with two kinds of decision requests: time activated and event activated. Time activated decisions are those that are scheduled to occur, like the regular dispatch of AGVs to recharge their batteries. Event activated decisions are required in response to changes in system states, like the scheduling of a job to a machine when it is available for new work. The two can be unified to one. Events can be planned to occur at specific times to reduce all decisions to time activated one. However, real life systems are susceptible to disturbances so the timetables generated are seldom reliable. Time activated decisions usually relate to some events, and so can be reduced to event activated. This usually means the addition of procedures that continuously check certain states to detect the relevant event; making this conversion is computationally expensive. A universal control core needs to maintain a time scheduled list of activations and the ability to accept real time event occurrences and activate the scheduling engine. The scheduling engine does not need to respond to all activations, it works according to the strategy it adopts.

5.3.3 Scheduling engine

The scheduling engine contains all the knowledge for making decisions. The design of the scheduling engine depends on the particular installation and production requirement.

A functional model of the scheduling engine [Fig.27] holds the definition part and an execution part. The definition part holds the specification of the queue formation problem, the contention resolution procedure and the rule set for the execution of queue formation. The execution part consists of the firing mechanism to interpret the timing information. The contention resolution mechanism then uses the defined procedure to provide the decision to the decision mechanism. The decision mechanism accepts the decision from the contention mechanism and employs real time data and made the decisions using rules from the rule set.

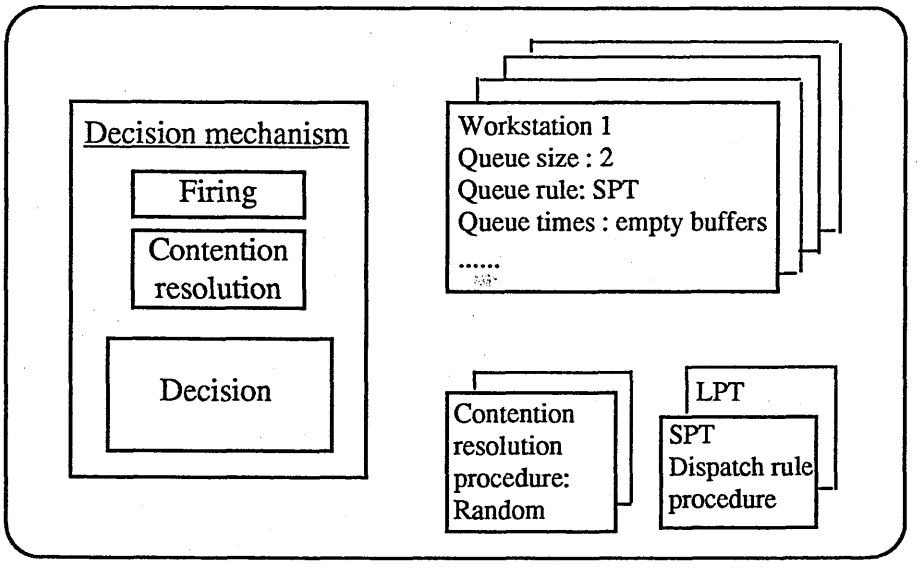


Fig 27. Functional model of scheduling engine

The specification of queue formation for all workstations can be conveniently represented in a frame based structure. The contention resolution procedures and the rules in the rule set are in a suitable procedure form. The definition of the procedures can be integrated with the corresponding mechanisms to reduce software overhead in implementation.

The separation of definition from the actual mechanism allows the incorporation of a meta-level monitor that changes the decision strategy in an adaptive manner. The states of the system is monitored and scheduling strategies modified according to the current system situation. With a separate definition, this requires the replacement of the definition of procedure only, without the trouble of modifying a procedural mechanism. This model of the scheduling engine is designed to achieve low cost software design and implementation. Batteries of standard definition frames for workstations and decision procedures for decisions can be defined. The specification of the complete scheduling system becomes an exercise of integrating standard modules from these frame and rule bases to suit the particular system requirements. Modifications to satisfy special needs can also be done easily with these generic sets acting as reference models.

This implementation requires a good understanding of the dynamics of the system and access to all the relevant information.

5.4 FMS SCHEDULING DECISIONS

5.4.1 Tool and fixture configuration

An important operational problem for controlling flexible manufacturing systems is the determination of the tool and fixture configuration of the system. In most low cost FMS, the number of tool pockets and buffer positions is limited, and the loading and setting of both tools and fixtures are not automated. This important decision has to be made with suitable lead times for the preparation of the tools and fixtures. This is best done off-line with the aid of capacity planning tools. This fixes the configuration and capability of the system for the time period and forms a constraint for the real time scheduling system.

In performing tool and fixture configuration planning, some machines and orders will be known to be more heavily used than others. This information is most useful as a guide to the general shop situation for real time decisions.

5.4.2 Job scheduling

There are two main perspectives to the job scheduling problem. From the workstation point of view, the decisions are in the formation of a queue of waiting jobs to be processed, and then the sequencing of jobs in this queue to select the next job to be processed. The waiting queue size can be dictated by the physical organisation of the buffer position. In conceptual organisation, the buffer size is the maximum number of jobs that can be queued and the total processing time of jobs in the queue. The sequencing of jobs in the queue can be dictated by the physical hardware layout. When there is scope for selection, this selection can be on the basis of better utilisation of equipment, criticality of jobs or other shop production objectives. This view is fully represented by the network of queue model.

The job view of the scheduling process is centred around the jobs moving in the system. Jobs select the processing queues to join to complete the routing. Machine selection is needed only when alternate routings are feasible. The criteria for this selection again can be based on better utilisation of equipment, criticality of jobs and other shop production objectives.

For the load/unload stations, there are also the decisions on part launches and batch launches. Both the workstation and job views apply and so do the decision criteria,

though on a larger scale.

5.4.3 Material handling system scheduling

In a simple rail guided vehicle system, the material handling system has to decide how to position the various pallets and deliver the required pallets to the workstations. The characteristics of the material handling decisions are governed by the FMS hardware layout. The demand on the transport system is affected by the ratio of processing times to transport times. Transport times can be insignificant or dominant, but more often the importance of transport times fluctuates according to the near random demand of transports by the workstations.

5.4.4 Interaction of scheduling decisions

In a multi-machine system, the job queue formation and selection problems for each machine interact with one another. Since a job queue commits assignments to one machine, this reduces the flexibility in selection for other machines. The 'minimal commitment policy' or opportunistic scheduling [146,79] that assigns jobs to queues at the last moment minimises these interactions. The timing of queue formation decisions for different machines has to be resolved satisfactorily to ensure conflicts are properly handled. In cases when an overall loading situation is known from the configuration programme, the heavily loaded machines can have preference over others. However, there is the danger of starving others machines when critical machines are overloaded with jobs that can be shared. Treating all workstations equally may reduce the utilisation of the critical machine, affecting all downstream workstations. In well balanced systems, bottleneck shifts in real time, further aggravating the scheduling problem.

The scheduling system has to accommodate the job view of scheduling. Jobs need to join queues for processing. Especially for critical jobs, it is important to be able to plan ahead and perhaps even hold back other jobs in the queue of the next processing workstation to allow rapid transit. A switching of view is necessary to function in a complicated environment.

There is a close relationship between workstation scheduling and material handling system scheduling. The timing of pallet movements decided by job scheduling can cause a sudden peak in transport requirements, overwhelming transport capacity. This results in pallets not being delivered in time and idle workstations. In many cases, a dif-

ferent job selection can be made without loss of workstation utilisation and have no effect on job criticality. This results in the transport system being able to deliver the pallets in time for processing without extra workstation and job waiting. With buffer stations in machines, this can usually be achieved with good scheduling.

These interactions are not studied by steady state research. Queuing and aggregate simulation models consider these as random losses in the capacity of the system. In real time control, especially with the aid of human decisions, these dynamics could be studied and used to advantage by employing more complicated scheduling systems.

5.5 KNOWLEDGE FOR JOB SCHEDULING

5.5.1 Information requirement

For the part launch problem, the part can only be launched when the necessary pallets, fixtures, tools, machines and numerical control programs are ready. Apart from these resource requirement constraints, the decision to launch is also governed by scheduling considerations. Part launch affects the current job mix in the system, this influences the utilisation of the system and the flow of other jobs in the system. If the part is one of a batch, then the due date and priority of the batch are also factors affecting part launch. For batch production, there is the additional problem of batch launch, when to launch the first part of the batch and when the subsequent parts become candidate launch items.

In systems where alternate routings are available due to tool duplication or process commonality, then a job can select the machine that it employs to complete the process. The resource constraints are machine capability, the ability to accept the fixture, the power and work envelope of the machine and the availability of tools and NC programs. The scheduling decision factors are the relative efficiencies of the candidate machines, the size of the queues of the machines, the waiting time of the job, the criticality of the job that may dictate the use of less efficient or more expensive machines to complete the process rather than waiting.

The job selection problem is the selection of jobs from a queue of jobs waiting to be processed. The resource constraints are the same as the job selection problem. The scheduling factors are the processing time of the operation, the relative criticality and priority of the job, any planned maintenance and tool changes.

The transport problem includes the selection of transport if more than one is available; the selection of transport routing if alternatives are available; the order of job to execute if there is a queue of transport requests waiting. The constraint is the ability of the transport to move the part, which is not a problem in most flexible manufacturing systems. The decision factors in handling multiple requests are the distance of travel to pick up, the time required to finish the transport movements and the relative criticality and priority of the movement request.

The above analysis points to two different kinds of information requirement:

the real time availability of resources provides information for constraints related decisions; and

the information required by the scheduling system to optimise the work flow.

Classical scheduling research employs three types of dispatch factors: processing time, due date and random. This information is usually extracted from the routing structure. These dispatch factors are used on their own or in combinations in job dispatch rules. To make relevant decisions with these rules, the system has only to track the job progress and keep the current time for the operation of these rules. Relative machine efficiencies and the size of the waiting queues are the major factors used in machine selection algorithms. Relative machine efficiency factors are provided off line as known information, queue sizes depend on the dynamic situation of the system.

For real time information, apart from the states of system that were cited above, state change events and their timings are important for the decision system to time the decisions. In fact the state information can be generated from the state change events and maintained within the decision engine. This involves the risk of data discrepancies induced by hardware failures. These discrepancies can be checked and corrected with an overhead on the decision engine computer.

5.5.2 Knowledge for good decisions

Not all systems require the same set of decisions. The decision strategies that structure the decisions can be tuned to satisfy the particular requirements. Some systems have only one machine for each operation, thus having only the job selection problem and no machine selection problem. Other systems may have fixed sequence physical buffers as queues, waiting jobs are always processed in a 'first come first serve' manner and there is no job selection problem. It is common to have a mixed organisation where both

machine and job decisions are necessary. The structuring of decisions depends on the particular production environment of the system. This does not necessarily relate to the flexible manufacturing system machinery hardware configuration. The control strategy depends more on the system designers' perception of the production dispatch problem. It is important that a universal architecture provides data for different structures of decision making.

The structure of decision making also concerns the timing of decisions. Different orders of decision making produce very different results. In job queue formation, the decision commits the job to a certain machine, if the machine breaks down before the job can be processed, then the waiting time of the job is wasted and the job will have to be rescheduled. The further the planning looks ahead, the less resilient the system is to unplanned disturbances. However, there are system preparation operations like tool setting and changing that require a relatively long lead time. The control system must plan ahead enough for these essential operations to be executed. Again, the actual design depends on the production environment of the system. However, the control system must be supplied with all the real time information for it to time the different decisions.

5.6 DATA INTERFACE

5.6.1 Forms of data

There are two forms of data. Time related events that have a specific time attached, such as a pre-scheduled event. The other type is data expressed as states or values that are true until changes occur. Events are best organised into queues that bear the time sequence of the events. Because of the dynamic nature of queues, their lengths are non-determinative and queue handling software needs to be developed to facilitate their representation and maintenance. State values can be defined easily and their representation can be simply achieved as variables or facts in programming languages.

5.6.2 Lifespan of data

Data can be classified and hence organised by its stability into 'static' and 'dynamic' databases [Fig.28]. This allows suitable integrity and data transformation functions to be structured. The static database contains data that are seldom changed, defining the system hardware and operation modes. The dynamic data consists of data that change

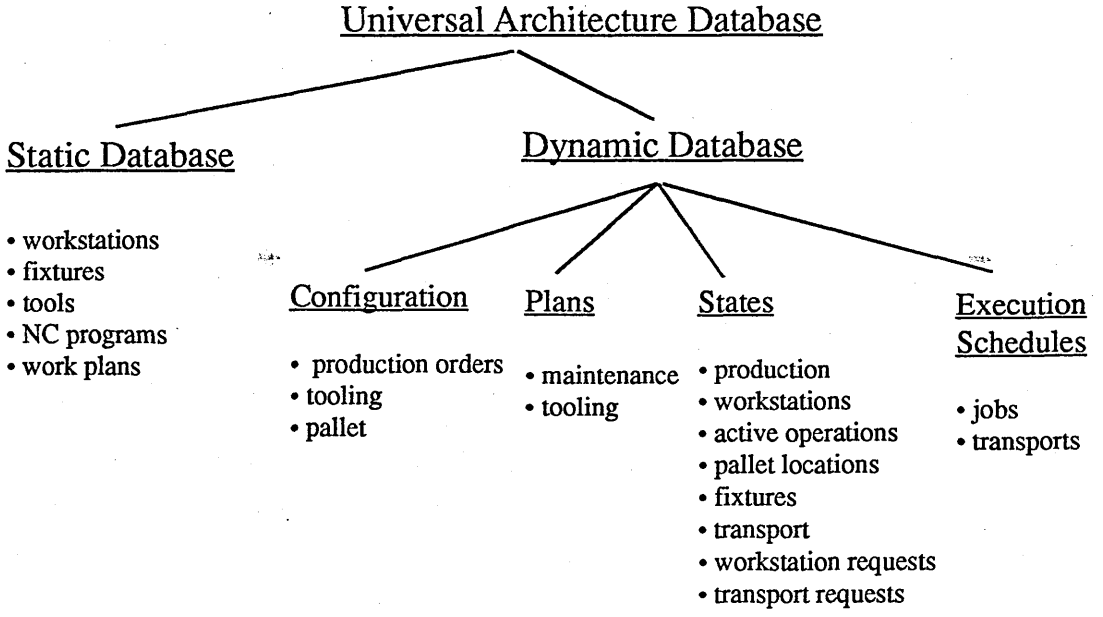


Fig 28. Database organisation of univeral architecture

with the flow of operations.

5.6.3 Static database

The static database includes:

• workstation data

the workstations in the system and their capability, maximum number of tool pockets and the acceptable type of pallets;

• fixture data

the fixtures in the system, the components the fixtures can hold, the maximum number of components that can be mounted and the pallets that it can be fixed to;

• tool data

the tools used in the system and their expected life;

• NC data

the NC programs needed for machines; and

• work plans

for all components, the necessary fixtures, all the routings, the workstations that can perform the processes, expected operation times and the associated tools and NC programs.

Some of these data items are updated when necessary, but the preparation of work plans has traditionally been performed by production engineering departments beyond the operational control of manufacturing systems.

5.6.4 Dynamic database

The dynamic database is subdivided into three databases according to the nature and the update requirement of the data. The 'configuration' database contains information about the tool, pallet configurations and the production requirements within the immediate time span. The 'plan' database holds the planned maintenance and tool change operations. The 'states' database contains all the real time states that reflect the current situation of the system.

The configuration database contains data that forms the operations environment of the manufacturing system at the particular time, it is composed of:

• production orders

the required quantities and due dates of the batches of components ordered, and all necessary identification and authorisation information;

• tooling data

the tool assignment in the machines and the life of these tools, for scheduling purposes; these can be abstracted to the machining operations the machines are tooled, it is useful to have an indicator as to the expected demand for the tools; and

• pallet data

the fixture mounted on the pallet and the order and workplan to which the pallet is assigned; it is useful to have an indicator as to the expected demand for the pallet.

The plans database consists of plans of activities in the form of queues:

• maintenance plan
holds the scheduled maintenance plan; and

• tooling plan
holds the plan for tool changes.

The states database holds the system data that are changed most rapidly. These state data are:

• production progress
the state of completion for all current active production orders, the quantities ordered, scheduled, started, completed, scraped and reworked;

• workstation states
the real time states of the workstations: busy, idle, down or fault;

• production states
the current operations being active, the workstations and the job specifications, and the time operations started;

• pallet locations
the locations of all the pallets;

• fixture states
the states of the fixtures, the number of components mounted, the remaining routing, operation time of next operations, total remaining operation times and the due time for the current component load;

• transport location
the location of the transport vehicle;

• workstation request states
the workstation request states: waiting or scheduled, the expected demand of the workstation and the next free time; and

• fixture request states

the fixture request states: waiting or scheduled, the expected demand of the fixture and the next free time.

All of this information can be established if the relevant state change information is received from the machining system at the following times:

start of processing at all stations;
end of processing at all stations;
start of transport motions; and
end of transport motions.

It is necessary to design the system such that the actual states of the workstations can be checked directly to ensure system integrity.

5.6.5 Communications with low level devices

The partial schedules worked out by the system are kept in two event queues:

• job event plan

consists of all the planned events for all the workstations; and

• transport event plan

consists of all the planned events of the cart.

These plans consist of one or more events depending on the planning horizon of the system, a plan representation gives the flexibility to suit the control strategy used.

CHAPTER 6
SCHEDULING SYSTEMS DESIGN CASES

6.1 A SCHEDULING SYSTEM DESIGN METHODOLOGY

A methodology is developed for the design of scheduling systems. Because of the complexity of factors involved, the detail design procedure forms a checklist of considerations rather than a prescriptive algorithm. See Fig.29.

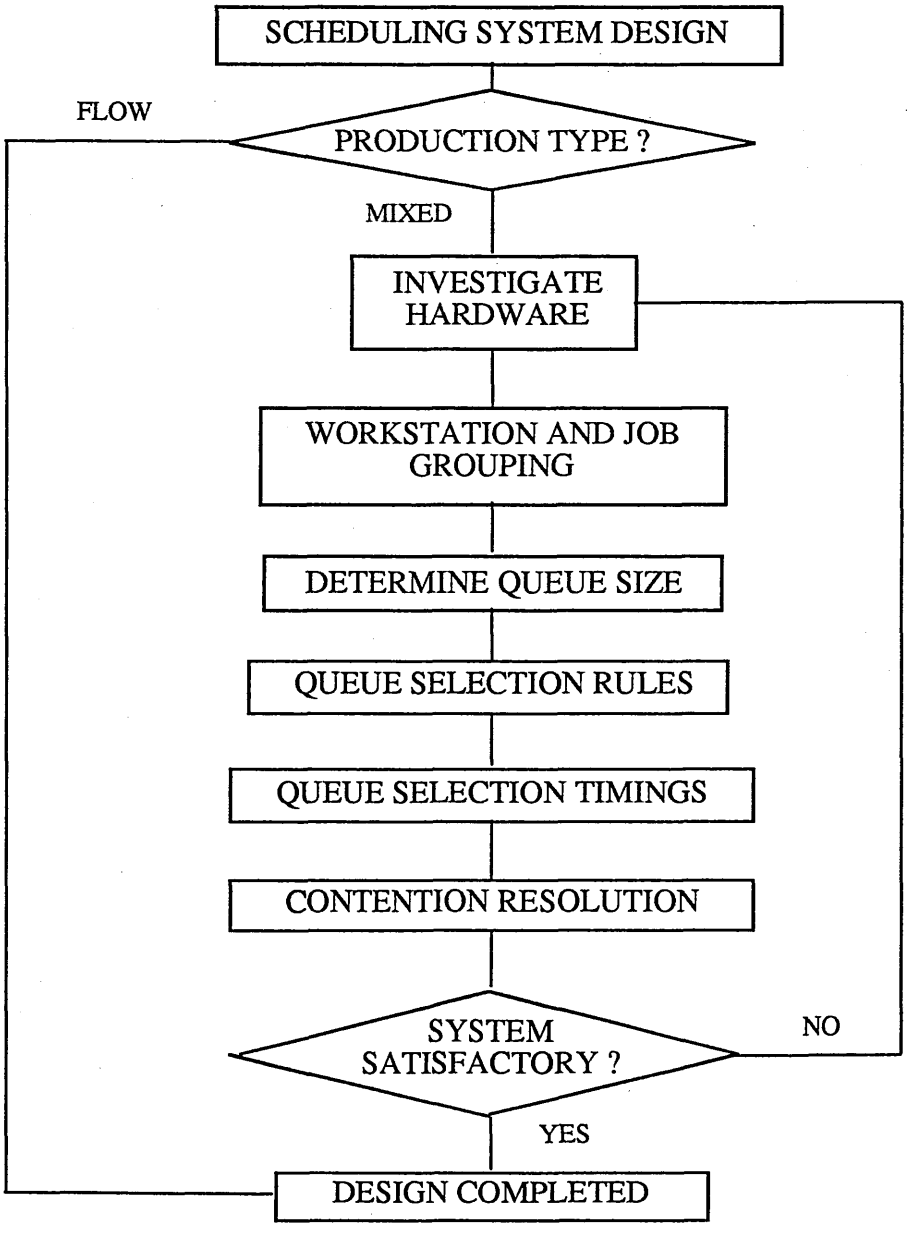


Fig 29. Scheduling design methodology

Production type determination :

An early distinction of the production type between flow and mixed allows the simple flow type to be isolated. The analysis gives the scope and complexity of the overall problem. For a strictly flow system, queue formation rules are not necessary. There has to be a number of jobs waiting to join the queue for queue selection to be effective, and this depends on the production type of the system.

Hardware features :

The hardware characteristics affect the queue sizes and the queue selection rules. The match between hardware and the production requirements sets the requirement of the scheduling control system.

Workstation and jobs grouping :

This step seeks to simplify the design by reducing the types of workstations and job types and thus the logic. If all workstations are identical and no product grouping can be performed, the same logic can be used for all workstations. If no grouping is possible, the queue formation analysis has to be done for all workstations.

Queue size determination :

The importance in queue size determination is the commitment of jobs to workstations and the information requirement for scheduling. It is desirable to have some jobs in the queue so that the workstation does not have to waste time. On the other hand, commitment of jobs to workstations prevents jobs pursuing other processing opportunities that might occur. Also long queues may stop other more urgent jobs from passing through if the queue or selection rules are inflexible. A longer queue is needed to project to a longer horizon and plan other activities based on this knowledge. There is a conflict between the real time flexibility of response and the desire to plan into the future.

Queue task selection procedure determination :

Using knowledge about the system, the most likely selection rules can be identified from the categories defined in Chapter Five. The intelligent category requires identification of the different possible phases the system may go through and the events that mark the changes. Rules relevant to each category can be found in scheduling research work. There are certain accepted results, the application requires good understanding of system operations.

Queue formation timing determination :

The timing of decisions can be picked from the groups defined in Chapter Five. The timing should balance the information need for planning and least commitment for flexibility. If possible, flexible timing schemes that employ knowledge of current queues should be used.

Queues integration :

The procedure for resolving potential contention between all workstations and job decision timings are developed. An analysis in the interaction between the different workstations is performed and the queue formation problems reworked if necessary.

6.2 A DESPATCH SYSTEM FOR PRODUCTION SHOP

One scenario in the use of flexible manufacturing systems is as a component production unit making parts to stock within a large production organisation. High level material requirement planning systems designate production targets for the system for short weekly production horizons, changes in production within the week being unlikely. Master production scheduling using capacity planning ensures a stable workload. The production information arrives in time for suitable off-line tool and fixture planning.

In this system, critical job consideration is not applicable as the production is planned by master production scheduling to capacity. A real time scheduling system that ensures high utilisation of machines can satisfy the production need.

Applying the design methodology for the design.

Production type determination :

A batch type production of parts that have mixed routings in the system.

Hardware features :

The four machining centres system with in-line rail guided vehicle hardware system as described in Chapter 4 is used. Each machining centre has a pallet transfer station to act as immediate buffer. Load/unload stations do not have buffers. A bank of 15 buffers can be shared between all the work stations. The rail cart exchanges the pallet from the station to which it delivers a pallet.

Grouping :

No specific grouping is applicable, some common tooling is used in the system.

Identical control logic is used for all workstations.

Queue size determination :

Since critical jobs are not important, there is no need for advanced planning data. Applying the least commitment policy, the queue size for machines is one for the physical buffer station. The queue size for load/unload stations is zero. This allows rapid response in the use of work stations to react to breakdowns.

Queue formation rule :

The 'shortest processing time first' rule known for high system utilisation is used. This rule follows the least commitment policy in reducing the amount of time committed in the processing queue. The known problem of slow production rate for long processing time jobs is not significant as they will be processed within the planned week. Ties are broken on the proximity of pallet to the current cart position.

Queue formation time :

To ensure that machine time is not wasted by unnecessary waiting, a job is selected when the queue is emptied, signalled by the machine finish processing a job. The load/unload station is refilled when processing on the pallet in the station is finished. Since there may not be parts available to join the queue then, the additional formation time when a job is finished is used. The job is checked for suitability and a request for a job is made as appropriate.

Queue integration :

Contention for when more than one machine is waiting for a job is resolved by the order of their requests.

6.3 A DESPATCH SYSTEM FOR JOB SHOP

Another scenario in the use of flexible manufacturing systems is as a production unit making components to urgent order, like producing parts for field repair or development models. Advance information on production requirements is scarce and excess capacity is allowed because of the time critical nature of the jobs involved. High machine utilisation is not relevant. A fast throughput time is important.

Applying the design checklist for the design.

Production type determination :

A random type production of parts.

Hardware features :

As above.

Grouping :

As above.

Queue size determination :

The least commitment policy is valid as there should be minimum amount of work that can possibly block an urgent job. Thus the queue size for machines is one and that for load/unload station zero. The dynamic formation of queues is used to plan for critical jobs.

Queue formation rule :

A switching algorithm is designed. Jobs are classified into late or normal. For jobs that are late, the 'earliest due date first' rule is applied. For normal jobs, the 'minimum slack time' rule is applied. This allows jobs to be ranked according to their urgency and processed accordingly. Further ties are broken by the proximity to the work station.

Queue formation time :

Taking the equipment view of scheduling, the queue formation time is as before.

Queue integration :

A more effective contention resolution mechanism is used. When there are more than one machine waiting for jobs, all the jobs available are checked for their criticality. The machine that process the more critical job has priority. Further ties are broken by the machine requests sequence.

6.4 AN ADAPTIVE RULES SYSTEM FOR DYNAMIC SHOP

A more complex application of flexible manufacturing system is when it is the main machining resource of a production sub-contractor producing parts to order. No long term planning information is available and production is continuous with no specific cycles. The production requirement is not steady and critical jobs have to be rushed

occasionally. As the main machining resource, the utilisation of the system must be high to maintain profit margin and to allow reserves for contingencies.

Applying the design checklist again.
Production type determination :

A mixed batch and job type production with random routings.

Hardware features :

As before.

Grouping :

Grouping not possible. Same logic applies to all workstations.

Queue size determination :

The least commitment policy is still valid, the minimum queue size used.

Queue formation rule :

A two tier hierarchy of rules is used, based on the complex dynamic shifting of immediate production emphasis [Fig.30]. The system is checked for critical jobs, if any exist, they are processed first. If the system does not have critical jobs, the scheduling depends on whether there are workstations blocked or idle, starved of work. If there are blocked workstations, jobs are selected to 'unblock' them as soon as possible. If there are no blocked machines, the closest job is selected to minimise the idle time of the work station. If the queue is formed for the buffer transfer station only, the status of the rail cart is checked. If the cart is heavily loaded, the pallet closest to the workstation is delivered, or else the 'shortest processing time first' rule is used.

Queue formation time :

The time that the cart completes a delivery is used to trigger queue formations. As all pallet movements must be performed by the cart, this is the point where decisions can be effected immediately. This decision point requires information from the material handling system, which may not be available.

Queue integration :

Contention resolution works with a system monitoring scheme. The approach is called 'goal identification' [147] because the system attempts to identify the immediate sub-goal of the system according to the critical resources determined. The cart is check

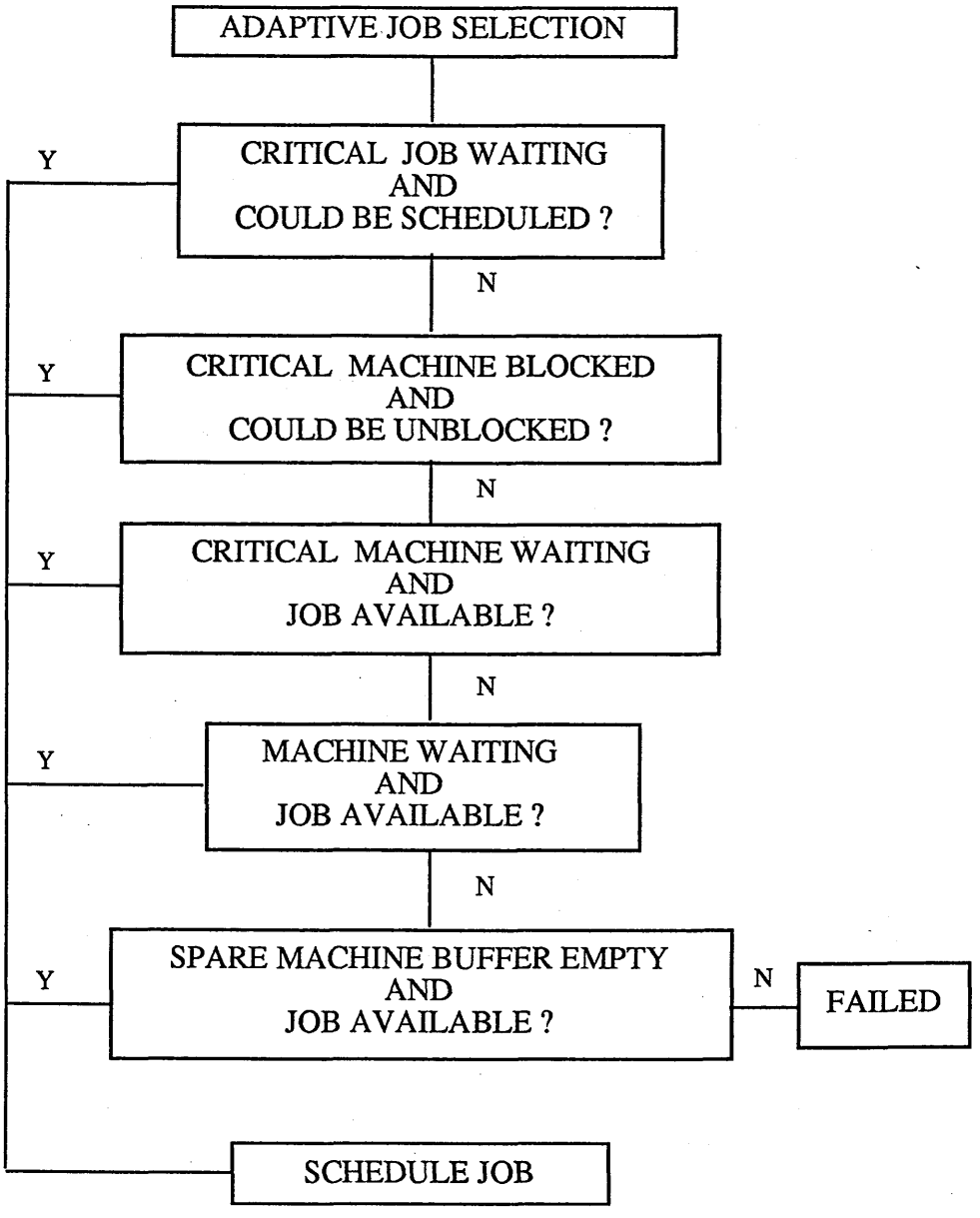


Fig 30. Adaptive job selection rule

for its criticality, and the status flagged for queue formation. The system relies on the tool and fixture configuration system to identify planned heavily loaded machines and pallets. This information is used for work station contention resolution. Further contention is resolved by whether the work station is currently idle and then 'first come first serve'.

This system tries to use as much real time information on job criticality and loading to schedule jobs. The complexity of the algorithm adopted allows it be ranked as a small expert system. Decomposing the scheduling algorithm to 'if-then' rules results in 20

rules in Appendix B. The concept of expert scheduling system is very vague and not necessarily useful. The three scheduling system cases can all be claimed to be knowledge based systems of different levels of knowledge and complexity. It is believed that a more methodological structure in the design of decision making is more appropriate than the potential rule conflicts in a free style expert rules elicitation.

6.5 MATCHING SCHEDULING SYSTEM TO PRODUCTION

The three cases illustrate the effect of tailoring the control system to achieve different production requirements with the same machinery hardware configuration. The inherent flexibility of the machinery can be better utilised with a control system that makes use of the production information. The human shop supervisor can deploy different strategies to cope with different changes, but is easily overwhelmed by the amount of information to digest. Human schedulers apply simple rules judiciously to achieve good results. A control system that is so designed can capture the essence of the system situation and work correspondingly. The difficulty is in the exact understanding of the system dynamics. Current scheduling research does not provide sufficient definite guidelines for optimal design of systems.

Optimality of these systems designed for different situations is difficult to prove, as there are no single benchmarks to test them against. A test on versatility for the different designs to cope with a range of situations is potentially useful, if the expected system operating range is known. However, it is not realistic to expect all systems to be designed to cater for all situations, the correct mix in design is an art as much as a science in its deterministic knowledge form. The illustrations here used conventional scheduling results in the design of control systems. The solution to the problems can equally use installation specific knowledge when they are known. The line between an 'expert system' and a conventional system is very thin. Systems designed with local installation knowledge and scheduling research concepts promise the best solutions.

CHAPTER 7
CONTROL ARCHITECTURE IMPLEMENTATION

7.1 CONTROL ARCHITECTURE DESIGN

7.1.1 Control core

To verify the design architecture and test the scheduling systems designed, Chameleon was re-worked to support the new system.

The simulation kernel in Chameleon acts as the control core to provide the interface and simulates the manufacturing system. The basic simulation kernel is enhanced with the definition of different modules for simulation support and the user interface enhanced to the new requirement [Fig.31]. The peer-to-peer communication queue structure between logic modules is stripped off the simulation kernel. The control of how different scheduling modules exchange information is absorbed into the scheduling decision structure. The kernel only performs simulation event processing and clock advance [Listing.1]. This provides the active time information for scheduling.

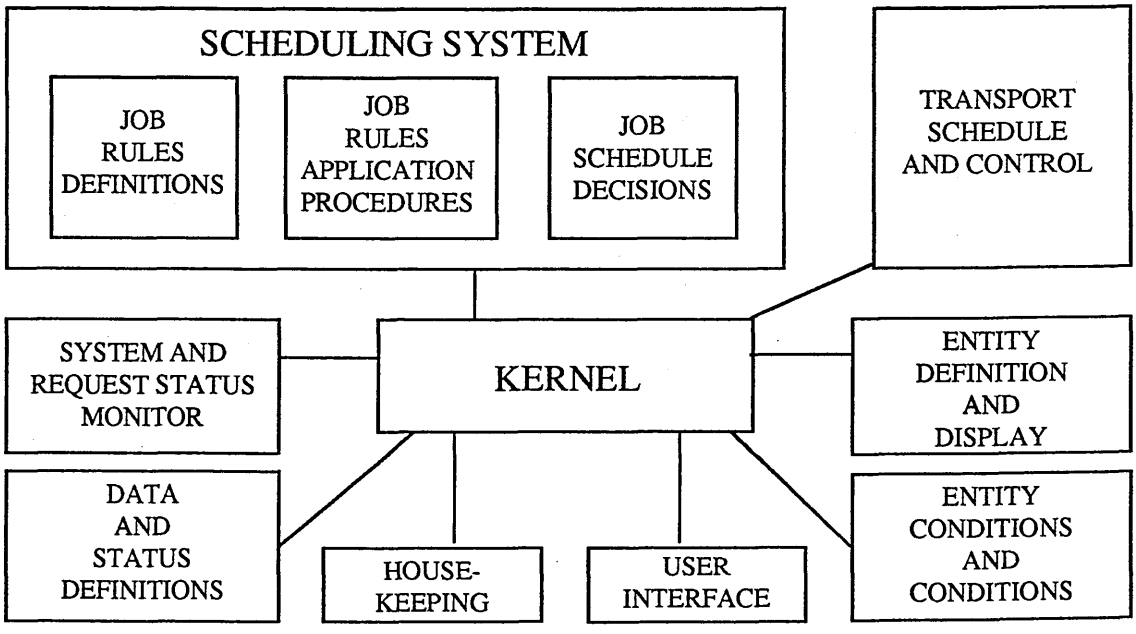


Fig 31. Improved Chameleon structure

```
/* Chameleon kernel */
kernel :-
     display_message,
     time(Real_start_time),
     assert(real_start_time(Real_start_time)),
     repeat,   /* main */
     [!
     user_interface,
     check_queue_event(Queue, Event, Time),
     advance_simulation_clock(Time),
     execute(Queue, Event, Time),
     system_monitor,
     regenerate_system
     !],
     check_simulation_end(Time_now),
     end_simulation(Time_now).
```

Listing 1. Chameleon kernel

7.1.2 Knowledge based job scheduling modules

The database is re-organised to support the different databases for passive data. Most of the data are updated with the execution of event actions. Some of the data for planning are updated by the scheduling system.

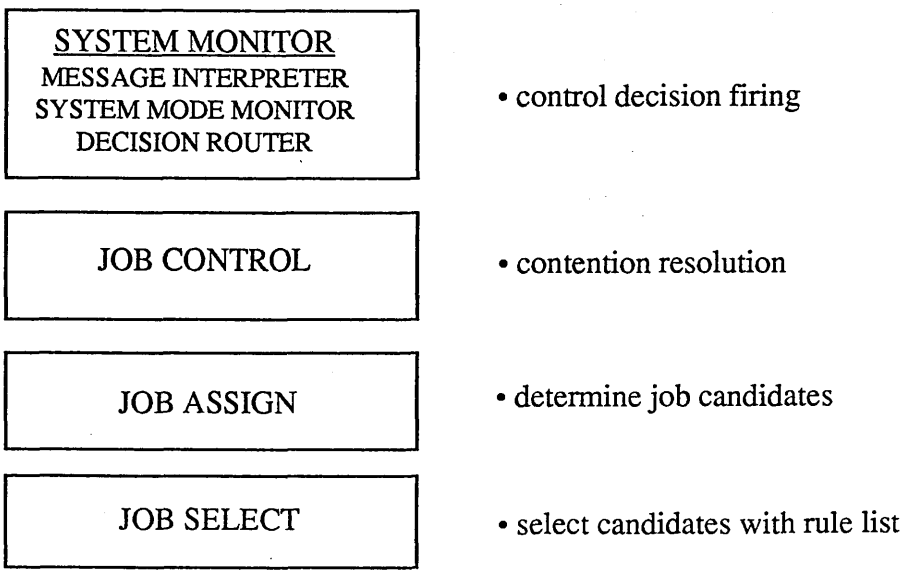


Fig 32. Scheduling modules hierarchy

The scheduling system consisted of a hierarchical structure to achieve the general purpose scheduling system model [Fig.32].

The top level monitor of the system is built up of three parts, a message interpreter, a system monitor and a decision routing function. The message interpreter acknowledges

```
/* message interpreter */
message_acknowledge:-
     retract(attention(monitor,ready,[cart,1],Time)),
     replace_wks_req([cart,1],ready),
     message_acknowledge.
message_acknowledge:-
     retract(attention(monitor,blocked,[cart,1],Time)),
     replace_wks_req([cart,1],blocked),
     message_acknowledge.
message_acknowledge:-
     retract(attention(monitor,ready,[[Wks,Wks_x],Fix],Time)),
     set_wks_req([Wks,Wks_x],ready),
     /* check system blockage */
     clear_blocked(Fix),
     message_acknowledge.
message_acknowledge:-
     retract(attention(monitor,blocked,Wks,Time)),
     set_request_status(Wks),
     message_acknowledge.
message_acknowledge:-
     retract(attention(monitor,buffer_cleared,[mc,Mc_x],Time)),
     check_wks([mc,Mc_x]),
     message_acknowledge.
message_acknowledge:-
     retract(attention(monitor,ready,Wks,Time)),
     check_wks(Wks),
     check_cart_status,
     message_acknowledge.
message_acknowledge.
```

Listing 2. Message interpreter

all event messages and updates the necessary state data for scheduling [Listing.2]. These messages are used as triggers to activate other decision making functions. Statements planted in the interpreter issue the triggers for the decision router. To achieve a clear representation of the job scheduling states, the work station events are interpreted

to maintain the request states of the work stations. The request states are :

ready — when the work station is waiting for jobs;

spare buffer — when work station is working but has unfilled buffers;

blocked — when there are no waiting jobs in the system that the work station can process; and

unblocked — when there are no waiting jobs in the system for the work station, but there are jobs being processed by other work stations that will be available when current operations are finished.

The system mode monitor checks the operation states of the system. In the model built, this checks the loading of the guided vehicle [Listing.3]. The decision routing function reads the trigger from the message interpreter. In situation that a job decision is needed, the job control module is fired.

```
/* system mode monitor */
system_mode_check:-
     ifthenelse(cart_limiting,
          change_job_rule([quick,spt]),
          (data(sim_mode(job,Standing_mode)),
           change_job_rule(Standing_mode))).
```

Listing 3. System mode monitor

The job control module performs [Listing.4] decision selection and identifies the machine or job decision according to the contention resolution mechanism specified. The functions to make the decisions are then called. Contention arbitration can be complex or simple.

```
/* job control module */
job_schedule(Sim_time,[Wks,waiting]) :-
     /* workstation idle */
     /* feed job as soon as possible */
     decide_rule_list(waiting,Rule_list),
     job_assign(Wks,Fix,Wp,Op,waiting,Rule_list),
     sim_key(J,Sim_time),
     schedule_job(waiting,Wks,Fix,Wp,Op,J,Sim_time,Sim_time).
```

Listing 4. Job control module (to be continued)

```
job_schedule(Sim_time,[Mc,buffer]) :-
     /* fill spare buffer */
     /* earliest ready */

     [! wks_free(Mc,J,Time_free) !],
     decide_rule_list(spare_buffer,Rule_list),
     job_assign(Mc,Fix,Wp,Op,spare_buffer,Rule_list),
     schedule_job(spare_buffer,Mc,Fix,Wp,Op,J,Time_free,Sim_time).
job_schedule(Sim_time,_).
```

Listing 4. Job control module

The job assign module [Listing.5] performs the decision specified by the control module using the job selection procedure specified for the particular machine. The location of a suitable pallet on the buffer is checked for immediate assignment to speed operations. Procedures to handle cases where no assignment is possible is also handled here.

```
/* job assignment  */
job_assign(Wks,Fix,Wp,Op,_,Rule_list) :-
     fixture_table(Wks,Fix),
     job_candidate(Wks,Fix,Wp,Op).
job_assign([mc,Mc_x],Fix,Wp,Op,_,Rule_list) :-
     fixture_dock([mc,Mc_x],Fix),
     job_candidate([mc,Mc_x],Fix,Wp,Op).
job_assign(Wks,Fix,Wp,Op,_,Rule_list) :-
     candidate_jobs(Wks,Job_list),!,
     job_select(Rule_list,Wks,Job_list,Fix,Wp,Op).
job_assign(Wks,Fix,Wp,Op,_,Rule_list) :-
     /* fail clause to retry other workstations */
     sim_time(Sim_time,_),
     assert(attention(monitor,blocked,Wks,Sim_time)),!,
     fail.
```

Listing 5. Job assign module

The job selection procedure is not a particular rule but a sequence of rules working like a sequence of filters [Listing.6]. If there are more than one job that can satisfy the first criteria, the next criteria is used to reduce them. If the final criteria still cannot screen out a unique solution, the first in the list is used. For any filters, if there are no jobs that satisfy the criteria, the whole set is passed along. The common approach of assigning weights to the different criteria to form a combined rating is not used. This avoids the difficulties in the determination of arbitrary relative weights and is intuitively easier to understand. This forfeits the ability to balance the importance of several criteria and offset one against another but is considered more effective if a clear immediate goal can be identified.

```
/* job select module */
job_select([],[Wks,Wks_x],[[Fix,Wp,Op]|Job_list],Fix,Wp,Op).

job_select(Rule_list,[Wks,Wks_x],[[Fix,Wp,Op]],Fix,Wp,Op).

job_select([Rule|Rule_list],[Wks,Wks_x],Job_list,F,W,O) :-
     job_optimise(Rule,Wks,Wks_x,Job_list,Select_list),
     job_select(Rule_list,[Wks,Wks_x],Select_list,F,W,O).
```

Listing 6. Job select module

A commonly used set of job decision rules are coded in a rule set and available for call by the job select module [Listing.7]. The repertoire of selection criteria is rich. The common job scheduling criteria of next operation processing times, remaining processing times, remaining number of operations, due date, slack times are all present. 'Unblock' is an important rule that employs local knowledge of the system, the process routing of jobs, the tooling of machines and the dynamic current production states, this locates jobs that can free machines waiting for jobs. Another rule that employs local knowledge calculates the distance of the transport cart to the buffer position, this handles the dynamic requirement of 'as soon as possible'. Two other criteria are coded to cater for the requirement of batch type production, they are the remaining number of parts in the batch and remaining workload in a batch. The codes for associating parts to form a set is specific to the particular association, the coding of which is trivial once the association is known.

```
/* job selection rules */

/* scan along buffer order from 1 to 15 */
job_optimise(buffer,Wks,Wks_x,[[Fix,Wp,Op]|T],[[Fix,Wp,Op]]).

/* shortest next operation processing time */
job_optimise(spt,Wks,Wks_x,Job_list,Select_list) :-
     filter_smallest_list(pt,Job_list,[],9e99,Select_list).

/* largest next operation processing time */
job_optimise(lpt,Wks,Wks_x,Job_list,Select_list) :-
     filter_largest_list(pt,Job_list,[],0,Select_list).

/* shortest remaining processing time on fixture */
job_optimise(srpt,Wks,Wks_x,Job_list,Select_list) :-
     filter_smallest_list(rpt,Job_list,[],9e99,Select_list).

/* largest remaining processing time on fixture */
job_optimise(lrpt,Wks,Wks_x,Job_list,Select_list) :-
     filter_largest_list(rpt,Job_list,[],0,Select_list).

/* smallest number of remaining operations */
job_optimise(srop,Wks,Wks_x,Job_list,Select_list) :-
     filter_smallest_list(rop,Job_list,[],9999,Select_list).

/* largest number of remaining operations */
job_optimise(lrop,Wks,Wks_x,Job_list,Select_list) :-
     filter_largest_list(rop,Job_list,[],0,Select_list).

/* smallest number of remaining parts in po batch */
job_optimise(srbq,Wks,Wks_x,Job_list,Select_list) :-
     filter_smallest_list(rbq,Job_list,[],9e99,Select_list).

/* largest number of remaining parts in po batch */
job_optimise(lrbq,Wks,Wks_x,Job_list,Select_list) :-
     filter_largest_list(rbq,Job_list,[],0,Select_list).

/* smallest number of remaining processing in po batch */
job_optimise(srbp,Wks,Wks_x,Job_list,Select_list) :-
     filter_smallest_list(rbp,Job_list,[],9e99,Select_list).
```

Listing 7. Job selection rules (to be continued)

```
/* largest number of remaining processing in po batch */
job_optimise(lrbp,Wks,Wks_x,Job_list,Select_list) :-
     filter_largest_list(rbp,Job_list,[],0,Select_list).

/* shortest route for cart to pick up */
job_optimise(quick,Wks,Wks_x,Job_list,Select_list) :-
     filter_smallest_list(quick,Job_list,[],9e99,Select_list).

/* smallest slack time for fixture */
job_optimise(slack,Wks,Wks_x,Job_list,Select_list) :-
     filter_smallest_list(slack,Job_list,[],9e99,Select_list).

/* job that can unblock specified blocked workstation */
job_optimise([unblock,Blk,Blk_x],Wks,Wks_x,Job_list,Select_list) :-
     filter_blocked_list([Blk,Blk_x],Job_list,Select_list).

/* job that can unblock some blocked workstation */
job_optimise(unblock,Wks,Wks_x,Job_list,Select_list) :-
     find_blocked(Blk_list),
     filter_blocked_list(Blk_list,Job_list,Select_list).
```

Listing 7. Job selection rules

The hierarchical structure allows systems of different complexity be built in the simulator. The different modules are easily replaceable. To ensure compatibility between different modules, the messages or triggers sent from one to another are standardised. The format of these messages in Prolog form adopted are:

messages from event to message interpreter :

attention(monitor,Message,Parameters,Time),

typical messages are ready, buffer_cleared, blocked; and parameters are the work stations and pallets involved.

messages from interpreter to decision router :

trigger(decision,Message,Parameter,Time),

typical messages are schedule; parameters and time repeats the message from events.

messages from the decision router to the job contention module are issued as pending logic events in the event queue :

event(System_time,job_control,[job,schedule],Parameter,Time).

messages from job contention to job decision are direct Prolog function call :

job_assign(Wks,Fix,Wp,Op,Wks_request,Rule_list),

this call requests the job decision module to return an appropriate fixture and its associated workplan and operation numbers for the work station designated according to the stations request state and current rule list.

The design of this structure is influenced by the need to trim each decision within the limited working memory of the Prolog used. The design encompasses all the necessary information transfer between the decision modules.

To illustrate this structure, the cases for the simple production shop despatch rules and the complex decision rules are described in Section 7.2.

7.1.3 Transport scheduling module

The decisions of job scheduling modules depend on the rail cart to execute. Thus the decision module of the transport system is equally important. During certain phases of simulation, the cart becomes the bottleneck in the system and the job scheduling decisions do not make sense any more. Scheduling of carts in the system is an order of magnitude more difficult than jobs. This is because the processing time element depends on the position of the cart, which is dynamic and can only be predicted at great computing cost. To allow the model to concentrate on the job scheduling decision systems, the transport scheduling system designed to support the research is simplified. Requests from job scheduling for transport can come in two classes. A 'first come first serve' class will execute the transport requests sequentially. Critical requests can be assigned 'as soon as possible' status and the cart services these requests as soon as the current transport move is completed. When the cart is idle, there are procedures that can move the pallets closer to their next operations machine.

7.2 CODING SCHEDULING MODULES

7.2.1 Simple production shop despatch scheduling

The message interpreter [Listing.8] of the simple despatch system triggers the decision router on receiving the message of work stations ready, for this represents the availability of work stations and new operations.

```
/* message interpreter for simple scheduling system */
message_acknowledge:-
     retract(attention(monitor,ready,[cart,1],Time)),
     replace_wks_req([cart,1],ready),
     message_acknowledge.
message_acknowledge:-
     retract(attention(monitor,blocked,[cart,1],Time)),
     replace_wks_req([cart,1],blocked),
     message_acknowledge.
message_acknowledge:-
     retract(attention(monitor,ready,[[Wks,Wks_x],Fix],Time)),
     set_wks_req([Wks,Wks_x],ready),
     /* check system blockage */
     clear_blocked(Fix),
     ifthen(wks_blocked([cart,1],_),check_cart_status),
     set_trigger(job,[Wks,Wks_x]),
     message_acknowledge.
message_acknowledge:-
     retract(attention(monitor,blocked,Wks,Time)),
     set_request_status(Wks),
     message_acknowledge.
message_acknowledge:-
     retract(attention(monitor,buffer_cleared,[mc,Mc_x],Time)),
     check_wks([mc,Mc_x]),
     set_trigger(job,[mc,Mc_x]),
     message_acknowledge.
message_acknowledge:-
     retract(attention(monitor,ready,Wks,Time)),
     check_wks(Wks),
     check_cart_status,
     set_trigger(job,Wks),
     message_acknowledge.
message_acknowledge.
```

Listing 8. Despatch shop message interpreter

The decision router [Listing.9] passes on the parameters to the job control module.

```
/* decision router for simple shop scheduling */
invoke_logic:-
     /* bypass logic invoke */
     retract(bypass_logic(T)).
invoke_logic:-
     /* logic event to be processed */
     /* no need to invoke another logic call */
     data(pending_event(logic,_)).
invoke_logic:-
     /* machine buffer choked */
     /* set cart to clear   */
     wks_choked(Wks,Fix),
     sim_time(Sim_time,J),
     set_pending_time(Sim_time),
     add_data(pending_event(logic,event(J,transport_control,
            [pallet,clear],[Wks,Fix],Sim_time))).
invoke_logic:-
     /* wks ready for next assignment */
     retract(trigger(decision,[wks,ready],[Wks],Time)),
     sim_time(Sim_time,J),
     set_pending_time(Sim_time),
     add_data(pending_event(logic,event(J,job_control,
            [job,schedule],[Wks,waiting],Sim_time))).
invoke_logic:-
     /* spare buffer ready for next assignment */
     retract(trigger(decision,[buffer,ready],[Wks],Time)),
     sim_time(Sim_time,J),
     set_pending_time(Sim_time),
     add_data(pending_event(logic,event(J,job_control,
            [job,schedule],[Wks,buffer],Sim_time))).
invoke_logic.
```

Listing 9. Despatch shop decision router

The job control module [Listing.10] receives the request states of work stations and uses the algorithms matching the request case, supplying a job when possible. The job selection rule is the same for both and is passed to the job selection procedure. All outstanding requests that can be satisfied are assigned.

```
/* job control functions for simple shop scheduling */

job_schedule(Sim_time,[Wks,waiting]) :-
     /* workstation idle */
     /* feed job as soon as possible */
     job_assign(Wks,Fix,Wp,Op,waiting,[spt]),
     sim_key(J,Sim_time),
     schedule_job(waiting,Wks,Fix,Wp,Op,J,Sim_time,Sim_time).
job_schedule(Sim_time,[Mc,buffer]) :-
     /* fill spare buffer */
     /* earliest ready */
     [! wks_free(Mc,J,Time_free) !],
     job_assign(Mc,Fix,Wp,Op,spare_buffer,[spt]),
     schedule_job(spare_buffer,Mc,Fix,Wp,Op,J,Time_free,Sim_time).
job_schedule(Sim_time,_) :-
     /* no assignment available */
     /* check system integrity  */
     /* set cart optimisation   */
     find_q_len(transport,_,1),
     sim_key(J,Sim_time),
     set_pending_time(Sim_time),
     add_data(pending_event(logic,event(J,transport_control,
             [pallet,optimise],[],Sim_time))).
job_schedule(Sim_time,_).
```

Listing 10. Despatch shop job control

The job assign [Listing.11] procedure is purely data processing to find the job that can satisfy the control request.

```
/* job selection   */
job_assign(Wks,Fix,Wp,Op,_,[spt]) :-
     fixture_table(Wks,Fix),
     job_candidate(Wks,Fix,Wp,Op).
job_assign([mc,Mc_x],Fix,Wp,Op,_,[spt]) :-
     fixture_dock([mc,Mc_x],Fix),
     job_candidate([mc,Mc_x],Fix,Wp,Op).
```

Listing 11. Despatch shop job assign (to be continued)

```
job_assign(Wks,Fix,Wp,Op,_,[spt]) :-
     candidate_jobs(Wks,[spt]),!,
     job_select([spt],Wks,Job_list,Fix,Wp,Op).
job_assign(Wks,Fix,Wp,Op,_,[spt]) :-
     /* fail clause to retry other workstations */
     sim_time(Sim_time,_),
     assert(attention(monitor,blocked,Wks,Sim_time)),!,
     fail.
```

Listing 11. Despatch shop job assign

7.2.2 Goal identification adaptive scheduling

The message interpreter [Listing.12] of the dynamic despatch system triggers the decision router on receiving the message of cart ready. This signals that job assignment can be effected immediately. This is the only timing that allows the job scheduling system to take reasonable account of the state of loading of the cart.

```
/* message interpreter for goal identification */
message_acknowledge:-
     retract(attention(monitor,ready,[cart,1],Time)),
     replace_wks_req([cart,1],ready),
     set_trigger(transport,[cart,1]),
     message_acknowledge.
message_acknowledge:-
     retract(attention(monitor,blocked,[cart,1],Time)),
     replace_wks_req([cart,1],blocked),
     message_acknowledge.
message_acknowledge:-
     retract(attention(monitor,ready,[[Wks,Wks_x],Fix],Time)),
     set_wks_req([Wks,Wks_x],ready),
     /* check system blockage */
     clear_blocked(Fix),
     ifthen(wks_blocked([cart,1],_),check_cart_status),
     set_trigger(transport,[cart,1]),
     message_acknowledge.
```

Listing 12. Goal scheduling message interpreter (to be continued)

```
message_acknowledge:-
     retract(attention(monitor,blocked,Wks,Time)),
     set_request_status(Wks),
     message_acknowledge.
message_acknowledge:-
     retract(attention(monitor,buffer_cleared,[mc,Mc_x],Time)),
     check_wks([mc,Mc_x]),
     message_acknowledge.
message_acknowledge:-
     retract(attention(monitor,ready,Wks,Time)),
     check_wks(Wks),
     check_cart_status,
     set_trigger(transport,[cart,1]),
     message_acknowledge.
message_acknowledge.
```

Listing 12. Goal scheduling message interpreter

The decision router [Listing.13] invokes the job control module. Parameter passing is not important in this scheme as there is only one cart in the system to act as trigger.

```
invoke_logic:-
     /* bypass logic invoke */
     retract(bypass_logic(T)).
invoke_logic:-
     /* logic event to be processed */
     /* no need to invoke another logic call */
     data(pending_event(logic,_)).
invoke_logic:-
     /* machine buffer choked */
     /* set cart to clear   */
     wks_choked(Wks,Fix),
     sim_time(Sim_time,J),
     set_pending_time(Sim_time),
     add_data(pending_event(logic,event(J,transport_control,
            [pallet,clear],[Wks,Fix],Sim_time))).
```

Listing 13. Goal scheduling decision router(to be continued)

```
invoke_logic:-
     /* cart is scheduled for work within 6 minutes */
     [!
     sim_time(Sim_time,_),
     searchq(transport,_,event(_,Cart,[deliver,start],
          [_,_,before,Eta_time],_)),
     Time_gap is Eta_time - Sim_time
     !],
     ifthen(Time_gap @< 120.0,
          retract(trigger(decision,[cart,ready],[Cart],Time))),
     fail.
invoke_logic:-
     /* wks ready for next assignment */
     retract(trigger(decision,[cart,ready],[Cart],Time)),
     sim_time(Sim_time,J),
     set_pending_time(Sim_time),
     add_data(pending_event(logic,event(J,job_control,
             [job,schedule],[Cart],Sim_time))).
invoke_logic.
```

Listing 13. Goal scheduling decision router

There are six levels of decisions in the job control module [Listing.14] to check the state of the system and make corresponding decisions. This particular strategy is called 'goal identification' [147] as it attempts to identify the immediate system subgoal to make the corresponding decision. The job and work station criticality is assumed to be available from the tool and fixture planning system.

Case 1 : there are impending critical operations,

subgaol — get this job done as soon as possible,

action — find a machine that can perform this operation and schedule this job.

Case 2 : a critical machine is blocked, it cannot work because there are no operations that it can perform,

subgoal — free critical machine as soon as possible,

action — find an available machine, amongst the jobs that can unblock the critical machine find the one that has the shortest processing time in the unblock operation and can be delivered to the unblocking machine as soon as possible.

```
/* job control functions for goal identification*/

job_schedule(Sim_time,_) :-
     /* waiting critical job */
     waiting_critical_op([Fix,Wp,Op],Est,Wks_list),
     wks_available(Wks,Wks_req,T),
     member(Wks,Wks_list),
     sim_key(J,Est),
     schedule_job(Wks_req,Wks,Fix,Wp,Op,J,Est,Sim_time).
job_schedule(Sim_time,_) :-
     /* heavily loaded workstation became blocked */
     /* feed job as soon as possible */
     critical_wks(Blk),
     wks_blocked(Blk,Time),
     /* wks that can unblock others */
     wks_available(Wks,Wks_req,T),
     /* pick possible unblock job */
     setof([Fix,Wp,Op],
          (job_candidate(Wks,Fix,Wp,Op),
           capable_next_op(Blk,Fix)),
          Job_list),
     /* select quickest job and quickest cart */
     job_select([spt,quick],Wks,Job_list,Fix,Wp,Op),
     sim_key(J,Sim_time),
     schedule_job(Wks_req,Wks,Fix,Wp,Op,J,Sim_time,Sim_time).
job_schedule(Sim_time,_) :-
     /* heavily loaded workstation goes idle */
     /* feed job as soon as possible */
     critical_wks(Wks),
     wks_waiting(Wks,Ready_time),
     job_assign(Wks,Fix,Wp,Op,[quick]),
     sim_key(J,Sim_time),
     schedule_job(waiting,Wks,Fix,Wp,Op,J,Sim_time,Sim_time).

job_schedule(Sim_time,_) :-
     /* workstation idle */
     /* feed job as soon as possible */
     wks_waiting(Wks,Ready_time),
     decide_rule_list(waiting,Rule_list),
     job_assign(Wks,Fix,Wp,Op,waiting,Rule_list),
     sim_key(J,Sim_time),
     schedule_job(waiting,Wks,Fix,Wp,Op,J,Sim_time,Sim_time).
```

Listing 14. Goal scheduling job control (to be continued)

```
job_schedule(Sim_time,_) :-
     /* fill spare buffer */
     /* earliest ready */
     /* not fill long job on hand, set at 1 hour */
     Time_extent is 1200,
     Cut_off_time is Sim_time + Time_extent,
     find_spare_buffer(Mc,Ready_time),
     [! wks_free(Mc,J,Time_free) !],
     Time_free @< Cut_off_time,
     decide_rule_list(spare_buffer,Rule_list),
     job_assign(Mc,Fix,Wp,Op,spare_buffer,Rule_list),
     schedule_job(spare_buffer,Mc,Fix,Wp,Op,J,Time_free,Sim_time).
job_schedule(Sim_time,_) :-
     /* no assignment available */
     /* check system integrity  */
     /* set cart optimisation   */
     sim_key(J,Sim_time),
     set_pending_time(Sim_time),
     add_data(pending_event(logic,event(J,transport_control,
             [pallet,optimise],[],Sim_time))).
```

Listing 14. Goal scheduling job control

Case 3 : a critical machine is waiting for a job,

subgoal — keep it busy,

action — find the job that can be delivered to it as soon as possible.

Case 4 : some workstations are waiting for a job,

subgoal — find selected workstation a job,

action — use the standing rule selection procedure to find a job.

Case 5 : some machines have a spare buffer that can take a job that will feed the machine when it finishes the current job, and the machine's current job last less than an hour,

subgoal — find selected machine a job,

action — use standing rule selection procedure to find a job.

Case 6 : no job assignments are needed or possible,

subgoal — set rail cart to do something useful,

action — sent the cart to better organise the distribution of the pallets in the

system.

As Prolog uses a depth first search algorithm in execution, the job scheduling logic will first find if Case 1 is true, if so, the action is attempted. If the Case is not true or the action for the case cannot be executed, for example, no job assignment can be found, then the next case will be attempted, and so on. The precedence of the rules defines the order of execution.

The first three cases handle the critical situations of a shop, when critical jobs or resources need attention. A job is considered to be more important as the completion of the job is seen as the system objective. If critical machine utilisation is more important, simply swapping the order of the codes will reflect the change in importance. In fact, the order of the cases can be organised to be determined from a standing order of precedence which changes with system states, but this renders the program less readable for this illustration.

Cases 4 and 5 are the cases of normal operation. A different criteria of job selection is used depending on the situation of call, the current system status and the overall system objective. In normal operation situations of Cases 4 and 5, the selection of action to satisfy the same subgoal is referred to a rule decision procedure [Listing.15]. This forms the second tier of goal identification. The underlying system axiom that "work is production" strives to keep workstations busy at all times. Thus the second tier decision order is in unblocking idle machines, feeding waiting machines and then the stated system mode of operation. The first tier case identification screens out the critical situations where immediate subgoals and actions are identified. The second tier screens

```
/* Rule decision procedure for goal identification */
decide_rule_list(waiting,[unblock,spt,quick]) :-
     wks_blocked(Wks,_),
     Wks \== [cart,1].
decide_rule_list(waiting,[quick]).
decide_rule_list(spare_buffer,[unblock,spt,quick]) :-
     wks_blocked(Wks,_),
     Wks \== [cart,1].
decide_rule_list(spare_buffer,Rule_list) :-
     data(job_mode(Rule_list)).
```

Listing 15. Goal scheduling rule list monitor

out the more immediate cases and leaves the non-urgent situations to the third tier system operation modes.

The third tier goal decision sets the basic tone of operation in the system. It guides the production of parts according to manufacturing objectives. The status of the system and the future production requirement is used to analyse the objective. This is the most complicated part in the system for many considerations are needed to arrive at a suitable strategy for operation. At present, the built in logic only switches between the system mode and a quick mode when the transport system is congested. The main mode determination rests with the user who can change it at any time. The complexity of this decision requires a large volume of information, the most significant of which is in planned future production. A high degree of interaction with overall production control system is needed. This is a two way interaction that involves the reception of production orders and the feedback of production status and capacity. An efficient link between factory wide production control and individual manufacturing systems is an important step towards computer integrated manufacturing. At present, this delicate task is performed by the fine judgement of persons with intimate knowledge of the production environment.

The job assign procedure is the same as in Listing 5 performing purely data processing to find the job that can satisfy job control requirements.

7.3 SIMULATION ILLUSTRATIONS

7.3.1 Simulation experiments

7.3.1.1 Production environment sets

To complete the illustration, the two systems are used to produce the same data set of 50 components of 0001 to 0009. Cases for the heaviest loaded machine 4, the shortest processing time component 0005 and longest processing time component 0006 declared critical was also used for the goal identification set. Critical decision rules for goal identification were fine tuned to suit the different characteristics. An extra level of look ahead to assign critical job was designed. This look ahead pre-assigns the next operation for the critical job. For the long processing time job, a hold back of pre-assignment until current processing is within one hour to completion was also used.

These comparisons cannot be used as measures for the systems as they are designed to perform very different production tasks and the experiments do not reflect their target operations environment. These simulations illustrated the difficulties in the design of control systems that can cope with all situations.

7.3.1.2 Simulation results

The results are in Tables 4 and 5. The goal identification algorithm had a better all round performance than the simple dispatch system and started all machines faster.

The critical machine assignment did not affect production results significantly as the interactions between the critical machine and others were limited. The job priority simulations reflected the difficulties in achieving good scheduling control. For critical production of the very short processing time (10min) component 0006, priority despatch when machines were available was not effective, as the waiting time for jobs then being processed was long. The completion time of the order was reduced by one third through the use of look ahead logic to book one operation ahead. This was still more than three times the minimum time possible as the effect of scheduling other longer processing time jobs to improve utilisation in the system was stronger. The pilot experiments with shortest remaining processing time rule that dedicated solely to this component was more effective, but the overall utilisation of the system suffered. The actual degree of criticality to determine the amount of sacrifice is very difficult to quantify.

Critical production of the longest processing time component 0005 had another pattern. The production response of this job was secured easily as the very long processing times precluded other jobs. Because of the mismatch between the 4 hour machine processing time and the 10 minutes load/unload time, the simple look ahead algorithm held the load/unload station for very long periods making load/unload a bottleneck that throttled the production of all other orders. The completion time for the normally quick component 0001 was nearly doubled. A modified rule that performed the book ahead only when current operation was within one hour of completion improved the overall system utilisation significantly without affecting the progress of the critical job.

TABLE 4

PRODUCTION RESULTS FOR DEMONSTRATION EXPERIMENT

This table summarises the elapsed time (hours) before:

first component is completed (upper value); and

all components completed (lower value).

	Order no.								
Despatch Rule	PO_1	PO_2	PO_3	PO_4	PO_5	PO_6	PO_7	PO_8	PO_9
SIMPLE	3	3	4	6	5	5	6	9	2
	39	44	97	151	287	151	*291	146	132
GOAL	2	3	2	6	5	5	6	8	3
	38	44	101	126	*289	126	225	153	136
MC_4	2	3	2	6	5	5	6	8	3
	38	44	101	126	*289	126	225	153	136
PO_6 W	2	2	3	7	6	1	3	10	4
	39	44	101	127	*290	120	225	153	135
PO_6 L	2	3	3	8	7	1	5	10	5
	45	46	104	117	*291	84	247	161	137
PO_5 W	2	3	4	34	5	13	203	205	2
	38	39	85	280	202	257	*308	278	123
PO_5 L	2	4	4	201	5	201	202	204	11
	67	58	96	256	201	256	*302	282	134
PO_5 H	2	3	4	204	5	14	204	211	2
	40	39	85	270	203	247	*309	285	125

* production completion time

SIMPLE : simple production shop

GOAL : goal identification strategy

MC_4 : machine 4 declared critical

PO_6 PO_5 : production orders

W : critical job assignment when available machines are free

L : look ahead critical job assignment, assign next critical operation

H : look ahead assignment when the remaining current operation time is less than one hour

TABLE 5

START UP TIMES FOR DEMONSTRATION EXPERIMENTS

This table summarises workstation start up times :

time when machines first start on the first job (upper value) ; and

time when machine buffer is filled with a waiting job (lower value).

	MC_1	MC_2	MC_3	MC_4	Total	MAN_1	MAN_2
SIMPLE	24'15	37'51	46'6	11'48	120	0'36	1'48
	1h13'36	1h30'15	59'39	14'45	238.3		
GOAL	26'45	13'3	25'24	11'48	77	0'36	1'45
	39'42	46'3	47'57	3h16'42*	133.7*		
MC_4	34'6	13'33	26'0	12'18	86	0'36	1'45
	47'9	44'0	55'51	3'17'6*	147*		
PO_6 W	27'42	13'33	26'21	12'18	80	1'6	2'15
	40'57	57'6	49'21	1'2'45	210.2		
PO_6 L	46'0	13'33	26'30	12'18	98	1'6	2'15
	1h20'15	1h41'36	1h30'3	1h17'6	349		
PO_5 W	26'42	25'21	47'54	11'48	111.8	0'36	1'45
	39'39	46'0	1h15'12	13'0	173.9		
PO_5 L	50'39	25'48	38'9	11'48	126.4	0'36	1'45
	1h19'51	1h8'36	1h30'3	13'0	251.5		
PO_5 W	26'42	25'21	47'54	11'48	111.8	0'36	1'45
	39'39	46'0	1h15'12	13'0	173.9		

SIMPLE : simple production shop
GOAL : goal identification strategy
MC_4 : machine 4 declared critical
PO_6 PO_5 : production orders
W : critical job assignment when available machines are free
L : look ahead critical job assignment, assign next critical operation
H : look ahead assignment when the remaining current operation time is less than one hour

* decision system does not schedule next job until the current is within one hour of completion, completion time for first assigned job is 4h11'48, this time is not summed in the total.

7.3.2 Validation of control structure

The above illustrations from the design procedure to the actual simulation demonstrated the decision model extended from the network of queue model.

The function and power of the design methodology was shown in the development of scheduling systems for three different kinds of production installations.

For the implementation of the scheduling decision model, the crucial factors are the data interface and the scheduling framework. Both of these features are implemented in the Prolog simulator Chameleon. They are used to support the development of the two scheduling systems. The data interface is sufficient to support the two systems without modification, proving its adequacy. The scheduling structure is also validated by the coding of very different systems within the same framework. The enhancement to the model by defining a set of message formats between scheduling decision modules and the maintenance of work stations request states helps greatly in standardising scheduling system design. The provision of a set of standard job selection rules and their application in the form of sequential filters improves the decision flexibility of system design.

CHAPTER 8
DISCUSSION AND CONCLUSION

The production efficiency available now with automated machining facilities and flexible manufacturing systems is ready to be applied in all production shops. However, the current systems are limited by their inflexible control systems. Large flexible manufacturing systems development design work scheduling systems that are particular to the specific installation. For the increasing number of small systems, a better solution is needed. The key is in giving control flexibility to the installation that needs to satisfy the particular production requirements. Conscious study of factors affecting scheduling decisions and their relationship to the shop production environment, production part family and part delivery requirement is necessary. Each shop has its characteristics and variation. To equip FMSs for all these different installations with a single monolithic control would not employ the inherent flexibility of the machining hardware and would not take into account the time varying nature of production demands.

To achieve intelligent control of FMSs, the knowledge based model which highlights and uncouples the scheduling decisions from detailed communications logic is essential. The workflow scheduling control is of interest to the operation of FMSs. The communications is relevant only to the maintenance of system.

The design of a scheduling system is in coordinating the various resources to meet production requirements. The solutions for a control design problem should be in the form of what decisions are needed, when will they be made, what are the choices in decisions and how should the choices be judged. A model of the decisions is the model of the resources. The network of queue model identified the resources and used a simple relationship to link them together. The decision model developed in this thesis expands the classical concept in three directions. First, the resources can have individual identities and characteristics to allow for different production systems. This approach reflects the true flexible nature of the elements in a system and models the human interpretation of scheduling the problem. Secondly, each individual decision can be as complex as required, this gives the scope for knowledge based decision making. Thirdly, the integration of resources is organised as contention resolution decisions. These higher level decisions act as the arbitrator of resources and can employ more information in a more effective manner.

The power of the decision model lies in the flexibility in designing systems of different objectives and complexity but sharing the same core functions. These core functions can then be developed into a standard control core to be supplied with all small flexible manufacturing systems allowing the fine tuning of final scheduling system to installation requirement. These functions are respectively the data structure to provide information for decision making and a design structure for scheduling systems.

The design of flexible manufacturing systems control according to this model is assisted by a methodology which forms a checklist of decisions for all workstations in the system and the considerations for contention resolutions between the work stations.

The successful implementation of this model relies on a complete data interface that extracts all the relevant information from the communications system to feed the scheduling system. A comprehensive study of decision making in flexible manufacturing control results in the definition of this data interface. Equally important is the structure for scheduling decisions implementation. This structure allows core functions to be identified and thus standardised. The relationship between the different modules in the scheduling system have to be defined and the format of information flow between them established. Both of these are illustrated with implementations in the Prolog simulator.

This thesis completed a generic decision making model for the intelligent control of flexible manufacturing systems and developed methodologies for its application and implementation.

CHAPTER 9
RECOMMENDATIONS FOR FURTHER WORK

There are two methods of implementing the universal architecture, depending on the distribution of intelligence between the individual workstations and the supervisory computer of the FMS. The decision engine can reside in the flexible manufacturing system control computer or externally in a decision support workstation.

For systems that make extensive use of local machinery intelligence and have a limited system supervisory computer. It is advisable to use an external workstation for decision support. The required real time portion of the data interface is prepared by the system computer and updates the decision support system. The decision workstation holds the non-real time information and the knowledge and databases for decisions. This separated approach allows great flexibility in the choice of control model, decision workstation hardware and software. Depending on system requirements, the workstation can be a simple computer box like a personal computer or an engineering workstation using artificial intelligence software integrated with human interactive control and the necessary graphics equipment. This two box approach can be easily retrofitted to existing flexible manufacturing systems.

In systems where the system computer has the necessary computing power, the decision engine can be implemented in software and run concurrently with the low level control functions. This is a less desirable option as the tools available to the system designer are limited to those that are available in the current computer. Also, the development and testing of decision system interferes with the actual operations of the cell. The fine tuning of the system becomes a more difficult task.

Though the scheduling framework is in place for the mass introduction of flexible manufacturing systems, the exact understanding between the interactions of the different scheduling decisions are not precisely known. This results in significant intuitive work in the design of scheduling systems for particular sets of production requirements. Further work in scheduling theory using the proposed decision model should establish more solid foundation for the specification of scheduling systems. The effect of machine loadings and the use of loading information for scheduling is particularly important.

REFERENCES

Chapter 2

Development history

1. Williamson D T N, "The pattern of batch manufacture and its influence on machine tool design", Proc IMechE, V182 Pt1, 1967.

2. Sackett P J, Beddis M, "Developing an Approach to AMT", Proc 3rd European Conf on Automated Manufacturing, May 1985, Birmingham, UK.

3. Carter C F, "Trends in machine tool development and application", Proc 2nd Int Conf on Product Development and Manufacturing Technology, 1972.

4. Williamson D T N, "System 24 - a new concept of manufacture", Proc 8th Int Conf on Machine Tool Design and Research, 1967.

5. Williamson D T N, "New wave in manufacturing", American Machinist, Special Report 607, Sept 11 1967.

6. Jablonowski J, "Aiming for flexibility", American Machinist, Special Report 720, March 1980.

7. Hatvany J(ed), "World survey of CAM", Butterworths, 1983.

8. Hartley J, "FMS at Work", IFS(Publications), 1984.

9. Jablonowski J, "Re-examining FMSs", American Machinist, Special Report 774, March 1985.

10. Hollingum J, "Japan's industry puts its money into FMS applications", FMS Magazine, January 1983.

11. Hartley J, "FMS dominant at Osaka", FMS Magazine, January 1983.

12. Kearney & Trecker Marwin Limited, "KTM Manufacturing Systems", KTM sales literature.

Classification

13. The Charles Stark Draper Laboratory, Inc, Flexible Manufacturing Systems Handbook, Noyes Publications, 1984.

14. Mortimer J(Ed), "The FMS Report", Ingersoll Engineers,1982.

15. Browne J, Dubois D, Rathmill K, Sethi S P, Stecke K E, "Classification of flexible manufacturing systems", FMS Magazine, April 1984.

16. Warnecke H-J, Steinhilper R(Eds), Flexible Manufacturing Systems, IFS(Pub), 1985.

17. Economic Commission for Europe, Recent Trends in Flexible Manufacturing, United Nations Publications, 1986.

Roles of design and control

18. Marsh J, "Lights out at Longbridge", Computerised Manufacturing, Nov 1987.

19. Kochan A, "Ambitious FMS: assembly as well as machining", FMS Magazine, January 1988.

20. Kochan A, "Flexibility solves problems at Anderson Strathclyde", FMS Magazine, October 1986.

21. Kochan A, "British Aerospace aims sky high", FMS Magazine, April 1985.

22. Kochan A, "BAe ambitions are about to be realised", FMS Magazine, April 1987.

23. Kochan A, "KTM confirms UK leadership", FMS Magazine, April1985.

24. Kochan A, "A model FMS: cooperation pays off for HNH", FMS Magazine, January 1988.

Dynamic objectives

25. Conway R W, Maxwell W L, "Network dispatching by shortest operation discipline", Operations Research, v10, 1962.

26. Yamamoto M, Nof S Y, "Scheduling-rescheduling in the manufacturing operating system environment", Int J Prod Res, v23 n4, 1985.

Control factor - Fixtures

27. Sackett P J, Cooper D J, "Flexible handling and fixturing in manufacture", Proc 4th European Conf on Automated Manufacturing, Birmingham, May 1987.

Control factor - Tool

28. Cooper D J, Sackett P J, "Cutting tool rationalisation An implementation in an advanced system", Proc 2nd Intl Conf on Advances in Manufacturing Technology, Jordanstown, Sept 1985.

29. Cooper D J, Beddis M R, Sackett P J, "Tooling in integrated Manufacture", Proc IMechE Conf on Planning for Automated Manufacture, Coventry, Sept 1986.

Control systems implementation

30. Achatz R, Steinke H, "KAPLAN copes with capacity planning", FMS Magazine, October 1985.

Generic control systems

31. Wildish M, "Money-making machining cells", Machinery and Production Engineering, 16 Oct 1985.

32. Kochan A, "FMS market blossoms but suppliers lack initiative", FMS Magazine, January 1986.

33. Hartley J, "Cells feature strongly at Japan's FMS show", FMS Magazine, January 1986.

34. Hollingum J, "Helping buyers maximise the benefits of FMS", , FMS Magazine, October 1986.

35. Werner F, "Search for a standard cell for small batch production", FMS Maga-

zine, July 1987.

36. Kochan A, "Trend in Europe is towards standard modules", FMS Magazine, July 1988.

37. Hammer H, "Availability, performance and service of FMS", FMS Magazine, October 1987.

38. Foyer P, "Getting started in FMS", Proc 1st Intl Conf on Flexible Manufacturing Systems, Brighton, Oct 1982.

39. Greenwood N R, "FMS - control and communications. The problems and the potential", Proc 5th Intl Conf on Flexible Manufacturing Systems, Stratford-upon-Avon, Nov 1986.

40. Greenwood N R, Implememting Flexible Manufacturing Systems, Macmillan Education, 1988.

41. Hudson P G, Webb S, "An FMS user's design and application of a computer control system", Proc 2nd Intl Conf on Machine Control Systems, Birmingham, May 1987.

42. Taylor I K, Ford M J, "The relationship between numerical control systems, programmable controllers and overall computer control in the development of a cell controller", Proc 1st Intl Conf on Machine Control Systems, Brighton, 1984.

Chapter 3

Job shop scheduling:

43. King J R, Spachis A S, "Scheduling : bibliography & review", Int Journal of Physical Distribution and Materials Management, v10 n3, 1980.

44. Conway R W, Maxwell W L, Miller L W, Theory of Scheduling, Addison-Wesley, 1967.

45. Johnson S M, "Optimal two- and three-stage production schedules with set up times included", Naval Research Logistics Quarterly, v1, 1954.

46. Jackson J R, " An extension of Johnson's results on job lot scheduling", Naval Research Logistics Quarterly, v3, 1956.

47. Parker R G, Rardin R L, "An overview of complexity theory in discrete optimizations: Part I. Concepts", IIE Transactions, v14 n1, 1982.

48. Parker R G, Rardin R L, "An overview of complexity theory in discrete optimizations: Part II. Results and Implications", IIE Transactions, v14 n2, 1982.

49. Gere W S, "Heuristics in job shop scheduling", Management Science, v13 n3, 1966.

50. Panwalkar S S, Iskander W, "A survey of scheduling rules", Operations Research, v25 n1, 1977.

51. Graves S C, "A review of production scheduling", Operations Research, v29 n4, 1981.

52. Blackstone J H, Phillips D T, Hogg G L, "A state-of-the-art survey of dispatching rules for manufacturing job shop operations", Int J Prod Res, v20 n1, 1982.

53. Kiran A S, Smith M L, "Simulation studies in Job Shop Scheduling", Computer & Industrial Engg, v8 n2, 1984.

54. Jackson J R, "Jobshop-like queueing systems", Management Science, v10, 1963.

55. Gordon W J, Newell G F, "Closed queuing systems with exponential servers", Operations Research, v15 n2, 1967.

56. Steudel H J, Pandit S M, Wu S M, "A multiple time series approach to modeling the manufacturing job-shop as a network of queues", Management Science, v24 n4, 1977.

57. Shanthikumar J G, Buzacott J A, "The time spent in a dynamic job shop", European Journal of Operations Research, v17, 1984.

58. Buzacott J A, Shanthikumar J G, "On approximate queueing models of dynamic

job shops", Management Science, v31 n7, 1985.

59. Panwalkar S S, Dudek R A, Smith M L, "Sequencing research and the industrial scheduling problem", Proc Symposium on theory of scheduling and its applications, Raleigh NC, May 1972.

60. King J R, "The theory-practice gap in job-shop scheduling", Production Engineer, March 1976.

61. Newman P A, "Scheduling in CIM systems", Artificial Intelligence : implications for computer integrated manufacturing (Artificial Intelligence in Industry), Kusiak A (Ed), IFS(Pub), 1988.

Artificial intelligence for scheduling

62. Hunt van D, Artificial Intelligence and Expert System Sourcebook, Chapman & Hall, 1986.

63. Barr A, Feigenbaum E A, Cohen P R, The Handbook of Artificial Intelligence, 3 vols, William Kaufman, 1981.

64. Winston P H, Artificial Intelligence, Addison-Wesley, Los Altos, 2nd Ed, 1984.

65. Feigenbaum E A, McCorduck P, The Fifth Generation, Addison-Wesley, Reading, 1983.

66. Nilsson N J, "Artificial intelligence : engineering, science or slogan?", AI Magazine, v3 n1, 1982.

67. Simmons R, "Semantic networks: their computation and use Simmons R, "Semantic networks: their computation and use for understanding sentences", Computer Models of Thoughts and Language, Schank R, Colby K (Eds), W H Freeman, San Francisco, 1973.

68. Minsky M, "A framework for representing knowledge", The Psychology of Computer Vision, Winston P H (Ed), McGraw-Hill, New York, 1975.

69. Fox M S, "Knowledge representation for decision support", Knowledge Representation for Decision Support Systems, Methlie L B, Sprague R H (Eds), North Holland, 1985.

70. Randall D, King J, "An overview of production systems", Machine Intelligence, v8, Elcock E W, Michie D (Eds), John Wiley & Sons, New York, 1977.

71. Duda R O, Gaschnig J G, "Knowledge-based expert systems come of age", Byte, September 1981.

72. Ernst G, Newell A, GPS: A Case Study in Generality and Problem Solving, Academic Press, New York, 1980.

73. Fikes R E, Nilsson N J, "STRIPS: A new approach to the application of theorem proving to problem solving", Artificial Intelligence, v2 n3&4, 1971.

74. Kowalski R A, "The early years of Logic Programming", Communications of ACM, v31 n1, 1988.

75. Clocksin W F, Mellish C S, Programming in Prolog, Springer-Verlag, New York, 2nd Ed, 1984.

76. Weber D M, Moodie C L, "A knowledge-based system for information management in an automated and integrated manufacturing system", Robotics and Computer-Integrated Manufacturing, v4 n3/4, 1988.

77. Mettrey W, "An assessment of tools for building large knowledge-based systems", AI Magazine, Winter 1987.

78. Stone J, "Commercial AI trends seen at AAAI-87", AI Magazine, Winter 87.

79. Fox M S, Smith S F, "ISIS - a knowledge-based system for factory scheduling", Expert Systems, v1 n1, 1984.

80. Smith S F, Fox M S, Peng Si Ow, "Constructing and maintaining detailed production plans: investigations into the development of knowledge-based factory scheduling systems", AI Magazine 1986.

81. Bensana E, Bel G, Dubois D, "OPAL: a multi-knowledge-based system for industrial job-shop scheduling", Int J Prod Res, v26 n5, 1988.

82. Elleby P, Fargher H E, Addis T R, "A constraint based scheduling system for semi-conductors wafer fabrication", IFIP Conf Workshop on Knowledge Based Production Management Systems, Galway Aug 23-25, 1988.

83. Sauve B, "A contribution to intelligent job shop planning and control", IFIP Conf Workshop on Knowledge Based Production Management Systems, Galway Aug 23-25, 1988.

84. Buchanan B G, "Expert systems : working systems and the research literature", Expert Systems, v3 n1, 1986.

Control structure

85. Nof S Y, Barash M M, Solberg J J, "Operational control of item flow in versatile manufacturing systems", Int J Prod Res, v17 n5, 1979.

86. Stecke K, "A hierarchical approach to production planning in flexible manufacturing systems", Proc 20th Allerton Conf on Communication Control and computing, 1982.

87. Suri R, Whitney C K, "Decision support requirements in flexible manufacturing", J Manufacturing Systems, v3 n1, 1984.

88. Jones A T, McLean C R, "A proposed hierarchical control model for automated manufacturing systems", Journal of Manufacturing Systems, v5 n1, 1986.

89. O'Grady P J, Bao H, Lee K H, "Issues in Intelligent Cell Control for Flexible Manufacturing Systems", Computers in Industry, v9, 1987.

90. Duffie N A, Piper R S, Humphrey B J, Hartwick J P, "Hierarchical and non-hierarchical manufacturing cell control with dynamic part-oriented scheduling", Proc 14th North American Manufacturing Research Conf, 1986.

91. Lewis W C, Barash M M, Solberg J J, "Single queue management of a job shop as implemented by a data flow architecture", Proc 23rd Conf on Machine Tool Design

and Research, 1982.

FMS queuing network models:
92. Solberg J J, "A mathematical model of computerized manufacturing systems", 4th Intl Conf on Production Research, Toyko, Aug 1977.

93. Stecke K E, Solberg J J, "The optimality of unbalanced workloads and machine group sizes for flexible manufacturing systems", Working Paper 290, The University of Michigan, 1982.

94. Stecke K E, Morin T L, "The optimality of balancing workloads in certain types of flexible manufacturing systems", European J of Operational Research, v20, 1985.

95. Suri R, Hildebrant R R, "Modelling flexible manufacturing systems using mean-value analysis", J Manufacturing Systems, v3 n1, 1984.

96. Hildebrant R R, "Scheduling flexible machining systems using mean value analysis", Proc IEEE Conf on Design & Control, Albuquerque, New Mexico, 1980.

97. Buzacott J A, "Modelling manufacturing systems", Robotics & Buzacott J A, "Modelling manufacturing systems", Robotics & Computer Integrated Manufacturing, v2 n1, 1985.

98. Buzacott J A, Shanthikumar, "Models for understanding flexible manufacturing systems", AIIE Transactions, v12 n4, 1980.

99. Buzacott J A, Yao D D, "Flexible manufacturing systems: a review of analytical models", Management Science, v32 n7, 1986.

100. Buzacott J A, Yao D D, "On queueing network models of flexible manufacturing systems", Queueing Systems, v1 pt1, 1986.

FMS simulation models
101. Lenz J E, Talvage J J, "General computerised manufacturing system simulator(GCMS)", Project Report 7, Optimal Planning of Computerised Manufacturing Systems, Purdue University, 1977.

102. Lenz J E, MAST User's Manual, CMS Research Inc, Oshkosh, Wisconsin, 1980.

103. Browne J, Rathmill K, "The use of simulation modelling as a design tool for FMS", Proc 2nd Intl Conf on Flexible Manufacturing Systems, London, Oct 1983.

104. Crookall J R, "Planning and simulation of FMS", Annals of the CIRP, Vol 34/2/ 1985.

105. ElMaraghy H A, "Simulation and graphical animation of advanced manufacturing systems", Journal of Manufacturing Systems, n1, 1982.

106. Rathmill K, Chan W W, "What simulation can do for FMS design and planning", FMS Magazine, v1 n3, 1983.

107. Carrie A S, Adhami E, "Introducing an FMS by simulation", Proc 2nd Intl Conf on Flexible Manufacturing Systems, Oct 1983.

108. Carrie A S, "The role of simulation in FMS", Flexible Manufacturing Systems: Methods and Studies, A. Kusiak(Ed), Elsevier, 1986.

109. Suri R, Cao X, "Optimization of flexible manufacturing systems using new techniques in discrete event systems", Proc 20th Allerton Conf on Communications, Control and Computing, 1982.

110. Haddock J, "An expert system framework based on a simulation generator", Simulation, v48 n2, 1987.

111. la Commare U, lo Valvo E, Noto la Diega S, "Simulation of FMS : an effective tool in productivity evaluation", Proc IASTED Intl Symposium on Modelling and Simulation, Lugano, 1985.

112. Stecke K E, Solberg J J, "Loading and control policies for a flexible manufacturing system", Int J Prod Res, v19 n5, 1981.

113. Chan T S, Pak H A, "Modelling of a controller for a flexible manufacturing cell", Proceedings of IMechE, v200 nB4, 1986.

114. Chan T S, Pak H A, "Heuristical job allocation in a flexible manufacturing system", Intl J Advanced Manufacturing Technology, v1 n2, 1986.

115. Choi H G, Malstrom E M, "A literature review and analytical methodology for traditional scheduling rules in a flexible manufacturing system", Technical Report, Iowa State University, 1983.

116. Choi R H, Malstrom E M, "Physical simulation of work scheduling rules in a flexible manufacturing system", Proc 8th Annual Conference on Computers and Industrial Engineering, 1986.

117. Weston R H, Sumpter C M, Gascoigne J D, "Distributed Manufacturing Systems", Robotica, v4, 1986.

118. Nelder G P, Analysis of Control Factors in Flexible Manufacture, MSc thesis, Cranfield Institute of Technology, 1987.

Knowledge based scheduling

119. Bullers W I, Nof S Y, Whinston A B, "A logic representation of manufacture control", Proc 6th Int Joint Conf on Artificial Intelligence, Tokyo, 1979.

120. Young R E, Rossi M A, "Towards knowledge-based control of flexible manufacturing systems", IIE Transactions, March 1988.

121. Shaw M J P, Whinston A B, "Automatic planning and flexible scheduling: a knowledge-based approach", Proc IEEE Intl Conf on Robotics & Automation, St Louis, 1985.

122. Shaw M J, "Knowledge-based scheduling in flexible manufacturing systems: An integration of pattern-directed inference and heuristic search", Int J Prod Res, v26 n5, 1988.

123. Subramanyan S, Askin R G, "An expert system approach to scheduling in flexible manufacturing systems", Flexible Manufacturing Systems: Methods and Studies, A. Kusiak(Ed), Elsevier, 1986.

124. Kusiak A, "FMS scheduling: a crucial element in an expert system control struc-

ture", Proc IEEE Intl Conf on Robotics & Automation, 1986.

125. Ben-Arieh D, Moodie C L, "Knowledge based routing and sequencing for discrete part production", Journal of Manufacturing Systems, v6 n4, 1987.

126. Alexander S M, "An expert system for the selection of scheduling rules in a job shop", Computers & Industrial Engineering, v12 n3, 1987.

127. Steffen M S, "A survey of artificial intelligence based scheduling systems", Proc 1986 Fall Industrial Engineering Conference, 1986.

Intelligent cell control

128. Wu S-Y D, Wysk R A, "MPECS - an intelligent flexible machining cell controller", Simulation on CIM and Artificial Intelligence Techniques : Proc European Simulation MultiConference, Vienna, 1987.

129. Gross J R, Ravi Kumar K, "Intelligent control systems", Artificial Intelligence : implications for computer integrated manufacturing (Artificial Intelligence in Industry), Kusiak A (Ed), IFS(Pub), 1988.

130. O'Grady P, Lee K H, "An intelligent cell control system for automated manufacturing", Intl J Prod Res, v26 n5, 1988.

131. Chryssolouris G, Wright K, Pierce J, Cobb W, "Manufacturing systems operation: dispatch rules versus intelligent control", Robotics & Computer-Integrated Manufacturing, vol 4 n3/4, 1988.

132. Browne J, et al, "An approach to the design of Production Activity Control system", presented Seminar on AI Based Production Activity Control before IFIP WG5.7 Conference, 23 Aug 1988.

Chapter 4

Chameleon

133. Shannon R E, "Knowledge based simulation techniques for manufacturing", Intl J Prod Res, v26 n5, 1988.

134. Van del Horst A, "Declarative - non procedural - approach in simulation languages", Simulation on CIM and Artificial Intelligence Techniques : Proc European Simulation MultiConference, Vienna, 1987.

135. Futo I, Szeredi J, "A discrete simulation system based on artificial intelligence methods", Discrete Simulation and Related Fields, Javor A (Ed), North-Holland, 1982.

136. Futo I, "Combined discrete-continuous modeling and problem solving", AI Graphics and Simulation, Society for Computer Simulation, San Diego, 1985.

137. O'Keefe R M, Roach J W, "Artificial intelligence approaches to simulation", J Operations Research Society, v38 n8, 1987.

138. Fan I S, Sackett P J, "A Prolog simulator for interactive flexible manufacturing systems control", Simulation, v50 n6, June 1988.

139. Lesser V R, Erman L D, "A retrospective view of the Hearsay-II architecture", Fifth Int Joint Conf on Artificial Intelligence, Cambridge, MA, 1977.

140. Ammons J C, Govindaraj T, Mitchell C M, "Human-aided scheduling for FMS : a paradigm for human-computer interaction in real time scheduling and control", Proc 2nd ORSA/TIMS Conf on FMS : Operations research models and applications, Aug 1986.

141. Nakamura N, Salvendy G, "An experimental study of human decision making in computer based scheduling of flexible manufacturing systems", Intl J Prod Res, v26 n4, 1988.

142. Sackett P J, Rowlinson P, "CIM technology and the role of the industrial engineer", Proc 3rd Int Conf on Human Factors in Manufacturing, Nov 1986.

Chapter 5

Universal architecture

143. Tocher K D, "Control", Operational Research Quarterly, v21 n2, 1970.

144. J R Jackson, "Networks of waiting lines", Operations Research, v5, 1957.

145. Fan I S, Sackett P J, "An universal architecture for flexible manufacturing cell control", Proc 4th Int Conf on Computer Aided Production Engineering, 20-23 Nov, 1988.

146. Hayes-Roth B, "A blackboard architecture for control", Artificial Intelligence, v26 n3, 1985.

Chapter 6

Scheduling cases

147. Sackett P J, Fan I S, "Goal identification for flexible manufacturing system control", IFIP Conf Workshop on Knowledge Based Production Management Systems, Galway Aug 23-25, 1988.

APPENDIX A
WORK PLAN DETAILS

Work plan : wp_3_0001
Component: comp_0001, 'component no 1',
Fixtures : 100001,100002
Operations :

no		NC	time	Preceding op
1	'load'	[]	10	[]
2	'cut 1'	['1001']	40	[1]
3	'unload'	[]	10	[2]

Work plan : wp_3_0002
Component: comp_000, 'component no ',
Fixtures : 100003,100004,100005
Operations :

no		NC	time	Preceding op
1	'load'	[]	15	[]
2	'cut 1'	['1002']	45	[1]
3	'unload'	[]	10	[2]

Work plan : wp_3_0003
Component: comp_0003, 'component no 3',
Fixtures : 100006,100007
Operations :

no		NC	time	Preceding op
1	'load'	[]	10	[]
2	'cut 1'	['1003']	75	[1]
3	'unload'	[]	10	[2]

Work plan : wp_3_0004
Component: comp_0004, 'component no 4',
Fixtures : 100008
Operations :

no		NC	time	Preceding op
1	'load'	[]	10	[]
2	'cut 1'	['1004']	20	[1]
3	'unload'	[]	10	[2]

Work plan : wp_4_0005
Component: comp_0005, 'component no 5',
Fixtures : 100009,100010
Operations :

no		NC	time	Preceding op
1	'load'	[]	10	[]
2	'cut 1'	['1005']	240	[1]
3	'unload'	[]	10	[2]

Work plan : wp_4_0006
Component: comp_0006, 'component no 6',
Fixtures : 100011
Operations :

no		NC	time	Preceding op
1	'load'	[]	10	[]
2	'cut 1'	['1006']	10	[1]
3	'unload'	[]	10	[2]

Work plan : wp_4_0007
Component: comp_0007, 'component no 7',
Fixtures : 100012
Operations :

no		NC	time	Preceding op
1	'load'	[]	10	[]
2	'cut 1'	['1007']	45	[1]
3	'cut 2'	['1008']	30	[2]
4	'unload'	[]	10	[3]

Work plan : wp_4_0008
Component: comp_0008, 'component no 8',
Fixtures : 100013,100014,100015
Operations :

no		NC	time	Preceding op
1	'load'	[]	10	[]
2	'cut 1'	['1009']	45	[1]
3	'cut 2'	['1010']	30	[2]
4	'unload'	[]	10	[3]

Work plan : wp_4_0009
Component: comp_0009, 'component no 9',
Fixtures : 100015,100016,100017
Operations :

no		NC	time	Preceding op
1	'load'	[]	10	[]
2	'cut 1'	['1011']	20	[1]
3	'cut 2'	['1012']	50	[2]
4	'unload'	[]	10	[3]

Work plan : wp_4_0010
Component: comp_0010, 'component no 10',
Fixtures : 100018,100019,100020
Operations :

no		NC	time	Preceding op
1	'load'	[]	10	[]
2	'cut 1'	['1013']	25	[1]
3	'cut 2'	['1014']	25	[2]
4	'unload'	[]	10	[3]

Work plan : wp_4_0011
Component: comp_0011, 'component no 11',
Fixtures : 100021,100022
Operations :

no		NC	time	Preceding op
1	'load'	[]	10	[]
2	'cut 1'	['1015']	55	[1]
3	'cut 2'	['1016']	25	[2]
4	'unload'	[]	10	[3]

Work plan : wp_4_0012
Component: comp_0012, 'component no 12',
Fixtures : 100023
Operations :

no		NC	time	Preceding op
1	'load'	[]	10	[]
2	'cut 1'	['1017']	45	[1]
3	'cut 2'	['1018']	30	[2]
4	'unload'	[]	10	[3]

Work plan : wp_4_0013
Component: comp_0013, 'component no 13',
Fixtures : 100024,100025
Operations :

no		NC	time	Preceding op
1	'load'	[]	10	[]
2	'cut 1'	['1019']	45	[1]
3	'cut 2'	['1020']	30	[1]
4	'unload'	[]	10	[2,3]

Work plan : wp_4_0014
Component: comp_0014, 'component no 14',
Fixtures : 100026,100027,100028
Operations :

no		NC	time	Preceding op
1	'load'	[]	10	[]
2	'cut 1'	['1021']	60	[1]
3	'cut 2'	['1022']	60	[1]
4	'unload'	[]	10	[2,3]

Work plan : wp_4_0015
Component: comp_0015, 'component no 15',
Fixtures : 100029
Operations :

no		NC	time	Preceding op
1	'load'	[]	15	[]
2	'cut 1'	['1023']	35	[1]
3	'cut 2'	['1024']	60	[1]
4	'unload'	[]	15	[2,3]

Work plan : wp_4_0016
Component: comp_0016, 'component no 16',
Fixtures : 100030,100031
Operations :

no		NC	time	Preceding op
1	'load'	[]	10	[]
2	'cut 1'	['1025']	15	[1]
3	'cut 2'	['1026']	35	[1]
4	'unload'	[]	10	[2,3]

Work plan : wp_4_0017
Component: comp_0017, 'component no 17',
Fixtures : 100032,100033,100034
Operations :

no		NC	time	Preceding op
1	'load'	[]	10	[]
2	'cut 1'	['1027']	40	[1]
3	'cut 2'	['1028']	45	[1]
4	'unload'	[]	10	[2,3]

Work plan : wp_4_0018
Component: comp_0018, 'component no 18',
Fixtures : 100035,100036
Operations :

no		NC	time	Preceding op
1	'load'	[]	10	[]
2	'cut 1'	['1029']	50	[1]
3	'cut 2'	['1030']	30	[1]
4	'unload'	[]	10	[2,3]

Work plan : wp_4_0019
Component: comp_0019, 'component no 19',
Fixtures : 100037
Operations :

no		NC	time	Preceding op
1	'load'	[]	10	[]
2	'cut 1'	['1031']	75	[1]
3	'cut 2'	['1032']	35	[1]
4	'unload'	[]	10	[2,3]

Work plan : wp_4_0020
Component: comp_0020, 'component no 20',
Fixtures : 100038,100039,100040
Operations :

no		NC	time	Preceding op
1	'load'	[]	10	[]
2	'cut 1'	['1033']	40	[1]
3	'cut 2'	['1034']	45	[1]
4	'unload'	[]	10	[2,3]

Work plan : wp_4_0021
Component: comp_0021, 'component no 21',
Fixtures : 100041,100042
Operations :

no		NC	time	Preceding op
1	'load'	[]	10	[]
2	'cut 1'	['1035']	55	[1]
3	'cut 2'	['1036']	40	[1]
4	'unload'	[]	10	[2,3]

Work plan : wp_4_0022
Component: comp_0022, 'component no 22',
Fixtures : 100043,100044,100045
Operations :

no		NC	time	Preceding op
1	'load'	[]	10	[]
2	'cut 1'	['1037']	45	[1]
3	'cut 2'	['1038']	30	[1]
4	'unload'	[]	10	[2,3]

APPENDIX B
RULE FORM OF GOAL IDENTIFICATION DESPATCH

/* RULE 1 */
IF cart is free
THEN start scheduling decision

/* RULE 2 */
IF critical jobs exist
    AND machine available to process jobs
THEN despatch job with top priority

/* RULE 3 */
IF critical machine is blocked
    AND it is not being unblocked
    AND could be unblocked
THEN free blocked critical machine as soon as possible

/* RULE 4 */
IF critical machine is waiting
    AND jobs are available
THEN despatch job that could be reached quickest

/* RULE 5 */
IF machine is waiting
    AND other machines are blocked
    AND some jobs available to unblock this machine
THEN despatch job that can unblock and has the shortest imminent processing time

/* RULE 6 */
IF machine is waiting
    AND no unblocking could be done
THEN despatch job that could be reach quickest

/* RULE 7 */
IF no other despatch operation
    AND spare buffer available
THEN schedule buffer

/* RULE 8 */
IF schedule buffer
    AND machine has more than one hour operations remain
THEN do not fill buffer

/* RULE 9 */
IF schedule buffer
    AND cart is heavily loaded
THEN despatch job that could be reached quickest

/* RULE 10 */
IF schedule buffer
    AND cart is not a limiting factor
THEN despatch job that has the shortest imminment processing time

/* RULE 11 */
IF machine contention
THEN break by criticality

/* RULE 12 */
IF machine contention tie
THEN break by order of request

/* RULE 13 */
IF buffer contention
THEN select machine whose operation will complete earliest

/* RULE 14 */
IF there are more than two cart requests within 5 minutes
THEN the cart is heavily loaded

/* RULE 15 */
IF despatching for quick reach
THEN check work table first

/* RULE 16 */
IF despatching for quick reach
THEN check buffer first

/* RULE 17 */
IF a machine can not find a job to process
THEN it is blocked

/* RULE 18 */
IF a machine is blocked
    AND a job currently processed has a next operation that this machine can work
THEN it is being unblocked

/* RULE 19 */
IF a job is declared critical by configuration program
THEN it is critical

/* RULE 20 */
IF a machine is declared critical by configuration program
THEN it is critical