# DESIGN, IMPLEMENTATION AND SIMULATION OF A HYBRID ARCHITECTURE FOR THE CONTROL OF SCHEDULING ACTIVITIES IN AN FMS

**A thesis presented in partial fulfilment of the requirements**

**for the degree**

**of Master of Technology**
**in Manufacturing and Industrial Technology at**
**Massey University**

**Desmon Chrisman Ginting**

**1995**

# Acknowledgments

I wish to express my sincere thanks and gratitude to Dr. Simon F. Hurley, my chief supervisor for his expert guidance, helpfulness, encouragement and support throughout my study, particularly for his constructive criticism and patient correction of my thesis. Without his help, this work would not have been possible.

My deep thanks and gratitude also to Dr. Saeid Nahavandi, my second supervisor, for his criticism, expert guidance, understanding, enthusiasm and helpfulness in many ways.

My sincere thanks also to the following:

(i) The Indonesian Government that has given me this beneficial opportunity for studying in New Zealand.

(ii) The New Zealand Government, through the Ministry of Foreign Affair and Trade Overseas Development Assistance Program for financing my study at Massey University.

(iii) Michael Butler, my fellow postgraduate student for invaluable discussions at the beginning of this work.

(iv) All staff, technicians and postgraduate students in the Department of Production Technology, Massey University for their friendship and assistance.

Finally, I wish to express my deepest gratitude to my family: Vita and Ephraim, for their love, support, encouragement throughout the completion of my study and for bearing the brunt of having much of my attention diverted from them. I hope I can repay the debt with interest. To them, this thesis is dedicated. I would like also to thank Mum and Dad who never doubted that I would and could complete this Master's degree.

# Abstract

An alternative architecture for the control of scheduling activities in an FMS shop floor, called Hybrid Control Architecture, is being researched at Massey University, Palmerston North. The architecture incorporates the strengths of the four standard FMS control architectures: centralised, hierarchical, modified hierarchical and heterarchical. The objective is to maximise the advantages associated with these existing architectures and minimise their problems.

The major characteristics of the hybrid control architecture are levels of control with a single supervisor, full autonomy for subordinates, full intelligent entities and simplicity in scheduling. The architecture offers prospects of decomposition of the control problems, reduced control system complexity by localising information without eliminating global information, increasing modifiability and possible gradual implementation.

The method used for scheduling jobs is the "hybrid" auction-based scheme wherein the top level (the shop controller) acting as the centralised auctioneer and awards the processing of a task to the work centre with the best bid. The "hybrid" auction-based scheme is a part of a three-level scheduling framework: task selection, bidding function and local scheduling. Such a scheduling scheme provides an opportunity to incorporate different dispatching heuristics to achieve a global goal.

To study the scheduling aspect of the hybrid control architecture, a simulation package serving two functions - the control facility and simulator - is developed. The modelled FMS shop floor and the auction-based sequencing functions of the shop, cell and machine controllers are constructed using a real-time control software. A windows programming language is employed to create the simulator and other control functions of the shop, cell and machine controllers. Such an approach is called SIMCON (Simulation Control) and the main benefit is that there exists a unique opportunity to develop the simulator that is initially used as a simulation tool and, later, as supporting control software in the real situation.

In this research, the selection of tasks to be auctioned is based on "First Come First Auctioned" (FCFA). The bidding function is based on the Earliest Finishing Time (EFT) and the rule used to load a task on a machine is based on FCFS (First Come First Served). Experimental results illustrates the "hybrid" auction-based scheme and verifies the operation of the implemented hybrid control architecture.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# CHAPTER 1

# INTRODUCTION

## 1.1 Research Background

The development and implementation of Cellular Manufacturing (CM) and Flexible Manufacturing Systems (FMSs) have contributed to the ability to both increase the flexibility of manufacturing operations and reduce lead times. This has brought emphasis on the importance of shop floor control systems, since activities must now be tightly controlled to achieve shorter lead times and high system utilisation.

The focal point for any shop floor control system is the scheduling of work orders (Chryssolouris, 1992). The scheduling methods employed depend primarily on the control architecture in use. There are four standard control architectures: (1) centralised, (2) hierarchical, (3) modified hierarchical, and (4) heterarchical (Dilts et al., 1991)[1].

Each of these architectures has its own specific advantages and disadvantages. Such a fact motivated this research by the challenge to construct a control architecture incorporating best elements from these architectures. Since scheduling of work orders is the primary activity of the FMS control system, the design of the proposed control architecture in this research is specifically focused on the scheduling aspect.

---

1. Discussed in Chapter 2.

## 1.2 Research Objectives

The objectives of this research are to:

1. Building on the strengths of the existing control architectures, design and implement a new architecture for the control of scheduling activities in an FMS shop floor.
2. Construct a computer-based simulation to study the scheduling aspect of the new control architecture.

## 1.3 Thesis Structure

This thesis is structured as follows:

1. **Literature Review** (Chapter 2)

   This focuses primarily on cellular manufacturing, flexible manufacturing systems, shop floor control architectures and simulation in manufacturing. The purpose is to provide a background for designing and implementing the proposed control architecture.
2. **Proposed Architecture** (Chapter 3)

   Based on the literature review, the proposed architecture, called the hybrid control architecture, is outlined. Also, the conceptual model for the scheduling system and its functionality is included.
3. **Implementation** (Chapter 4)

   The abstract model of the hybrid control architecture is formulated for evaluation by a computer-based simulation. This includes model implementation and description of the software used.

4. **Experimentation and Analysis** (Chapters 5 and 6)

   Simulation input data, measures of performance used to study the scheduling system of the hybrid control architecture, and the execution of the simulation are covered. The simulation results are interpreted and recommendations made for actual implementation. Areas for further study are also outlined.

As a summary, Chapter 7 highlights important issues concerning the hybrid control architecture and the simulation results obtained.

# CHAPTER 2

# LITERATURE REVIEW

## 2.1 Introduction

As indicated in Chapter 1, of importance to this research are types of manufacturing systems, Cellular Manufacturing, Flexible Manufacturing Systems (FMSs), shop floor control architectures for FMSs, and simulation in manufacturing. This chapter presents a review of these subjects.

## 2.2 Types of Manufacturing Systems

In a broad sense, a manufacturing system may be modelled by $m$ distinct queues in front of $m$ work-stations. Workpieces arrive at the system according to stochastic processes. The arriving workpieces are supplies of raw materials and orders for finished products (Figure 2.1). They enter as inputs to gateway work-stations and from there on, except for last work-stations, the output of each work-station becomes the input for the next work-stations.

Queues at work-stations consist of raw materials, parts, and subassemblies waiting for processing at a given machine. The "implementation box" in Figure 2.1 makes deterministic changes - subtracting or adding workpieces - to the queues. Subtractions correspond to the process of raw materials or the use of intermediate subassemblies to build other subassemblies and finished products.

Organised material flows will enable all work-stations to manufacture workpieces on time and in sufficient quantity. This is the management function which determines the commands which prescribe the material flow in the system (Figure 2.1).

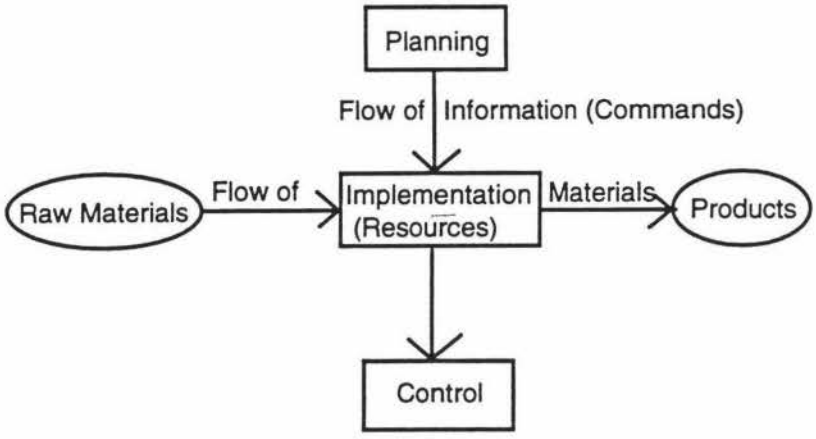


Figure 2.1: Flows concerning manufacturing (Hitomi., 1994)

A manufacturing system can generally be categorised into two areas: the processing area and the assembly area. There are five general approaches, in practice, to structuring the processing area: the project shop, job shop, flow shop, continuous system and cellular system approaches.

### 2.2.1 *Project Shop*

In a project shop, a product's position remains fixed during manufacture. Materials, people, and machines are brought to the product for processing or assembly as needed. For example, the aircraft and shipbuilding industries organise their facilities as project shops.

### 2.2.2 *Job Shop*

In a job shop, machines with the same or similar material processing capabilities are usually grouped together (Chryssolouris, op.cit.). The part or batches of parts moves, according to the process plan, through the system visiting the different groups of machine. In order to accommodate many different part types, material handling must be very flexible. Therefore, a significant amount of manually controlled material handling equipment is usually employed, such as forklifts and handcarts. Such a shop is used for manufacturing a diverse mix of product, often custom or semi-custom, produced in low volumes (Ashton et al., 1990).

### 2.2.3 *Flow Shop*

A flow shop is characterised by a product flow layout where machines are ordered according to the process sequences of the parts to be manufactured. Successive jobs undergo the same sequence of operations. Only one part type is produced at a time. A typical example is the transfer line, which is often used in the automotive industry. This system is normally used for products produced in large volume.

### 2.2.4 *Continuous System*

The third approach to structuring the processing area is the continuous process system. In contrast to the other types of manufacturing systems which perform the manufacturing of discrete parts, continuous systems produce gases, liquids, or slurries. As in a flow line, processes are arranged in a series of directly connected processes or operations that link raw material with finished products. The continuous system is the least flexible of the types of manufacturing systems (Chryssolouris, op.cit.).

### 2.2.5 *Summary*

The job shop and other three systems previously mentioned are often classified as four classical manufacturing systems (Black, 1983; Kamrani et al., 1994). In recent years, the emergence of Cellular Manufacturing Systems as a new type of manufacturing system organisation and its highly automated form - Flexible Manufacturing Systems - have enabled manufacturers to produce a high number of product variations with both cost and quality-control advantages over the four traditional manufacturing systems (Black., ibid; Chen et al., 1994).

The concept of these cellular and flexible manufacturing systems, first developed in the early 1980s (Kamrani et al., op.cit.), are discussed in the following sections.

## 2.3 Cellular Manufacturing

An increasingly competitive environment has compelled manufacturers to look for ways to increase productivity. In a traditional factory, machine layout is predominantly arranged using the job shop approach. This functional layout has a number of disadvantages. A major problem is long and uncertain throughput times which in turn increases work-in-progress inventory, late delivery and eventually losses of sales (Kusiak et al., 1991).

Cellular manufacturing (CM) takes advantages of similarities between parts and processes. The basic concept is to form a group of machines, tooling and people into manufacturing cells. They are grouped according to the process combinations that occur in families of parts. Each cell contains the capability to produce a certain family of parts. The objectives of CM are: (1) to handle the problems associated with the job shop approach and improve the productivity of multi-product, small-to-medium batch manufacturing systems (Zhang et al., 1992), and; (2) to reduce the complexity of

manufacturing systems through simplification of material and information flows. In order to achieve such objectives, all the facilities required to support the conversion from raw material to final product are usually included within the cell. Intra-cellular material flow is performed either automatically or manually. Transportation times and the times spent in the system for the different parts are reduced compared to those of the job shop (Chryssolouris, op.cit.).

Cells can be categorised into two general groups: manned and unmanned (Kamrani et al., op.cit.; Black, op.cit.). In manned cells, trained operators operate multiple machines which are conventional and programmable. When a robot replaces workers in a cell, the cell is categorised as an unmanned cell. Of course, the robot must possess the workers' capabilities to treat each event in real-time.

Furthermore, the advent of sophisticated automated equipment such as robots, machine tools, and transport vehicles have increased the flexibility of the cellular manufacturing. A high number of part variations is now possible to be produced. This has led to the development of Flexible Manufacturing Systems which is discussed in the next section. This flexibility can be defined as: (1) the ability to produce the same parts on different machines and different parts on the same machine; (2) the flexibility of machines to accommodate design changes in parts, and; (3) the flexibility of existing machines to produce new parts (Miltenburg et al., 1987).

## 2.4 Flexible Manufacturing System

There is a growing interest in the development and implementation of flexible manufacturing systems (FMSs). Mertins and Wieneke-Toutaoui (1991) stated that in 1990 about 20% of investment into metal cutting machine tools in Germany was in FMSs.

Warnecke (1983) defined an FMS as:

> "Several automated machine tools of the universal or special type and/or flexible manufacturing cells (FMCs)[1] and as necessary, further manual or automated work stations. These are interlinked by an automatic workpiece flow system in a way which enables the simultaneous machining of different workpieces which pass through the system along different routes (p.682)".

In addition, the processing instructions are usually stored in the computer memory of a machine tool. This enables the FMS to perform customised operations automatically on each workpiece (Gunasekaran et al., 1993). Sensory devices in the machines and robots are used for automatic inspection.

FMSs are designed to produce a family of parts at low to medium volumes (The Charles Stark Draper Laboratories, 1984). The products may require operations on several machine-tools at various cells and the routing that specifies a sequence of operations to be performed and the operations themselves may differ for each product. An FMS producing 200 to 500 different products can be regarded as advanced (Bakker, 1988). Towards a classification of FMSs, Groover (1980) divided FMSs into two distinct types: (1) dedicated FMS, and; (2) non-dedicated FMS. Dedicated FMSs manufacture a fixed set of part types, whereas non-dedicated FMSs machines a greater variety of parts.

Decision-making issues in FMSs are divided into three levels (Jackson et al., 1989; Bilberg et al., 1991): Strategic, Tactical and Operational Levels (Figure 2.2). The distinguishing features at each level are its function and the time horizon employed (Jackson et al., ibid.). The Strategic Level corresponds to the corporate game plan. The Tactical Level is concerned with planning functions. Tactical Level functions include NC programming (Bilberg et al., op.cit.), process planning, selection of part types,

1. An FMC consists of several computer-controlled machines, equipped with means for automatic changing of parts and tools (Arzi et al., 1993).

design of fixtures and cutting tools, and the Master Production Schedule (MPS) (Rachamadugu et al., 1994). The MPS consists of specific product configurations, quantities and dates (Chryssolouris, op.cit.). Also, it contains purchasing from subcontractors, inventory control and labour requirements (Bilberg et al., op.cit.). The Operational Level is where shop floor control is to be found. Shop floor control generates a detailed plan with a short term horizon that determines which parts are going to be processed at the individual cells. Manufacturing control is made possible by interaction between these three different levels. Orders and directives flow "down" from the Strategic to the Tactical and finally to the Operational Level and reports flows "upward" therefore closing the loop between manufacturing activities (Jackson et al, op.cit).

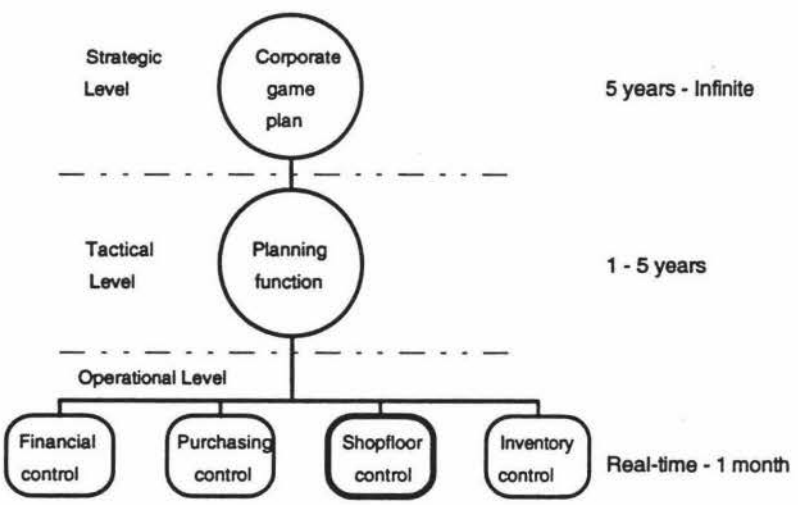


Figure 2.2: The hierarchy of decision making in FMSs (Browne, 1988)

Shop floor control at the Operational Level interacts with many different sub-systems, including financial, purchasing and inventory control. The major functional elements of the shop floor control subsystem include (Browne, ibid.): scheduling, dispatching, material movement control, process control and monitoring.

Structurally, the shop floor can be decomposed into three levels: Shop, Cell and Machine (Bourne et al., 1984; Disney et al., 1994). The Shop controller deals with task scheduling, dispatching tasks among cells and monitoring systems. The main responsibility of cell controllers is local task scheduling. Machine controllers concentrate on the execution of specific real-time operations on parts (process control).

In short, the primary focus of the shop floor control system is the *scheduling* of work orders. The scheduling methods employed depend primarily on the control architecture in use. The next section discusses control architectures for FMSs and how they impact the scheduling of work orders.

## 2.5 Shop Floor Control Architectures for an FMS

The choice of a particular control architecture directly influences the regulation of the flow of materials through a shop. Conventional manufacturing control systems adopt a centralised approach (Love et al, 1989). Advances in the area of manufacturing technology, computer and communications, however, have led to the consideration of other alternative control architectures, such as hierarchical and modified hierarchical control approaches. Hierarchical control uses a philosophy of "levels" with master/slave relationships between levels, command data flows downward and sensory data flows upward (Duffie et al, 1987b). In the modified hierarchical approach, each hierarchical level is given some degree of autonomy, with respect to higher levels (Cassandras, 1986).

Furthermore, the desire to fully eliminate global information and to locate the decision making where the information originated led to the consideration of decentralised control architecture, utilising highly autonomous entities. This heterarchical consideration was made possible by technological advances in the area of distributed computing.

Figure 2.3 illustrates the four standard control architectures (Dilts et al., op.cit., p.82). Boxes indicate control components and circles symbolise manufacturing entities (robots, AGVs, CNC machines, etc). The control interrelationship and information flows are shown by the connecting lines. The following sections discuss in more detail the characteristics of each type of control structure.

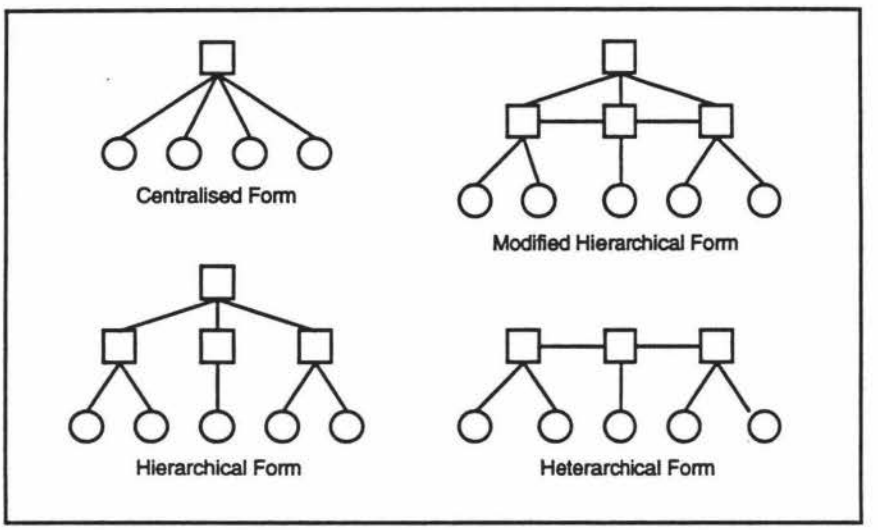


Figure 2.3: The Four Standard Control Architectures

### 2.5.1 *Centralised Control Architecture*

This architecture evidenced by a single computer used to perform all planning and control activities of every manufacturing-related activity within a plant (Love et al., op.cit.). It maintains a global database to record all relevant information concerning activities of the whole system.

There are no "shop and cell controllers". The responsibility for the shop and cell controllers is concentrated in a single controller, while simple (non-intelligent) machine controllers are dispersed throughout the shop floor. To generate an optimal schedule for processing parts, the central controller uses large amounts of information, such as process plans, due dates, processing times, setup times, travel times, equipment status,

and relationships between these variables (Duffie et al., 1994d). Instructions are issued to the machine controllers by the central controller in turn receives monitoring information from the machine controllers to use in making future decisions (Dilts et al., op.cit.).

Such systems inevitably become difficult to manage effectively. Accuracy of data is of prime importance if the system is to generate realistic and effective job schedules. It is also critically dependent upon timely and accurate feedback of shop performance data. System performance mainly rests upon good communications between the central controller and the non-intelligent machine controllers.

In addition, maintaining adequate levels of accuracy in bills of material, routings, and inventory data is especially difficult in large systems. The centralised control architecture, however, offers several advantages, such as: fewer computers are required, complete global information and overall status can be obtained from a single source (Dilts et al., ibid.).

### 2.5.2 *Hierarchical Control Architecture*

A rigid pyramidal structure and a tight coupling between master and slave are central to this architecture (Duffie et al., 1988a). The goal is to limit the size, functionality and complexity of any individual control module (Jackson et al., 1987). There exists "shop and cell controllers". The shop controller, however, still has global information and an overall view of the shop floor. Control instructions flow "down" the pyramidal structure, and feedback flows "upward" through the hierarchy.

The main responsibility of the shop controller is to coordinate and oversee the interactions of the cell controllers. The cell controllers control their own group of machine-tools. The machine controllers have direct control over the motors, actuators,

and sensors on the machines (Disney et al., op.cit.). Table 2.1 lists the functions of the hierarchical control architecture at the shop, cell and machine levels.

| | Planning Function | Scheduling Function | Execution Function |
|---|---|---|---|
| Shop Level | Determining part routes through the shop. Splitting part orders into batches to match material transport and cell capacity constraints. | Determining the start/finish times for part batches at each work-station. | Interacting with the cell levels to facilitate part transfer and delivery. |
| Cell Level | Determining the part routes through the cell (e.g., selecting processing machine). Includes replanning in response to machine breakdowns. | Determining the start/finish times for each part on each processing machine in the work-station. | Interacting with the machine level to assign and remove parts and to synchronise the activities of the devices. |
| Machine Level | Operations-level planning (e.g., change to the parts) | Determining the start/finish times for the individual tasks. Determining the sequence of part processing when multiple parts are allowed. | Interacting with the machine controller to initiate and monitor part processing. |

Table 2.1: Functions of the Hierarchical Architecture (Disney et al., ibid., p.52.)

Advantages of hierarchical control architectures include:

(1) decomposition of the control problems in the hierarchy to limit the size, functionality and complexity of any individual control module;
(2) existence of command/feedback mechanisms, allowing for adaptive behaviours to be incorporated in the control architecture;
(3) the shop controller has global information and an overall view of the shop floor, and;
(4) a rigid structure and a tight coupling between master and slave modules which usually results in fast response times (Duffie et al., op.cit/a).

Disadvantages of hierarchical control architectures include:

(1) vulnerability, if the central control system breaks down, the whole system will become inoperative (Bakker, op.cit.);
(2) the rigid structure implies that the only exchange of allowed information is restricted only between a supervisor and its subordinates;

(3) any extension must be foreseen in the design phase as subsequent unforeseen modifications are difficult to implement (Duffie et al., op.cit/a.);
(4) prohibitively complex and unreliable with the increasing size and complexity of the manufacturing system (Veeramani, 1993), and;
(5) a crash at some level may cripple the entire system (Veeramani et al., ibid.).

In an attempt to handle some of the shortcomings of the hierarchical control architecture, the *modified* hierarchical architecture was developed.

### 2.5.3 *Modified Hierarchical Architecture*

The concept of levels exists in this architecture, however, this is not "rigid" master/slave relationships. Local area networks (LANs) and the availability of computing power have made it possible to give some amount of autonomy to the lower levels. The main responsibility of the shop controller is not to coordinate but to initiate and to supervise activities (Cassandras, op.cit.). The cell controllers act not only as slaves but as intelligent assistants to the shop controller.

In discussing the relationship between the shop and cell controllers, O'Grady introduces the notion of a modified hierarchical form (Dilts et al., op.cit.). In what he terms 'decentralised mode', a number of characteristics are explained (Dilts et al., ibid.; O'Grady, 1986): the main function of the shop controller is to launch the first task of a job at a cell and pass enough information to this first cell controller. The first cell controller then evaluates its resources and schedules its own activities. In case the task can not be done on (or near to) time, the cell controller notifies the shop controller. The shop controller has the option of either agreeing to a revised finished time or rescheduling the task at a different cell. When the task nears completion, this first cell controller has to arrange for enough information to be passed to the subsequent cell controller to permit the completion of the job.

O'Grady (ibid.) states that "in this manner the shop control system (SCS) is managing by exception. It is only when there is a significant departure from planned activities that the SCS is involved; otherwise the shop carries on with the allocated loading" (p.85.).

A principal advantage of this architecture is that some rudimentary tasks are delegated from the shop controller to the cell controllers. This may enable the shop controller to respond more readily to more important requests from subordinates. However, there exist connectivity problems with the peer-to-peer interaction among cells which can complicate the control system design (Dilts et al., op.cit.).

### 2.5.4 *Heterarchical Control Architecture*

Recently, to overcome the disadvantages associated with each of the three previously discussed control architectures, Duffie and Piper (1987c) proposed a heterarchical approach. Unlike the previous architectures, the heterarchical control architecture utilises a distributed control approach which creates a flat architecture dividing control responsibilities among cell controllers. This system contains a congregation of autonomous cell controllers which acts on the basis of information exchanged amongst each other (Veeramani et al., op.cit.). There are no master/slave relationships. The cooperation between entities is achieved through an auction-based scheme.

Specifically, when a cell (acting as a manager cell) initiates a task, a bidding token is passed over the LAN to communicate the task to each cell (Tilley et al., 1992). Cells which are able to process the task will, in turn, transmit a bidding message to the manager cell. The bidding message contains the estimated earliest time that the task can be finished. After the deadline for bid submission has passed, the manager cell awards the task to the "best" bidder. The successful cell acknowledges acceptance of the job award by means of a confirmation message. If there are a number of tasks already in the

queue at the winning cell, the task joins the end of the queue, otherwise it will be processed immediately.

Cells indicate completion of the task by means of a task completion message. The cell that has finished a task checks to see if more tasks are required to complete the job. If all tasks have been completed, the workpiece is sent to the storage area, otherwise the cell will act as a manager cell and determine through an auction which cell is best suited to perform the next task.

The main advantages of the heterarchical architecture are:

(1) reduced software complexity;
(2) higher modularity and extensibility (Duffie et al., op.cit./c), and;
(3) easy to be reconfigured to accommodate addition or removal equipment.

The main disadvantages of this architecture, however, are as follows:

(1) as in the modified hierarchical form, there exists the complexity of peer-to-peer operating system/relationship among cells, and;
(2) no clear supervisor.

### 2.5.5 *Discussion*

Based on the review regarding these shop floor control architectures, it is discovered literally that even though levels of control have been introduced in the hierarchical form to limit the complexity of any individual control module, subsequent unforeseen modifications are difficult to implement and a crash at some level may cripple the entire system. To overcome these problems, the modified hierarchical form was developed. Some rudimentary tasks are now delegated to cell controllers. However, production schedules are still planned before the actual production. Preplanned schedules sometimes can not compensate for "surprises" in real time. As previously indicated, the cell controllers can of course notify the shop controller if it can not do a task assigned

on time. This can nevertheless add connectivity problems that already exist with the peer-to-peer interaction among the cells. Moreover, the shop controller has to agree to a revised finished time or rescheduling the task at another cell.

Recently, the heterarchical control architecture was introduced mainly to reduce software complexity and increase modularity and extensibility. Nevertheless, no clear supervisor in this form means no direct control. No direct control means that there is no "someone" who supervises the cell controllers to accomplish a global goal. In addition, connectivity problems also exist with the peer-to-peer interaction among cells.

It was felt, therefore, the need for a control architecture which, on the one hand, can overcome the above problems and, on the other, is expected to maximise the advantages associated with the existing control architectures. This has led to the development of a proposed control architecture which is called the hybrid control architecture. This is discussed in Chapter 3.

## 2.6 Simulation in Manufacturing

Simulation is a systems analysis activity that is performed by modelling a hypothetical or real dynamic system and observing its behaviour over time (Pollacia, 1989; Law et al., 1994). Two key words in the definition are: (1) model, and; (2) experimentation.

A model is a representation of a hypothetical, proposed or real system which includes the rules on how the components of the system interact with each other. In practice, a software package is generally used to build the model. Experiments are then executed by running the model for a simulated time period. The user specifies the initial state of the model. At the conclusion of the simulation, the output provided is a set of statistical performance measures.

Robinson (1994) states that "there is a tendency to want to model everything without stopping to consider exactly what is necessary" (p.34.). Deciding upon correct model scope and level of detail is therefore a key element.

Within manufacturing, the scope of the model could be somewhere between two extremes: the whole company and an individual operation. Robinson (1993) has grouped the scope of manufacturing models into seven categories, these are :

(1) Facilities Planning

The objective of a simulation study in this category is usually to ensure that a new facility will perform correctly. For example, by imitating the operation of the facility, bottlenecks are identified, shortages found and solutions sought.

(2) Obtaining the Best Use of Current Facilities

Not only for a new facility, simulation can also be used to determine whether or not a current facility is performing effectively.

(3) Developing Control Logic

More than just physical equipment, there is a wide variety of issues concerning working practice in manufacturing, for instance, product routing, work scheduling and labour organisation. Use of simulation enables the best working practice to be developed.

(4) Materials Handling

Simulation is used here, for example, to model the material flows and methods of handling.

(5) Company Modelling

Company modelling is particularly useful for simulating operations across more than one location.

(6) Operational Planning

Common applications are to test a production schedule by simulating the resulting plant performance.

(7) Training Operation Staff

Simulation provides a low-risk environment in which plant supervisors and operators are trained in the operation of a facility.

In summary, within manufacturing, simulation is a powerful tool in decision making with the potential to substantially assist gaining correct decisions. It is important that this technology is used effectively. Experience has shown that this is the only approach for obtaining practical solutions to real life problems of flexible manufacturing systems (El-Tamimi et al., 1989).

## 2.7 Summary

Cellular Manufacturing and Flexible Manufacturing Systems have emerged in recent years as new types of manufacturing system organisation. They enable manufacturers to produce a variety of products with short lead times, high quality and prices close to those of mass production.

This chapter has outlined four existing shop floor control architectures for controlling activities in an FMS environment. They are: centralised, hierarchical, modified hierarchical and heterarchical architectures.

As previously mentioned, Chapter 3 outlines the new control architecture called the hybrid control architecture. To study its behaviour, it is necessary to create a simulation model to emulate its working. Design and implementation of the simulation study conducted along with analysis of results obtained are discussed in Chapter 4 and 5 respectively. As indicated in Chapter 1, the hybrid control architecture and the subsequent simulation study in this research are especially focused on the scheduling aspect.

# CHAPTER 3

# THE PROPOSED CONTROL ARCHITECTURE: A HYBRID FORM

## 3.1 The Hybrid Control Architecture

Figure 3.1 illustrates the hybrid control architecture. As in Figure 2.3, control components are represented by the boxes and circles symbolise manufacturing resources (robots, AGVs, CNC machines. etc). The connecting lines indicate the control interrelationships. The specific characteristics of the hybrid control architecture are outlined in Table 3.1

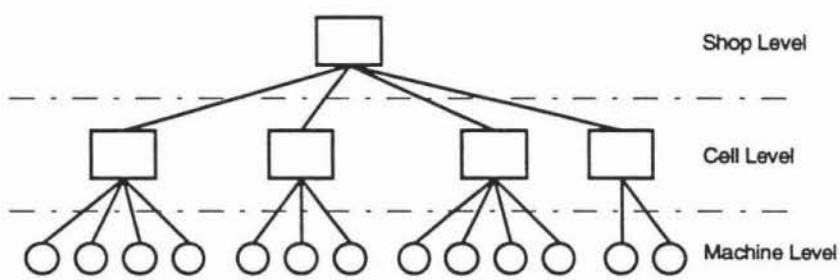


Figure 3.1: The Hybrid Control Architecture

| Characteristics of the Hybrid Form |
| --- |
| • Levels of control<br>• Full autonomy for subordinates<br>• Full intelligent entities<br>• No peer-to-peer interaction<br>• Auction-based scheme for scheduling jobs |

Table 3.1: The Characteristics of the Hybrid Control Architecture

### *Levels of Control*

The concept of levels of control with established supervisor/subordinate relationships as in the hierarchical control architecture is also a property of this hybrid control architecture. The main distinction between hierarchical, modified hierarchical and the hybrid form lies in the degree of autonomy of the subordinates.

### *Full Autonomy & Full Intelligence*

No autonomy for subordinates in the hierarchical form, some amount of autonomy for subordinates with respect to higher levels in the modified hierarchical form, and full autonomy for subordinates, as in the heterarchical form, in the hybrid form. In addition, the hybrid form considers cell controllers - as in the heterarchical form - as "full" intelligent entities, as opposed to intelligent assistants in the modified hierarchical form.

### *No peer-to-peer Interaction*

Unlike the modified hierarchical and heterarchical forms[1], there is no peer-to-peer interaction among subordinates in the hybrid form. In other words, there is no "horizontal negotiation" between cells. The aim is primarily intended to avoid the connectivity problems of peer-to-peer communication and to reduce some of the complexity of the control system design.

The proposed method of work allocation is therefore through "vertical negotiation" (Figure 3.2). Cells do not negotiate with each other, but they negotiate with the shop controller. The shop controller acts as "a single final decision maker" in arranging scheduling and routing of work-parts. This is the main distinction between the hybrid form and the heterarchical form. In the heterarchical form, each cell can act as "a final decision maker" whenever it initiates a task.

---

1. The difference between the modified hierarchical and heterarchical forms is that in the modified form, initiation is started by a command from the shop controller/supervisor, whereas in the heterarchical form - because there are no external higher levels of control - initiation is started by a particular cell which has finished processing a task.

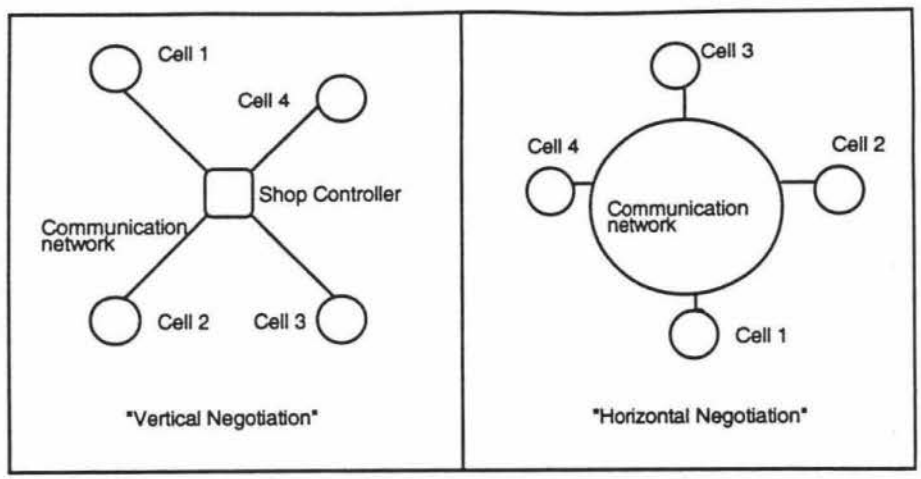


Figure 3.2: Types of Negotiation

## 3.2 The "Hybrid" Auction-based Scheme

An auction-based scheme as in the heterarchical architecture is employed to allocate tasks. The main distinction between hybrid and heterarchical forms lies in the method of interaction. As indicated above, the method of interaction in the hybrid form is "vertical negotiation". The shop controller acts as a centralised "auctioneer" and the selection of tasks to be auctioned is based on "First Come First Auctioned".

Specifically, when the shop controller initiates a task of a job, a bidding token is passed over a LAN to communicate the task to each cell. Depending on the information in the task-announcement packet, cell controllers decide whether the auction is relevant for them in terms of the availability of machines in the cells and the capacity limit of their queue/buffer. Cells which are able to process the task interact with their particular machine controllers to estimate the earliest finishing time (EFT) for the task. The EFT is calculated based on:

(1) the estimated processing time of the auctioned task which is determined from basic data such as batch size, setup time, loading time and machining time specified in the task-announcement packet;
(2) the sum of the estimated processing time of the tasks in the queue;
(3) activity duration, and;
(4) estimated travel time.

The activity duration refers to the difference in time between the arrival time of a task at the queue and its earliest start time (Baker, 1974). The estimated travel time is calculated based on the travel time between the shop controller and the cell, or between two cells.

The cell controllers, in turn, transmit a bidding message which contains the EFT of the task to the shop controller. After the deadline for bid submission has passed, the shop controller awards the task to the "best" bidder and the workpiece is transported to the winning cell. Transportation is arranged by the shop controller.

The successful cell acknowledges acceptance of the task award by means of a confirmation message to the shop controller. If there is a number of tasks already in the single queue at the cell, the task joins the end of the queue, otherwise it will be processed immediately.

The queue is maintained on a First-In-First-Out (FIFO) basis. The total size of the queue at a cell reflects the capacity level of the cell. Where the input queue of a cell has reached maximum capacity, the cell would not join an auction declared by the shop controller.

Cell controllers indicate completion of tasks by means of a task completion message to the shop controller. The shop controller is responsible for determining if more tasks are needed to complete the job[2]. If all tasks have been completed, the workpiece is sent to the storage area[3], otherwise the shop controller will announce an auction to determine which cell is best suited to perform the next task.

---

2. For each job, the shop controller knows the sequence of tasks that have to be performed to complete the job.
3. The storage area is maintained by the shop controller.

## 3.3 Advantages and Disadvantages

The hybrid control architecture contains global information as well as local databases. The global information is, of course, not as complex as the global information in the hierarchical/centralised forms. The local databases serve local users without requiring access to any other part of the system. Access to data not available locally is made possible by transfer or replication of data by the higher level. The higher level facilitates communication and integration among the local databases.

Other advantages arising from the hybrid control architecture include those associated with the centralised, hierarchical and heterarchical control architectures, such as:

(1) Single source for system status information.

(2) Access to global information.

(3) Gradual implementation and reduced software development problems.

(4) One clear supervisor.

(5) Full local autonomy.

(6) Simplicity in scheduling system.

No schedule is made before the actual production starts, jobs are allocated to cells as and when they arrive.

The disadvantages of the hybrid control architecture include:

(1) High dependence on the shop floor.

This means if the shop controller goes "down", the system will "die".

(2) Slightly limited range of product variety.

This is due to limited number of different tools that may be provided in each cell. This indicates that the hybrid control architecture is mainly intended to be employed in a dedicated FMS.

(3) Limited optimisation.

No attempt is made to optimise the productivity of the cells, for example, by changing the sequence of operations once allocated.

These can be considered as minor disadvantages. Advances in the area of network communications and sophisticated computing power have increased the reliability of the computer-based plant at the shop level (Popovic et al, 1990). Moreover, since the global information at the shop level is not as complex as in the hierarchical/centralised forms, the shop controller should be in charge of solving central automation problems only, leaving the cell controllers to cope with the local problems in their immediate environment.

Limited optimisation may be circumvented in part by carefully planning of routings, process plan for each product, lot size and configuration of the system.

## 3.4 Summary

The major characteristics of the hybrid control architecture are a single supervisor, local autonomy, full intelligent entity and simplicity in scheduling.

Chapter 4 illustrates the implementation of this architecture which incorporates two software packages to serve two functions: control and simulation.

# CHAPTER 4

# IMPLEMENTATION OF THE HYBRID CONTROL ARCHITECTURE

## 4.1 Introduction

This chapter discusses the conceptual framework of the implementation, the software implementation, hardware requirements, the interface and detailed control functions for the shop, cell and machine controllers as well as the functions of the simulator.

The concept of the simulation model is based on a simulation package serving two functions - the control facility and simulation tool. Technically, it involves two software packages - a real-time control software[1] and a Windows programming language. Such a prototype system is called SIMCON (Simulation Control). Communication between these two packages is performed via an interface which is written using a Windows programming language.

The real-time control software used is FIX DMACS[2] for Windows. It provides real-time data to both plant personnel and other software applications throughout an enterprise (Intellution, 1994). In this way, the information in the FIX DMACS database reflects the current situation on the shop floor. It does this by communicating directly with the I/O (Input-Output) devices already in place on the shop floor, through a software interface called an I/O Driver (Intellution, ibid). This software is employed to

---

1. A real-time control software is a software that is specifically designed to automate an industrial process. It has Supervisory Control and Data Aquisition (SCADA) functions.
2. FIX DMACS stands for Fully Integrated Control - Distributed Manufacturing Automation and Control Software.

construct the modelled FMS shop floor and to implement the auction-based sequencing functions of the shop, cell and machine controllers.

To simulate the arrival of jobs over time, a simulator is created using a Windows programming language. The Windows programming language used is Microsoft Visual Basic. It provides the capabilities to read and write data from other applications, such as FIX DMACS. This is done via DDE and Easy Database Access (EDA). DDE is Microsoft's dynamic data exchange for Windows applications, whereas EDA is an application program interface of the Visual Basic language that allows users to read or write data to a FIX DMACS database. Moreover, Microsoft Visual Basic is also employed to emulate the modelled shop floor by developing supporting control functions for the shop, cell and machine controllers[3].

## 4.2 Conceptual Framework of the SIMCON Package

Conceptually, Figure 4.1a illustrates the framework of the SIMCON package. The figure shows that for the purpose of the simulation study, the real-time control software is at the moment "off-line" with a simulation I/O driver[4] emulating the real world. Also, the figure shows that the simulator and supporting functions (the Visual Basic programs) are linked to the real-time control software (the modelled FMS shop floor and auction-based control functions) via *an interface*.

***Benefits***

By running the Windows programming language (in this case Visual Basic programs) and the real-time control software (in this case FIX DMACS database) simultaneously, the modelled FMS shop floor, auction-based control functions, simulator and

3. FIX DMACS is designed to control a process-based manufacturing facility, it does not contain the flexibility required to control a discrete manufacturing (Butler et al., 1994). Therefore, Microsoft Visual Basic is used to implement some functions.
4. FIX DMACS comes with a simulation I/O driver (SIM) that allows users to test how the modelled FMS shop floor operates before connecting to real I/O drivers.

supporting control functions will operate automatically, as if the real shop floor exists. The controls conducted and decisions made are the same as in the real time. Rapid analysis can be done and if it is required, the model can be reconstructed without risking disruption to production or to the control system. Changes can be made when still in the development phase. In addition, since the simulator interacts with objects that correspond to real life entities, there exists a unique opportunity to develop the simulator that is initially used as a simulation tool and, later, as supporting control software for the real-time control system (see figure 4.1b). For example, in the real situation, the simulator can be used to simulate a proposed schedule to see if feasible before actual implementation.

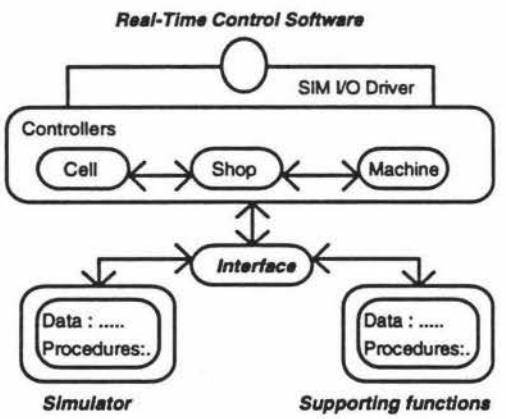


Figure 4.1a: Conceptual Framework of SIMCON

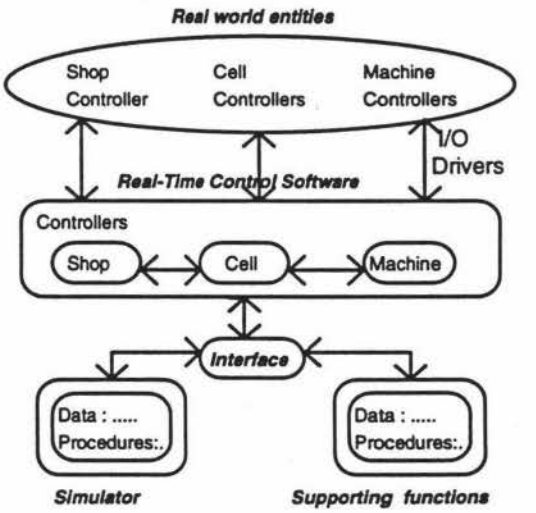


Figure 4.1b: Real-time Situation

### *Interface*

The interface facilitates Visual Basic and FIX DMACS to exchange information between them. Information is sent, for example, from the simulator to the interface. The interface then passes the information to the FIX DMACS database blocks. Creating an interface, offers the following benefits: (1). The interface may easily be modified with minimal disruption to both the control and simulation programs; (2). Extensions to the functionality of the simulator, supporting functions, modelled FMS

shop floor or the auction-based control functions may also easily be made. If it is required, the interface may be augmented by adding new items to cover additional application areas with minimal effort, and; (3). Changing the real-time control software with another control software but also operating under the same environment (Windows) may also be possible, since information that needs to be passed between the two softwares are known. Elements in the "new" control software that have to be connected to the interface may be designed, based on the information that is required to be sent or received via the interface.

## 4.3 Development Platform

The reasons for using FIX DMACS for Windows include:

1. FIX DMACS is a PC-based industrial control applications.
2. It provides SCADA functions at a fraction of the cost of proprietary hardware systems.
3. It requires no proprietary hardware to acquire data. It has an extension catalogue of I/O drivers that supports best-selling and specialty I/O devices. Even if a plant has I/O devices from different manufacturers on the same network, FIX DMACS' I/O drivers can work with all of them.
4. It contains a set of Visual Basic, C, C++, and Visual C++ language functions that provide read and write access to any data point in the FIX DMACS database.

The primary reasons for using Microsoft Visual Basic are that it is easily understood by any researcher who wishes to extend this research, and it is a complete programming language that supports structured programming constructs found in most other modern programming languages like PASCAL. For example, case statements, do-loop, if-then-else and while-loop statements. In addition, Visual Basic is a Windows programming language that is able to exploit the key features of Microsoft Windows, including multiple-document interface (MDI), object linking and embedding (OLE), dynamic data

exchange (DDE) and graphics. Also, it can be extended by calling procedures in dynamic-link libraries (DLLs).

### 4.3.1 *FIX DMACS - The Real-Time Control Software*[5]

FIX DMACS is a powerful software solution for industrial automation and can be applied for small installations or large, networked configurations. It enables plant engineers to configure a system environment that provides two main functions:

1. **Supervisory Control**

   Supervisory control, batch processing, data acquisition, continuous control, and statistical process control for industrial applications. This function is central to the operation of the FIX DMACS software. Figure 4.2 illustrates the SCADA functions of FIX DMACS.

2. **Process Information**

   Process information for plant managers, supervisors, and operators in the form of reports, displays, archived data, alarm messages, and statistical charts.

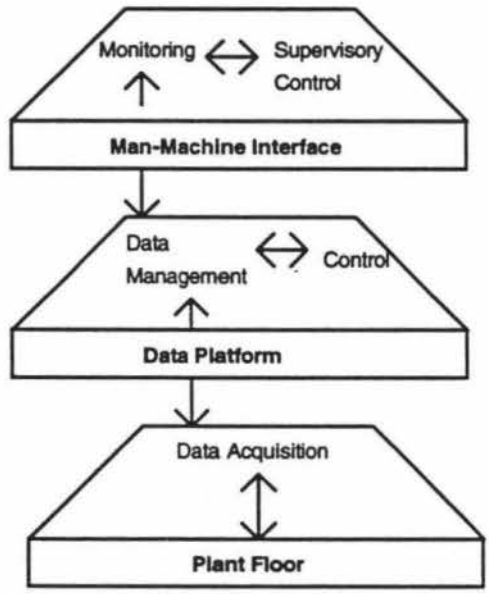


Figure 4.2: FIX DMACS' SCADA Functions

5. The primary source of information in this section is taken from the FIX DMACS for Windows Software's Manuals, Intellution, 1992 - 1994.

*Plant Floor*

At the plant floor level, FIX DMACS has the ability to retrieve data from the plant floor and to process that data into a usable form. Data can also be written to the plant floor, thereby establishing the critical two-way link that control and application software require.

*Data Platform*

At the data platform level, data acquired is manipulated and channelled according to the request of software applications (Data Management). In addition, FIX DMACS automatically apply algorithms that adjust process values and thereby maintain those values within set limits (Control).

*Man-Machine Interface*

At the man-machine level, FIX DMACS displays real-time plant-floor data to operators (Monitoring). Powerful numeric, text, and graphical formats are available to make data more accessible. Since FIX DMACS can read and write plant-floor data, it can be established a supervisory control station to manage which points are readable and writeable and which are read only.

In addition, FIX DMACS also has the ability to create reports of critical system and process information. Moreover, the open architecture of FIX DMACS provides plant engineers, as indicated previously, a set of Visual Basic, C, C++, and Visual C++ language functions to write software applications that resolve unique automation needs.

*FIX DMACS Capabilities*

Equally important to what FIX DMACS can do, is how it is done. FIX DMACS processing capabilities allow for a wide variety of configurations and processing strategies. The architecture of FIX DMACS allows plants to distribute critical functions among all computers (nodes) on the network (distributed processing). Each node can

communicate with all other nodes on the network, but local tasks are not necessarily dependent on other nodes. Some applications, however, only need one node to perform the required functions. FIX DMACS can also operate just as smoothly in a single computer environment (centralised processing). It is easy to convert a distributed node to a stand alone or a stand alone node to a distributed node.

### *FIX DMACS Programs*

The heart of the FIX DMACS is the process databases. The process database that users create with the Database Builder Program consists of blocks and chains. A block is a coded set of process control instructions that perform some or all of the following:

1. receiving values either from another block or directly from the Driver Image Table (DIT)[6];
2. manipulating values according to user's instructions;
3. comparing values against alarm limits;
4. scaling process values to a specified range;
5. performing calculations, and;
6. outputting values back to the DIT.

FIX DMACS provides 29 types of blocks, each capable of performing a unique function. In this research, four types of blocks are employed: Analog Input (AI), Analog Output (AO), Calculation (CA) and Program blocks. The Analog Input can be used to read analog values at set time intervals or by exception from an I/O address, such as a Programmable Logic Controller (PLC) register, and initiate the processing of other blocks through the Next Block field. The Analog Output can be used to send values to PLC registers and pass values to other blocks through the Next Block field. The Calculation block can perform complex or multiple equations by chaining one calculation block to another through the Next Block field. The Program block provides a powerful means of running short programs.

---

6. DIT is an area in computer memory used by the I/O driver to store process data.

When creating a database, two or more blocks may be connected to each other to create a control or monitoring loop. This is called a chain. For example, a particular chain may read an input data from the DIT, manipulate the data with a standard formula, and write the output data out to the DIT. The chain that executes this control strategy might consist of an Analog Input block connected to a Calculation Block connected to an Analog Output block.

Figure 4.3 shows how to use Program blocks in combination with Analog Output blocks to determine the winning cell. The sample problem in this figure is related to the shop controller's bid-evaluation procedure. It can be described as follows: assuming that based on the bidding information which contains a description of the auctioned task, "cell 1" and "cell 3" blocks indicate that they are able to process the task initiated by the "FC1" block. They interact with the "machine1a" and "machine3a" blocks[7] (which represent machine controllers for machine type 1 in cell 1 and machine type 1 in cell 3 respectively) to estimate the earliest time the task can be completed. The "c1eft" and "c3eft" blocks, in turn, accept a bidding message that contains the EFT of the task from the cell controllers 1 and 3. After the deadline for bid submission has passed[8], the "whowinac" block interacts with the "whowinac1" and "whowinac2" blocks to determine the winning cell.

***Program Block Code***

Programming statements inside the "whowinac" and "whowinac1" blocks can be seen in Table 4.1. The maximum allowed lines in each block is only 19 lines. However, a program block can be used as a master program to develop flexible, generic subprograms. By linking a master program block with subroutines in other program blocks, it is possible to use the blocks in many different applications. In Table 4.1, the

7. It is assumed that the "machine" block in the bidding information indicates that the machine is machine type 1. Hence, the cell controllers just interact with their machine controller "type 1".
8. Based on the experiment that has been done, the duration of the bid evaluation procedure is about 20 seconds.

"whowinac" acts as a master program that calls other blocks, for example, the "whowinac1" block.

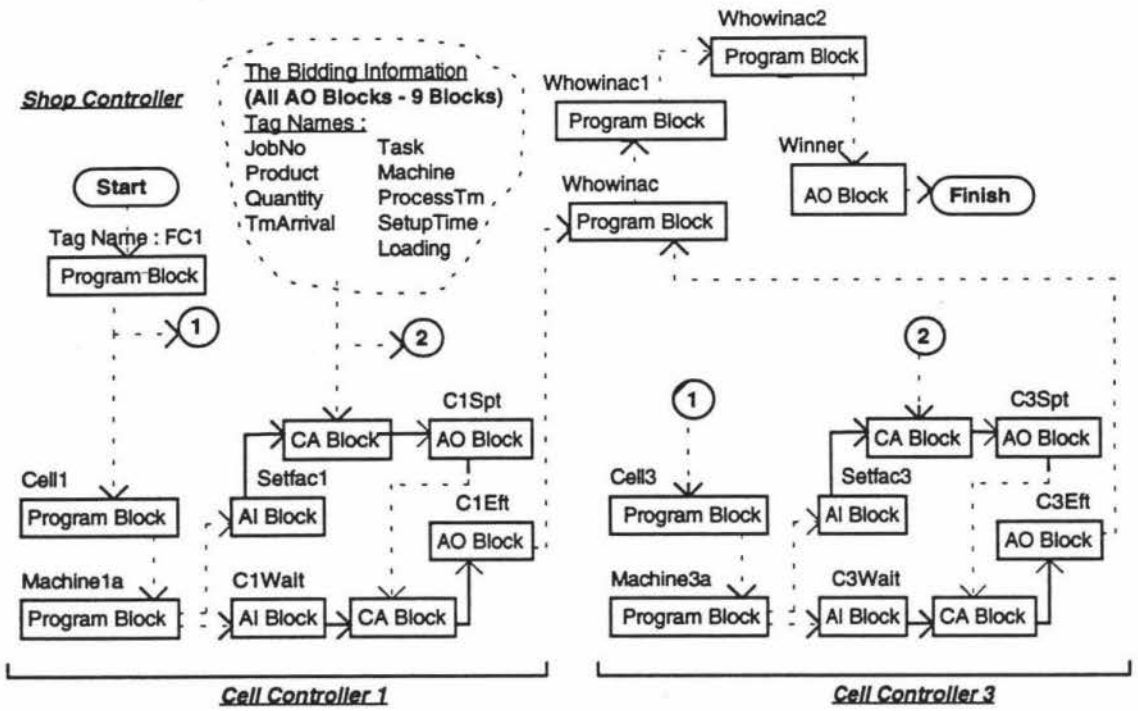


Figure 4.3: Using Program Blocks[9]

| **Program block- *whowinac*** | **Program block - *whowinac1*** |
|---|---|
| 0. setdebug | 0. setout response 2.00 |
| 1. delay 2 | 1. delay 1 |
| 2. setlim 0.5000 | 2. setlim 0.3000 |
| 3. if c11eft = 0.00 goto 7 | 3. if c1eft > c3eft goto 6 |
| 4. if c33eft = 0.00 goto 9 | 4. run whowinac2 |
| 5. run whowinac1 | 5. end |
| 6. end | 6. run whowinac3 |
| 7. run whowinac4 | 7. end |
| 8. end | |
| 9. run whowinac5 | |
| 10. end | |

Table 4.1: Program Block Code

9. Note that some blocks are designed to work in chains while others can operate either as a stand-alone block or in chains. The dotted lines indicate that the blocks work as stand-alone blocks, while the hairlines indicate that the blocks work in chains.

Using different types of available blocks, a FIX DMACS application program which corresponds to the three shop floor "actors" - shop controller, cell controllers, and machine controllers - including their auction control functions, is constructed. For the purpose of the simulation study, all controllers at the moment are "off-line". This means that they are not connected to real I/O drivers[10]. To run the FIX DMACS program, a simulation I/O driver (SIM) is used. The SIM driver is a matrix of addresses. Database blocks read values from and write values to the addresses. If one block writes to a specific address, other blocks can read the same value from the same address. In this way, the resources (e.g. PLCs) are emulated.

Further information concerning FIX DMACS database blocks that were used to implement the hybrid control architecture are included in Appendix A.

### 4.3.2 *Microsoft Visual Basic - A Windows Programming Language*

Visual Basic is a Windows programming language, an environment that combines event-driven programming and a visual route to user interface design. In this research, three main steps followed when Visual Basic applications were created. These steps are:

1. Create the interface

   This is done by drawing controls to make up the interface on programs' Windows, called forms. The forms serve as the interface of the application. Controls are used to get input data and to display output. Visual Basic provides 23 controls. Some of the controls commonly used include text boxes, command buttons, and list boxes. Each type of control has its own set of properties and events. The next step is to set properties.

2. Set properties

   This is to set properties for the controls created. A property is the value on a control, such as size, caption, or colour.

10. I/O driver is the software interface responsible for acquiring data from process hardware and storing it in the Driver Image Table.

3. Write code.

   The last step is to write code to describe the events that can happen to a form or its controls. Code consists of language statements, constants, and declarations. In a Visual Basic application, code is divided into smaller blocks called procedures. An event procedure contains code that is executed when an event occurs, for example, a user double clicking on a command button. The core of the Visual Basic code is Microsoft's QuickBasic.

Table 4.2 shows an example of an event procedure written in this research to send data to FIX DMACS database blocks via EDA. This procedure is executed by clicking the command button, named AutoWrite, after opening a file that contains the database blocks' names. Data sent are setup and loading factors which are generated randomly for each machine type in each cell.

In summary, Visual Basic contains all the design tools necessary to create Windows-based applications. The main tools are Forms, Toolbox, Properties Window, Code Window and Project Window. A Project Window lists forms, code modules, and custom control files that make up a Visual Basic application[11].

11. Further information concerning Visual Basic can be found in Microsoft Visual Basic's Manuals.

```
Sub AutoWrite_Click( )
        Randomize
        Dim loval1 as single, hival1 as single, value as single
        Static n as string * 8
        Static t as string * 10
        Static f as string * 7
        'Delete old eda group if any
        If (egroup <> 0) then eda_delete_group (egroup)

        'resize the handle list then copy the tags from the list

        egroup = eda_define_group(1,0)
        Redim writetofix(0 to list2.listcount)
        Redim ehandle(0 to list1.listcount)

        Open "setup" for output as #6
        For i = 0 to list1.listcount - 1
                list1.listindex  = i
                tmp$ = list1.text
                Call ntfparse(tmp$, n$, t$, f$)
                ehandle(i) = eda_define_ntf(egroup, n$, t$, f$, 0)
                Call eda_lookup(egroup)
                Call eda_wait(egroup)
                Loval1 = loval1.text
                Hival1 = hival1.text
                Value = (hival1-loval1)* Rnd + loval1
                Value= format(value, "#.##")
                WritetoFix(i) = eda_set_float(egroup, ehandle(i), value)
                WritetoFix(i) = eda_write1(egroup, ehandle(i))
                Print #6, tmp$; spc(3); value
        Next i
        Close #6

        eda_read_button.enabled = false

End Sub
```

Table 4.2: The Visual Basic Code

### 4.3.3 *Basic Architecture of "FIX DMACS - VISUAL BASIC"*

Figure 4.4 illustrates the structure of the implementation. The flow of process data from Visual Basic through FIX DMACS can be summarised as follows :

[1] Via EDA/DDE, Visual Basic applications writes data, such as initialise all system variables and new job arrival to the database blocks via "View"[12].

12. An operator display station.

[2] The database blocks, through the FIX DMACS internal database access functions, reads the data from "View". The database blocks perform either a special function or manipulate the data passed to them.

[3] The Scan, Alarm, and Control (SAC) program writes the data from the database to the Driver Information Table (DIT), processes it, and transfers the data to address fields of the simulation I/O Driver.

The flow of process data from FIX DMACS to Visual Basic can be summarised as follows (see Figure 4.4):

[4] The Scan, Alarm, and Control (SAC) program reads the data from the DIT, processes it, and transfers the data to the process database.

[5] The FIX DMACS internal database access functions read the data from the database and transfer it to the View program.

[6] Via DDE/EDA, Visual Basic applications read the data and process it.

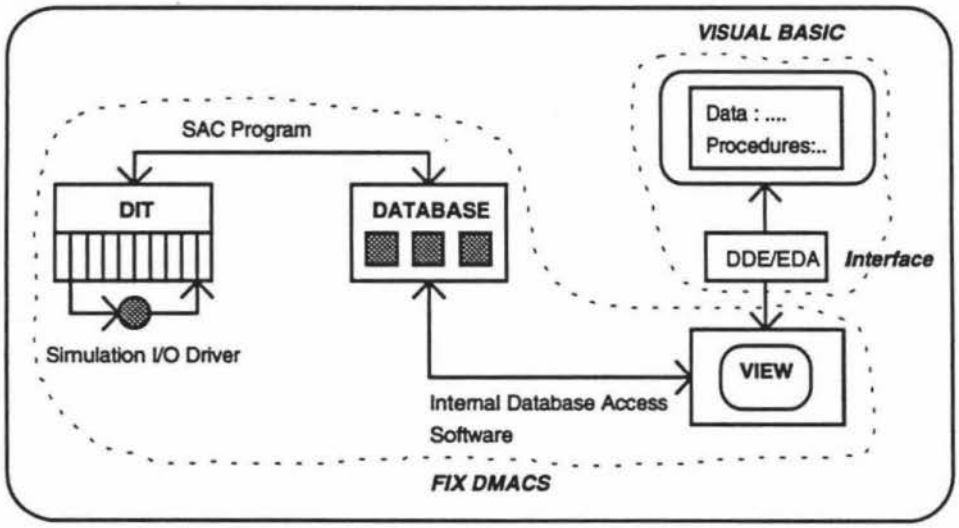


Figure 4.4: Basic Architecture of "FIX DMACS - VISUAL BASIC"

In the real "world", figure 4.5 shows that through the I/O driver, the database blocks pass the data to a network of sensors and controls connected to I/O devices such as PLCs.

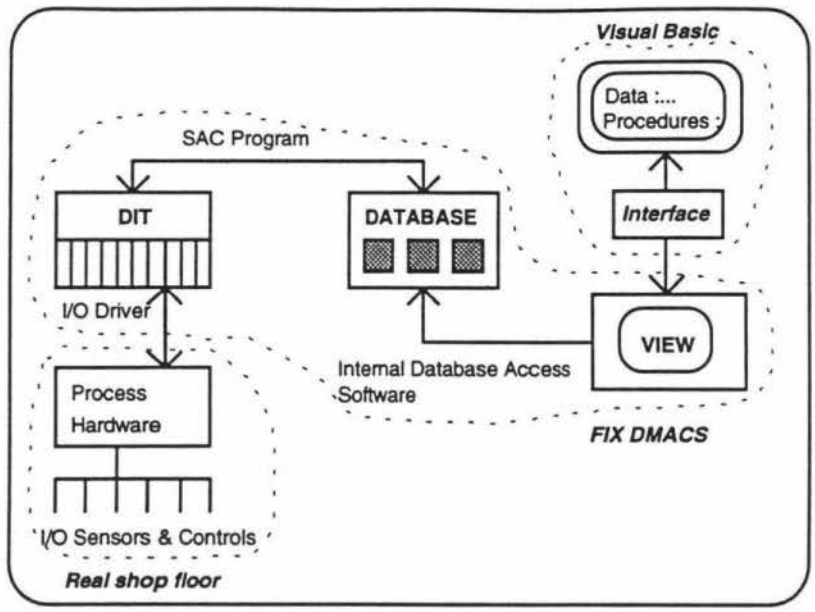


Figure 4.5: Basic Architecture of FIX Software in the Real "World"

Collectively, the simulation I/O driver, SAC, and the process database make up the data acquisition and management (DAM) function of FIX DMACS. Connected to the real plant floor, the DAM function provides the basis for *all* the industrial automation tasks that FIX DMACS can perform (Intellution, ibid.).

### *Hardware Requirements*[13]

The minimum requirements include: (1). Any IBM-80386 or 80486 computer or compatible preferably with a colour graphics adapter, VGA, super VGA, or XGA; (2). One megabyte of memory and one 3.5" 1.44M disk drive to run Visual Basic and FIX DMACS, respectively; (3). 8M of RAM; (4). Microsoft MS-DOS version 6.x or later. (5). Windows version 3.1x or later, and; (6). A mouse supported by Microsoft Windows.

13. To run in a real networked configuration, please refer to the FIX DMACS Manuals, particularly sections related to the Environment Setup and Hardware/Software required. In addition, the SIMCON program may need some modifications. In this research, however, network interface software is not required, since both softwares can be run in a stand-alone node.

## 4.4 The Functions of Each Controller and the Simulator

Table 4.3 describes the functions of each of the controllers and the simulator[14]. FIX and VB refer to FIX DMACS and Visual Basic, respectively. Figures 4.6a and 4.6b illustrate the overall logical flow of the SIMCON's functions. The numbers in the figures refer to the numbers in Table 4.3.

| Shop Controller | | Cell Controllers | | Machine Controllers | | Simulator |
|---|---|---|---|---|---|---|
| FIX | VB | FIX | VB | FIX | VB | VB |
| 1. Auction Control<br>2. Bid-Evaluation Procedure<br>3. Auction Initialisation | 4. Database Control<br>5. Job-Checking Procedure<br>6. Product-Information Module<br>7. Auction Control | 8. Auction Control | 9. Cell Queue Control | 10. Machine Control Algorithm | 11. Machine Database | 12. Initialisation Subroutine<br>13. New Job Generator<br>14. Process Routing Generator<br>15. Report Generator |

Table 4.3: SIMCON's Control Functions

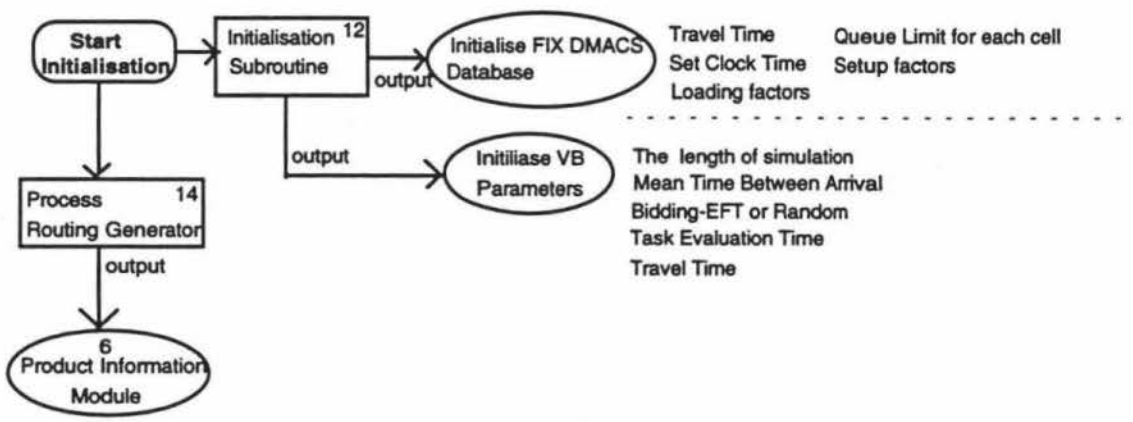


Figure 4.6a: Interaction between the SIMCON's functions - Initialisation

14. The functions written using Visual Basic are represented by a number of procedures (see section 4.3.2). The independence of an procedure leads to conceptualising the solution in terms of cooperating entities, not just one solution program. This implies that new functions can easily be added to the system to augment the existing level of functionality. For example, in the future the SIMCON package will probably be used by other researchers to test out the auction-based scheme incorporating various sequencing rules. Functions implementing these rules can easily be added to the SIMCON package.

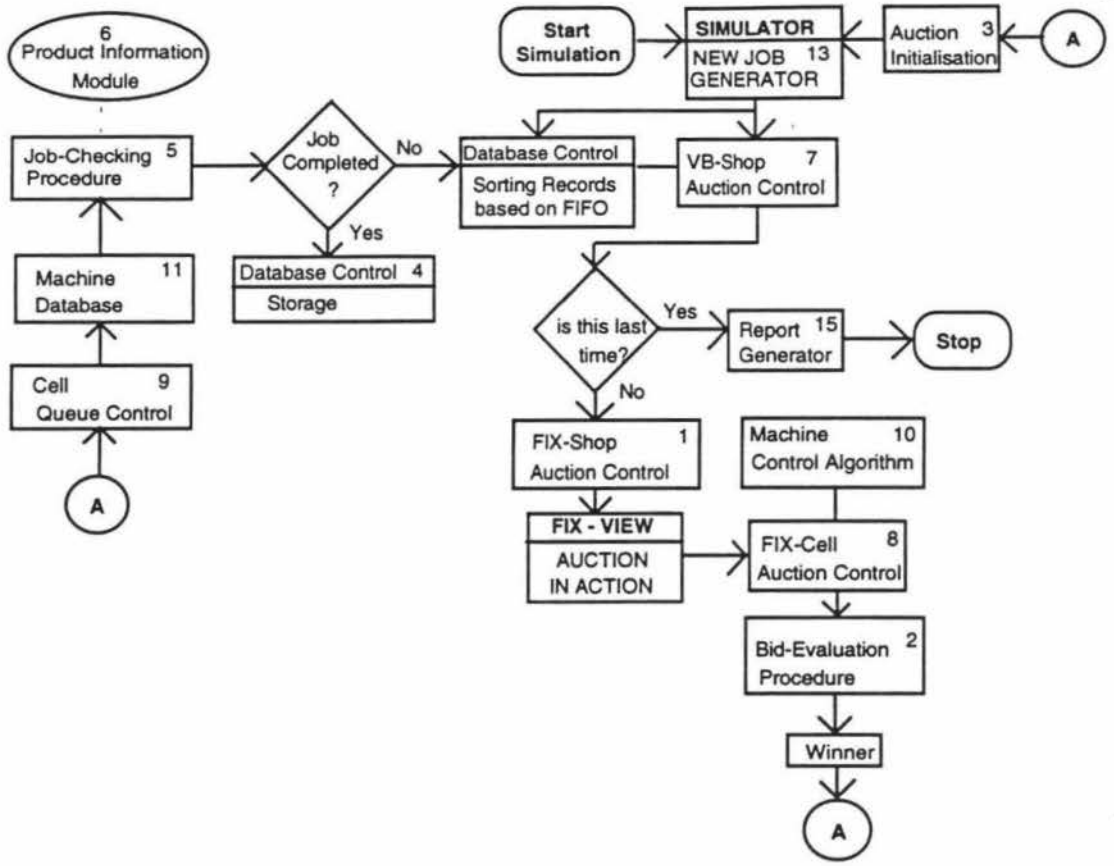


Figure 4.6b: Interaction between the SIMCON's functions - Simulation

### 4.4.1 *The Shop Controller Functions*

Altogether the shop controller has seven functions: three functions implemented within FIX DMACS and four functions written using Visual Basic (ref: Table 4.3).

The main functions of the shop controller are the *VB-Shop Auction Control*, *Fix-Shop Auction Control*, and *Database Control*. *Database Control* is a collection of information stored in an organised way. It has two tables containing records, which in turn, contain fields. It can be considered that a table as being analogous to a spreadsheet, with a record being a row and a field as a column (Potter et al. 1992).

The first table contains information concerning tasks to be auctioned, such as: Job Number, Product Type, Operation Number, Quantity, Machine Type, Setup Time,

Loading Time, Machining Time and Auctioning Time. The second table ("storage") contains information regarding completed jobs, such as: Job Number, Due Date, Product Type, Completion Time, Total Processing Time and Total Waiting Time. This table is for generating statistical performance measures at the conclusion of the simulation.

The first table accepts new jobs (a set of records) generated by the *Simulator* (*New Job Generator*) and existing jobs updated by the *Job-Checking Procedure*. It then sorts the records according to time index in ascending sequence and automatically loads the first record, the task with the earliest auctioning time, and makes it ready to be auctioned. In the mean time, the *Simulator* (*New Job Generator*) also triggers (after it generates a new job) the *VB-Shop Auction Control* function (see Figure 4.6b above). This activates the *VB-Shop Auction Control*'s code to send the first record in the first table via EDA to FIX DMACS database blocks. This, in turn, activates the *FIX-Shop Auction Control* to declare the tasks to the cell controllers. An Auction is now in action.

The second function of the *FIX-Shop Auction Control* is to display on the operator screen via FIX-DMACS View application the bidding information and other appropriate data, such as storage size, and buffer size of each cell and the shop controller. This enables monitoring of the current shop floor conditions. After the deadline for bid submission has passed, the *Bid-Evaluation Procedure* ranks the value of each bid and determines the winning cell based on the earliest time the task can be finished. The *Auction Initialisation* function then resets the FIX DMACS database blocks.

The *Job-Checking Procedure* is responsible for checking if there are any remaining tasks to be done to complete a job. If a job has been completed, this procedure updates the storage size and the Mean Time Between Completion (MTBC) statistic.

Information concerning the completed job is then sent to the *Database Control* (the second table: "storage").

If there are tasks still required to complete the job, the *Job-Checking Procedure* records the finishing time of the completed task. This will be considered as the auctioning time for the next task. By requesting information from the *Product-Information Module*, the procedure also updates appropriate machine type, setup time, loading time and machining time for the next task. All the information concerning the next task is sent to the *Database Control* (the first table: "auctioned tasks"). At its auctioning time, the task will be announced.

In addition, if there are no bids placed by the cell controllers on the current auction, the *Job-Checking Procedure* also triggers the *Database Control* to update the task-announcement packet[15]. The new auction is ready and this triggers the *VB-Shop* and *FIX-Shop Auction Control* functions to announce a "new auction packet".

### 4.4.2 *Cell and Machine Controllers Functions*

Collectively, the cell controllers have two functions: one function implemented within FIX DMACS (*FIX-Cell Auction Control*) and other function written using Visual Basic (*Cell Queue Control*), ref: Table 4.3.

The machine controllers also have two functions as a whole, one written using Visual Basic (*Machine Database*) and the other (*Machine Control Algorithm*) implemented within FIX DMACS.

With regard to an auction, the main responsibility of the cell controllers is to determine whether the auction is relevant to them or not. The *FIX-Cell Auction Control* deals

15. This means to update the first table of the Database Control.

with this (see Figure 4.7). Moreover, Figure 4.7 shows that if the cells are able to process the task, this function interacts with the *Machine Control Algorithm* to estimate the earliest time the task can be completed which, in turn, transmit the bid value to the *Bid-Evaluation Procedure*.

After the deadline for bid submission has passed, the shop controller via the *Bid-Evaluation Procedure* awards the task to the winning cell. The *Cell Queue Control* then receives information concerning the task from the *Bid-Evaluation Procedure* and sends it to the *Machine Database*. If the machine is busy, the *Cell Queue Control* updates the state of the queue at the cell.

The *Machine Database* also updates the values of any state variables related to its "machine" if necessary, such as estimated waiting time at the machine, machine's idle time, and the delay of the task in the queue.

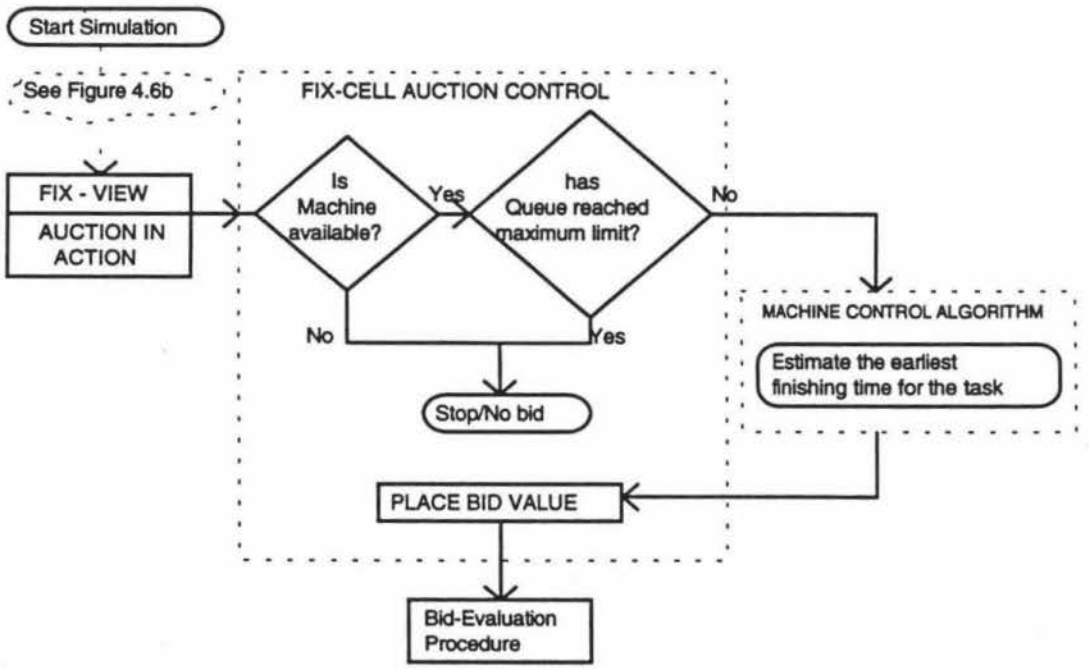


Figure 4.7: The FIX-Cell Auction Control and The Machine Control Algorithm[16]

16. Note that this figure is closely related to figure 4.6b.

## 4.5 The Simulator

Altogether the simulator has four functions and all are written using Visual Basic (ref: Table 4.3). These functions are: (1) the *Initialisation Subroutine; (2)* the *Process Routing Generator*; (3) the *New Job Generator*, and; (4) the *Report Generator*.

The *Initialisation Subroutine* function is employed to initialise the Visual Basic application's parameters as well as the FIX DMACS database blocks. The *Process Routing Generator* function is used to generate product types, including: (1) number of operations for each product type; (2) setup time, loading time, machining time, and machine type for each operation in each product type, and; (3) the sequence of operations to make up the product. Note that both functions are used to perform these specific tasks before the simulation run is started (see Figure 4.6a above).

The first function of the *New Job Generator* is to generate new jobs and send them to the *Database Control* (the first table). As previously mentioned in Section 4.4.1, the first table then sorts the records and automatically loads the first record, the task with the earliest auctioning time. The second function is to trigger the *VB-Shop Auction Control* to send the first record via EDA to FIX DMACS database blocks. This activates the *FIX-Shop Auction Control* to declare an auction. When the winner of the auction has been determined, the *Auction Initialisation* initialises the FIX DMACS database blocks which, in turn, triggers the *New Job Generator* to start the process again (see Figure 4.6b above). The same logical flow is repeated and this happens over time until reaching the end of the simulation.

In short, the *New Job Generator*, cooperates with the *Database Control* and *the VB-Shop Auction Control*, advances the simulation clock and makes the functions implemented within FIX DMACS database blocks run dynamically and automatically over time by creating a closed loop.

At the conclusion of the simulation, the *Report Generator* outputs a set of statistical performance measures by which the hybrid control architecture or cell layout may be evaluated.

## 4.6 The Interface

As previously mentioned, the *Interface* is required to enable the two software used in this research to communicate with each other. It is developed to act as a "bridge" between the simulator and the real-time control system. It facilitates real-time communication between them. The main protocols are: EDA and DDE.

All DDE conversations consist of four elements: the source, the destination, the topic, and the item (Potter et al, ibid.). In order to start the conversation, one of the two applications has to request data from the other. It is important therefore to indicate first the topic, or part of the source application, with which to establish the conversation. The actual data transferred through the interface is called the *item*. The program that request data is called the *client* (the destination), and the program that supplies the data is called the *server* (the source). FIX DMACS programs work as the server and the Visual Basic programs work as the client (see Figure 4.8 as an example). When the conversation begins, the client application can set up the link between the two programs in two ways. An *Automatic* link instructs the *server* to update the link every time the request data changes. A *manual* link exchanges information only when the *client* requests data from the *server*.

Both FIX DMACS and Visual Basic programs can work as either DDE clients or DDE servers. In this research, however, only *Automatic* links are used. *Manual* links are not used, because, based on the experiment, they are sometimes either late to up-date data (FIX DMACS database blocks are late to response) or even unable to complete the DDE conversation (FIX DMACS database blocks fail to complete their side as a

server). This causes unexpected results - run-time errors - in the Visual Basic's application. To avoid this, EDA is employed to create special functionality that establishes communication (read or write) between the FIX database and Visual Basic programs.

Information within a FIX DMACS system in an EDA conversation is referenced by Node, Tag and Field. A node contains a process database which contains many tags. Each tag refers to a database block (such as an Analog Input) which contains many fields, for example, A_CV refers to current value, and F_CV refers to floating point value. In order to read or write data to a particular field, an application must refer to the field by using a Node-Tag-Field (NTF) identifier[17].

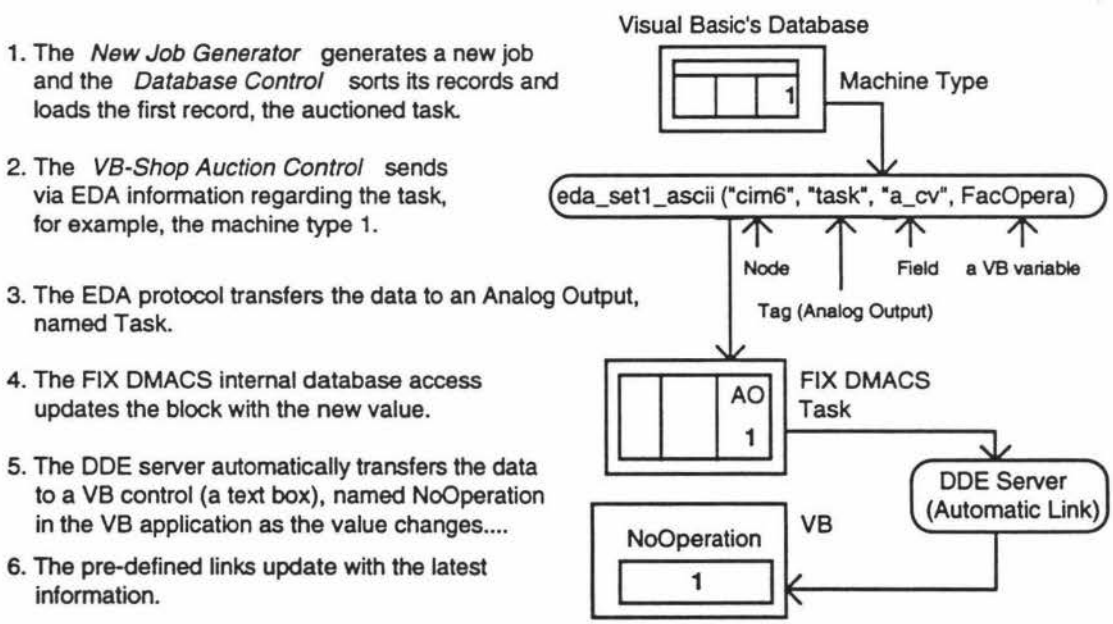


Figure 4.8: An Example of Communication between FIX & VB via The Interface

Table 4.4 describes detailed information that has to be exchanged between the two softwares. Communication is done via the *Interface*. As a summary, there are seven

17. For more information, see the FIX DMACS' *Easy Database Access Manual.*

areas. They are: bidding information, winner, no bid placed, current queue sizes, queue limit, waiting times at machines in each cell and activating FIX database blocks. For example, two areas are discussed below.

***Bidding Information***

This is concerning the first record (the task being auctioned) in the *Database Control.* The *VB-Shop Auction Control* sends this bidding information to 9 VB Controls (in this case text boxes) in the *Interface*. The *Interface* then passes this information via EDA protocol from these VB Controls to 9 FIX DMACS Database blocks (in this case Analog Output blocks, ref: Table 4.4).

When the winner of the auction has been determined, the *Interface* collects the bidding information via DDE (automatic link) from 10 FIX DMACS Analog Output blocks and sends the information to 10 Text Boxes in the *FIX-Cell Auction Control* (ref: Table 4.4).

***Waiting Time at Machines in Each Cell***

This is concerning updating estimated waiting times at each machine controller in the modelled FMS shop floor. The *Cell Queue Control* updates the waiting times for all machines and then sends the information to 13 Text Boxes in the *Interface*. The *Interface* then passes the information via EDA from these VB Controls to 13 FIX DMACS Analog Output blocks (ref: Table 4.4).

| | | DDE | | EDA | |
|---|---|---|---|---|---|
| **Functions Involved** | **Description** | **FIX Blocks (TagNames)** | **VB Controls (Text Boxes)** | **FIX Blocks Analog Outputs (TagNames)** | **VB Controls (Text Boxes)** |
| VB-Shop Auction Control | Bidding Information | | | JobNo<br>Product<br>Task<br>Quantity<br>Machine<br>TmArrival<br>SetupTime<br>Loading<br>ProcessTm | FcJobNo<br>FcProduct<br>FcOperation<br>FcQuantity<br>FcMachine<br>FcTmArrival<br>FcSetup<br>FcLoading<br>FcMachining |
| | Winner | Winner | CellWinner | | |
| | No Bid Placed | Bidqueue | FcQueue. | — | |
| Cell Queue Control | Current Queue Sizes | | | Queue1<br>Queue2<br>Queue3<br>Queue4 | QueueCell2<br>QueueCell3<br>QueueCell4<br>QueueCell5 |
| | Queue Limit | | | QueLimit1<br>QueLimit2<br>QueLimit3<br>QueLimit4 | QueueLimit1<br>QueueLimit2<br>QueueLimit3<br>QueueLimit4 |
| | Waiting Time at machines in each cell | | | C1a,b,c,d wait<br>C2b,d,e wait<br>C3a,b,c,d wait<br>C41e,42e wait | Waiting11,2,3,4<br>Waiting22,4,5<br>Waiting31,2,3,4<br>Waiting41,42 |
| FIX-Cell Auction Control | Bidding Information | JobNo<br>Product<br>Task<br>Quantity<br>Machine<br>TmArrival<br>SetupTime<br>Loading<br>ProcessTm<br>C1spt | JobNumber<br>ProductType<br>NoOperation<br>Quantity<br>CMachine<br>TmArrival<br>CSetup<br>CLoading<br>CMachining<br>Processing Time | | |
| New Job Generator | To activate FIX database | Token | SimToken | | |

Table 4.4: Information included in the Interface

## 4.7 Summary

This chapter has illustrated the implementation of the hybrid control architecture. The approach used is called SIMCON (Simulation Control). Technically it involves two

software packages: FIX DMACS for Windows and Microsoft Visual Basic for Windows. Communication is performed via an *Interface* which is written using Visual Basic. It acts as a "bridge" between these two softwares and the main protocols are EDA and DDE.

The FIX DMACS is used to construct the modelled FMS shop floor and to implement the auction-based control functions of the shop, cell and machine controllers.

The Microsoft Visual Basic is employed to simulate the arrival of jobs and run the functions implemented within FIX DMACS database blocks dynamically and automatically over time. Also, it is used to emulate the modelled shop floor by developing supporting control functions for the shop, cell and machine controllers.

Chapter 5 discusses the simulation study undertaken and the results obtained.

# CHAPTER 5

# TEST CASE ANALYSIS OF THE SIMCON

## 5.1 Introduction

The aims of this chapter are to demonstrate:

1. The modelled FMS.
2. The experimental considerations. These include: (a) the simulation input data; (b) the experimental set-up; (c) performance indicators, and; (d) the simulation method.
3. Statistical analysis of the results obtained from the simulation runs.

The primary objective of the test case analysis of the SIMCON is two fold: (1) to study the auction-based scheme for scheduling jobs in the hybrid control architecture, along with the "random" scheduling scheme for performance comparison, and; (2) to study and verify the operation of the implemented hybrid control architecture.

## 5.2 The Hypothetical FMS

As indicated in Chapter 3, the primary application area of the hybrid control architecture is in a dedicated FMS. The considered system in this test case analysis is therefore characterised by the repetitive production of a modest range of products. It has real-time, on-line control of part production and consists of different types of machines.

Figure 5.1 illustrates the conceptual layout of the cells. The FMS consists of a shop controller and four cells. Five different machine types are defined and neither cells 1, 2, or 3 has more than one of a particular type of machine in it. Cell 4 contains two

identical machines. Cells 1, 2, and 3 have 4, 3, and 4 machines, respectively[1]. In effect, the shop floor is a network of *four* multiserver queues. Each cell is categorised as a full intelligent entity and assumed to be equipped with the required tool magazines, pallets, fixtures and handling equipment to perform the processing operations. Programs are loaded into each cell for particular operations. As far as scheduling is concerned, no cell has greater importance than any other cell and the shop controller acts as a centralised auctioneer.

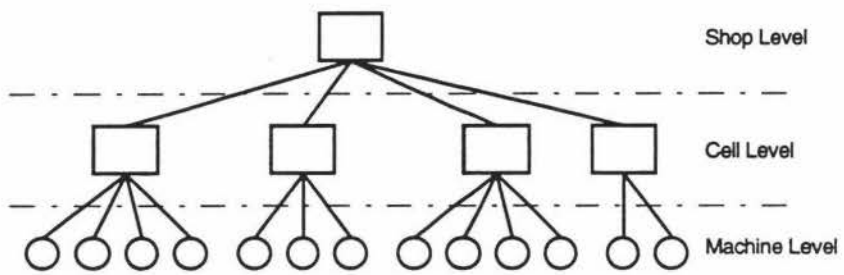


Figure 5.1: The Hybrid Control Architecture

## 5.3 Simulation Input Data

Jobs are the entities to be scheduled. Each job corresponds to a particular product type with a certain batch size. For the products to be completed, they require several operations on different machine-tools. The individual production operations which make up a product are reffered to as *tasks*.

In addition, the tasks considered are computer-controlled and processed by CNC machine tools, setups between consecutive tasks are automated, so the time to perform an operation are "nearly" deterministic (Kim et al., 1994). Processing times are calculated at particular machines by using the following formula:

1. In their experiment, Arzi and Roll (op.cit.) considered a flexible manufacturing cell which consisted of 4 machines. Another researcher, such as Stoeva (1990) considered 3 cells in his experiment: cell 1, 2 and 3 consisted of 1, 1, and 2 machines, respectively. Baker and Dzielinsky (1960) investigated shops with different numbers of machines and concluded that the size of the shop did not significantly affect the relative performance of the scheduling rules.

$$PT = ((Stf \times ST) + (((LT \times Ltf) + MT) \times Q)), \quad (5.1)$$

where:

| | | |
|---|---|---|
| PT | = | Processing time of an operation at a particular machine |
| Stf | = | Setup factor for a particular machine[2]. |
| ST | = | Setup time for an operation[3] |
| LT | = | Loading time for an operation |
| Ltf | = | Loading factor for a particular machine[4] |
| MT | = | Machining time |
| Q | = | Quantity (Batch size) |

The bidding message which contains the estimated earliest finishing time is calculated at particular machines as follows:

$$EFT = PT + WT + TT, \quad (5.2)$$

where:

| | | |
|---|---|---|
| EFT | = | Earliest Finishing Time that a task can be finished. |
| WT | = | Estimated Waiting Time. |
| TT | = | Transportation Time. |

The time required by the shop controller to reach the scheduling decision is denoted by EstDuration[5]. It is calculated as follows :

$$\text{EstDuration} = (\text{Communication Delay} \times 2) + \text{Task Evaluation Time.} \quad (5.3)$$

Other simulation input data is described below:

1. The number of operations for each product was generated from a U[1,6] distribution (see Appendix B).

2. The sequence of operations to complete a product was randomly generated before the simulation was run.

3. Loading/unloading times, setup times and machining times were generated for each operation from exponential distributions with means of 0.25 hours, 0.2 hours and 0.15 hours, respectively (see Appendix B).

---

2. Each cell can have several different set-ups for different families of tasks (see Appendix B).
3. This could include: cutting tool preparation time, part positioning and releasing time, and NC program changeover time (Rachamadugu et al., op.cit.).
4. Each cell can have several different loadings for different families of tasks (see Appendix B).
5. The time between the declaration of an auction and it being assigned to a cell for processing (Shaw, 1987).

4. The travel times between cells or between a cell and the shop controller are equal to 0.0972 hours.

5. The estimated total processing time for each product was the sum of the estimated machining times for its operations. In this case, it is assumed that Stf, Ltf and Q are one.

6. Queue capacity for the cells and the shop controller were 15 and 250 jobs respectively

7. Due dates are established for each job as follows: (5.4)

   Due Date = TNOW + (1.3 x Quantity x Estimated Total Processing Times) + (No.Of. Operation x (EstDuration + Transportation Time)) + Transportation Time

   TNOW is the time when the job arrives (Shaw, ibid.).

8. Batch sizes are generated from a U[5,10] distribution.

9. New jobs are generated by the simulator to arrive in batches at the shop controller. Job interrarival time is exponentially distributed. If a job arrives at a particular cell and finds the machine in that cell already busy, the job joins a single FIFO queue at the cell.

10. It is assumed that no machines breakdowns.

## 5.4 Measures of Performance

The response variables gathered from the simulation runs are as follows (Conway et al., 1967, pp.11-21):

**1. Shop Utilisation**

**a.** *Individual Machine Utilisation* ($U_{cj}$)

$$U_{cj} = (\frac{LOS - TI_{cj}}{LOS}) \times 100, \qquad (5.5)$$

where $LOS$ indicates length of simulation, $TI_{cj}$ is the total idle time of machine $j$ in cell $c$.

**b.** *Average Utilisation of the Machines in a Cell* ($U_c$)

$$U_c = \frac{\sum_{j=1}^{M_c} U_{cj}}{M_c} \tag{5.6}$$

where $M_C$ is number of machines in cell $c$.

**c.** *Average Utilisation of the Modelled FMS* ($U^{fms}$)

$$U^{fms} = \frac{\sum_{c=1}^{C} U_c}{C} \tag{5.7}$$

where $C$ is number of cells.

**d.** *Average Utilisation of Each Type of Machines* ($U_k$)

$$U_k = \frac{\sum_{k=1}^{M_k} U_{ck}}{M_k}, \tag{5.8}$$

where $M_k$ is total number of machines of type $k$ in the modelled FMS.

## 2. Average Waiting Time, exclusive of processing time ($\overline{W}$) :

$W_{ij}$ is the waiting-time of job i at the j$^{th}$ operation after completion of the (j - 1)$^{th}$ operation. The total waiting time for a job is the sum of the waiting times for all operations of the job:

$$W_i = \sum_{j=1}^{g_i} W_{ij} \tag{5.9}$$

where $g_i$ is the total number of operations to complete job i

The Average Waiting-Time is therefore:

$$\overline{W} = \frac{\sum_{i=1}^{n} W_i}{n} \tag{5.10}$$

where $n$ is the number of jobs completed.

## 3. Mean Flow Time ($\overline{F}$ )

$$\overline{P} = \frac{\sum_{i=1}^{n} P_i}{n} \tag{5.11}$$

where $P_i$ is the total processing time for job i and $\overline{P}$ is the average processing time.

$F_i$, the flow-time of job i, is the total time that the job spends in the shop. Mean Flow Time is then:

$$\overline{F} = \overline{W} + \overline{P} \tag{5.12}$$

**4. Average Lateness ($\overline{L}$)**

Let $C_i$ = Completion Time of job i, the time at which processing of the last operation of the job is completed. Let $D_i$ = the time at which some external agency would like to have the job leave the shop. Lateness $L_i = |C_i - D_i|$, where $L_i$ considers the absolute difference for each job, regardless of the sign of the difference. Average Lateness is then:

$$\overline{L} = \frac{\sum_{i=1}^{n} L_i}{n} \tag{5.13}$$

**5. % of Jobs Late :**

This is the percentage of Jobs late. It is calculated as follows:

$$\frac{[Jobs | L_i > 0]}{n} x\ 100\% \tag{5.14}$$

**6. Mean Tardiness ($\overline{T}$)**

Tardiness considers only positive differences - jobs which are completed after their due-date. Mean Tardiness is measured by:

$$\overline{T} = \frac{\sum_{i=1}^{n} T_i}{n} \tag{5.15}$$

where $T_i = \begin{cases} 0 & \text{if } C_i < D_i \\ C_i - D_i & \text{if } C_i \geq D_i \end{cases}$.

## 5.5 Experimental Considerations

Important considerations in experimental design are: the starting and stopping conditions of the simulation study, estimation of when steady state begins, and the sample size of the simulation observations.

### 5.5.1 *Definitions*

A simulation ***run*** is an uninterrupted recording of the system's performance under a specified combination of controllable variables (Emshoff et al., 1970). An ***observation*** of the simulated system is a segment of a simulation run (rather than an instant in the run) sufficient for estimating the value of each of the performance measures (Emshoff et al., ibid.).

### 5.5.2 *Starting and Stopping Conditions*

There are two states usually described in a simulation study: transient state and steady state. The transient state generally refers to the start-up or initial conditions of the simulation run. The simulation is started with an empty and idle facility. The initial part of the simulation run is atypical of the system's true operation (Graybeal et al., 1980). There is a transient period until the effect of this initial conditions become insignificant. Normally, it takes some time for the effect of the initial conditions to become insignificant and for the simulation model to stabilise, or reach *steady state*. Thus the system is said to be in the steady-state conditions when the initial conditions do not longer affect the variabilities inherent in the simulation model and successive observations of the system's performance are statistically indistinguishable (Emshoff et al., op.cit.).

The analysis of the simulation results under steady-state condiitions is desirable, because it is normally under these conditions that the true system's operation takes place. Since transients do occur in the initial part of the simulation run, it is important therefore to remove the biasing effect of statistics generated during this period before starting collecting data. Emshoff and Sisson (ibid.) state that there are no *fixed rules* for determining when steady state can be assumed. This indicates that, based on a certain performance measure and control variables, an educated guess is required to judge

when the simulation is in steady state. There are two methods commonly used to remove the effects of transients (Emshoff et al., ibid.).

The first method is to start the simulation with an empty and idle facility and run a long simulation so that the data from the transient period is insignificant relative to the data in the steady state. When the simulation has reached steady state, data gathered during the transient phase is discarded and the collection of the statistics begun anew[6]. This method is not difficult to arrange, but it is costly in terms of computer running time (Hoover et al., 1989). Moreover, if two of more simulation runs have to be conducted, two other problems exist, that is, repeating an initial warm-up of the simulation and uncertainty of the length of the transient phase. It would be ideal therefore if the initial conditions could be selected to correspond to the conditions when the system is in steady state. However, as Schroer, Black and Zhang (1985) state, this is generally not available.

The second method is to run a long simulation but periodically record and reset the statistical measures. The accumulated statistics during the first few intervals are generally cleared (but leaving the state of the simulated system as it is), to allow for the system to pass from transient to steady state. The ending state of this period is the starting state of the first interval (observation) under steady state. Subsequently, the ending state of observation j is the starting state of observation j+1. At the end of each simulation observation, all statistical accumulations are cleared, but all of the historical data on the activities that occurred during the system simulation are maintained. This involves storing the data on the left-over jobs. For example, the tasks that were not finished processing and the number of tasks left in the queues[7]. At this stage the system can be prepared for the next simulation observation.

---

6. Obviously the analyst must be able to estimate whether the simulation has reached steady state or not, based on a certain performance measure and control variables.
7. These have to be taken into consideration in the next simulation observation so that the system can do realistic capacity and material checks.

In this research, the second method described above is employed to study the steady-state characteristics of the hybrid control architecture. In brief, the SIMCON package was started with empty and idle conditions. It was run until steady state conditions were reached, and then the statistical measures were reset and the state of the simulated system was left as it was. One long simulation was then performed and the statistics were periodically recorded and reset. The statistical resettings were based on the passage of a certain number of simulated units of time. By using this approach, no limit was placed on the number of events occurring. At the completion of the run, events were still in the system and resources were being used. The time period for resetting the statistical measures in this research was after every 250 hours.

The next section discusses in some detail how the steady state point was identified, beyond which the transient effects were insignificant.

### 5.5.3 *Steady State Estimation*

A performance measure used to estimate whether the model has reached steady state or not, is Average Queue Length (AQL) and is calculated as follows (Hurley, 1992, p.86):

$$AQL_{jt_n} = \frac{\sum_{i=1}^{n} AQL_{jt_i}(t_i - t_{(i-1)})}{t_n} \qquad (5.16)$$

where $AQL_{jt}$ is the average queue length of the single queue facing cell j at time t.

Control variables used in attempting to achieve steady state are mean time between arrival (MTBA) and mean time between completion (MTBC). The MTBC is calculated as follows:

$$MTBC_k = \frac{(MTBC_{(k-1)} \times (k-1)) + (t_k - t_{k-1})}{k} \qquad (5.17)$$

where: k = number of jobs just completing;
$t_k$ = time of the current job completion event;
$t_{k-1}$ = time previous job completed processing, and;
$MTBC_{(k-1)}$ = the MTBC for the jobs that had completed processing before the current one being considered.

The process of finding steady state was as follows:

1. The SIMCON package was started with all queues empty and all machines idle. The MTBA was set at 1 hour. After the queues had built to a "desired level", the MTBA was reset greater than the MTBC, that is, to 2 hours. This occurred after 65 hours when the MTBC was 1.95 hours.

   The idea to increase the MTBA greater than the MTBC was to build queues slowly and jobs had time to get through system before the queues became full.

2. Starting from 125 hours, the MTBA was reset equal to MTBC and at each event, the AQL[8] was monitored. The idea was not to build or decrease queues and to find "desired levels" for the AQL and the MTBC. At about 900 hours, the MTBC and the AQL reached the "desired levels", that is, 1.54 hours and 13.60 jobs, respectively.

   The MTBA was then set at a fixed value (1.49 hours).

3. To reduce/eliminate the bias in the performance measures introduced during the transient phase, the SIMCON package was run up to 2650 hours.

4. After 2650 hours, all statistical accumulations (but leaving the state of the simulated system as was) were then cleared. The conditions at the end of this period became an *a priori* estimate of the steady state conditions and, in effect, were used to start the first observation.

A moving average of the performance measure (AQL) was calculated and graphed (see Figure 5.2). By observing the trend, it can be assumed that steady state had been reached at 1500 hours, since successive computations no longer vary significantly. The standard deviation of the moving average of the AQL can be seen in Table 5.1. The standard deviation of the sample is defined by:

8. Note that the AQL in this case was the AQL for all queues.

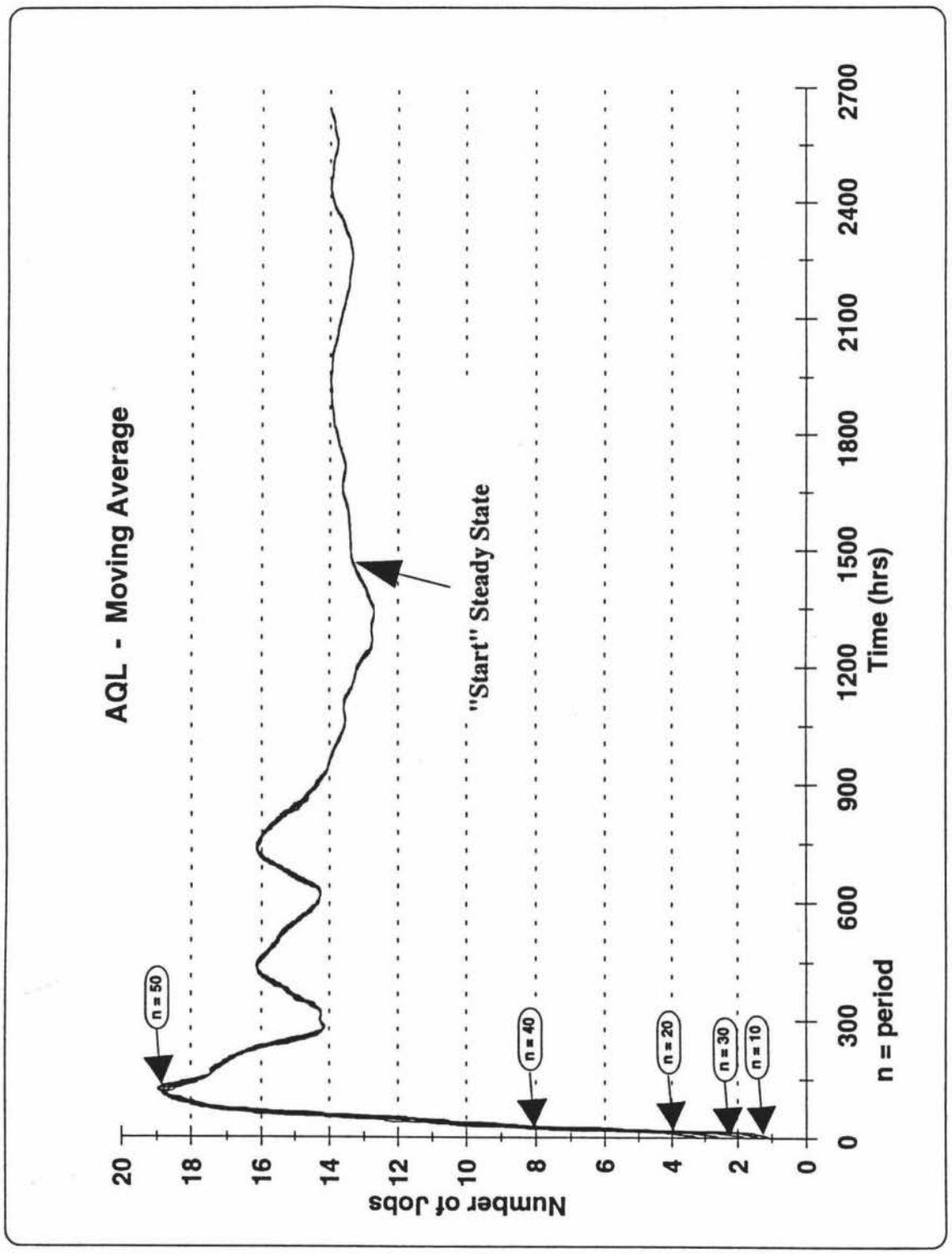


Figure 5.2: Moving Average "Average Queue Length"

$$s = \sqrt{\frac{\sum (X - \overline{X})^2}{n-1}} \tag{5.18}$$

where X = AQL , $n$ = sample size and $\overline{X}$ = the mean AQL from a sample of $n$.

The coefficients of variation of the "steady state" and the "transient state" are computed (ref: Table 5.1). It appears that the CV of the "transient state" has a greater dispersion than in the "steady state". This indicates that from 1500 simulated hours the model is judged to be in steady state.

| Parameters | Simulation Run | Moving Average Transient State | | | | |
|---|---|---|---|---|---|---|
| | 0 - 1500 hrs | n=10 | 20 | 30 | 40 | 50 |
| $\overline{X}$ | 14.2819 | 14.2873 | 14.2939 | 14.3008 | 14.3078 | 14.3151 |
| $s$ | 1.0780 | 1.0801 | 1.0823 | 1.0843 | 1.0861 | 1.0878 |
| CV | 7.548% | 7.560% | 7.572% | 7.582% | 7.591% | 7.599% |
| Sample size | 2098 | 2098 | 2098 | 2098 | 2098 | 2098 |
| | | Steady State | | | | |
| | 1500 - 2650 hrs | n=10 | 20 | 30 | 40 | 50 |
| $\overline{X}$ | 13.7118 | 13.7105 | 13.7091 | 13.7077 | 13.7063 | 13.7049 |
| $s$ | 0.2019 | 0.2019 | 0.2020 | 0.2020 | 0.2020 | 0.2021 |
| CV | 1.4725% | 1.4728% | 1.4731% | 1.4735% | 1.4739% | 1.4744% |
| Sample size | 2098 | 2098 | 2098 | 2098 | 2098 | 2098 |

Table 5.1: Coeficient of Variation for the AQL in the Transient & Steady States

### 5.5.4 *Sample size*

Fishman's procedure[9] is used to test the independence of each observation under steady state. If $AQL_j$ denotes the Average Queue Length of the jth observation after reaching steady state and N the number of observations, then the grand mean is:

$$\overline{AQL} = \frac{\sum_{j=1}^{N} AQL_j}{N} \tag{5.19}$$

9. Fishman, 1968 in Schroer et al., op.cit., pp.67-68.

Furthermore, since all statistical accumulations of the observation j were cleared before starting the observation j+1, then a reasonable assumption is that the $AQL_j$'s are independent and normally distributed (a reasonable approximation), it then follows that the test statistic:

$$c = 1 - \frac{\sum_{j=1}^{N-1} (AQL_j - AQL_{j+1})^2}{2\sum_{j=1}^{N} (AQL_j - \overline{AQL})^2} \tag{5.20}$$

is approximately normally distributed with the mean of zero and variance equal to:

$$\frac{(N-2)}{(N^2-1)} \tag{5.21}$$

The hypothesis of randomness is then: $H_0$: $c = 0$ (*independent observations*)

$H_1$: $c \neq 0$ (*correlated observations*)

$H_0$ will be rejected in favour of $H_1$ if the absolute value of

$$Z = \frac{c}{\sqrt{\frac{N-2}{N^2-1}}} \tag{5.22}$$

is greater than $Z_{\alpha/2}$, where $Z_{\alpha/2}$ is the upper $\alpha/2$ point on the standard normal distribution.

As described in section 5.1, the simulation is not only intended to study the auction-based scheme, but also to compare the results obtained to the performance of a random dispatching rule.

The random dispatching rule works by randomly selecting the machine to process the task. Specifically, when a task is ready to be processed, the shop controller checks to see the machine type required to perform the task. For example, the machine type

required be type 1. In the modelled FMS, only cell 1 and cell 3 have machine type 1. The shop controller randomly assigns the task to either cell 1 or cell 3.

There is no autonomy delegated to cell and machine controllers. Their roles are just updating, if necessary, the values of any state variables related to their "territory", such as: (a) estimated waiting time at the machine; (b) machine's idle time; (c) the delay of the task in the queue, and; (d) the estimated starting time of the task as well as its finishing time.

The first observations for these two scheduling approaches were started at time 2650 hours. Eleven additional observations of length 250 hours were then run. Table 5.2 records the AQLs for both scheduling approaches.

The calculation of the absolute value of Z (see equation 5.22) as a function of increasing number of observations is given in Table 5.3. If a 95% confidence level is assumed, then $H_0$ will be rejected if $Z > 1.645$ or $Z < -1.645$. For all observations, Table 5.3 shows that the Z is greater than -1.645 and less than 1.645 for both scheduling approaches. Therefore, $H_0$ cannot be rejected in favour of $H_1$ for both of the scheduling approaches.

| Observation | The Hybrid | Random |
|---|---|---|
| 1 | 11.30 | 20.25 |
| 2 | 17.45 | 22.95 |
| 3 | 8.06 | 24.20 |
| 4 | 7.80 | 25.46 |
| 5 | 21.57 | 21.81 |
| 6 | 11.76 | 17.21 |
| 7 | 10.70 | 26.59 |
| 8 | 25.21 | 27.57 |
| 9 | 11.94 | 38.35 |
| 10 | 12.79 | 50.11 |
| 11 | 13.33 | 52.44 |
| 12 | 15.40 | 44.24 |

Table 5.2: Average Queue Length

| | The Hybrid | | | Random | | |
|---|---|---|---|---|---|---|
| N | $\overline{AQL}$ | $c$ | $Z$ | $\overline{AQL}$ | $c$ | $Z$ |
| 3 | 12.27 | -1.37 | -0.83 | 22.47 | -0.48 | -0.29 |
| 4 | 11.15 | -1.03 | -0.53 | 23.22 | -0.32 | -0.16 |
| 5 | 13.24 | -1.07 | -0.48 | 22.93 | -0.69 | -0.31 |
| 6 | 12.99 | -1.38 | -0.56 | 21.98 | -0.50 | -0.21 |
| 7 | 12.66 | -1.34 | -0.51 | 22.64 | -1.06 | -0.40 |
| 8 | 14.23 | -1.07 | -0.38 | 23.26 | -0.80 | -0.28 |
| 9 | 13.98 | -1.35 | -0.45 | 24.93 | -0.44 | -0.15 |
| 10 | 13.86 | -1.34 | -0.43 | 27.45 | -0.23 | -0.07 |
| 11 | 13.81 | -1.34 | -0.41 | 29.72 | -0.14 | -0.04 |
| 12 | 13.94 | -1.34 | -0.39 | 30.93 | -0.14 | -0.04 |

Table 5.3: Calculation of the Z statistic

Thus, based on the Fishman's procedure, the hypothesis that the 12 observations for each scheduling approach are a sequence of independent random variables can be accepted.

## 5.6 Simulation Results and Analysis

It should be noted that, the SIMCON package was actually run to study the auction-based scheme employing the *bidding-SPT* (Shortest Processing Time) to calculate the bids[10]. However, the simulation was stopped after the second observation for the following reason:

As indicated in section 5.3, each machine in each cell has its own setup and loading factors generated before the first observation was run and they varied from machine to machine. This fact affected the calculation of the processing time for the auctioned task (see equation 5.1). Even though the batch size was also considered in this calculation, it was most likely that, based on the experiment, machines with lower setup and loading factors always won the auction. The single queue of the wining cell built up quickly, and not suprisingly, queues of particular cells were always full over the simulated time.

10. The calculation is calculated just based on the processing time required to complete a task. It does not consider the transportation time and estimated waiting time.

Such a circumstance also occurred in the second observation. Thus it is intuitively obvious that the *Bidding SPT* does not perform better than the *Bidding-EFT*[11].

### 5.6.1 *Data Collection*

Table 5.4 shows the statistical measurements resulting from twelve independent observations for each scheduling approach. $\overline{X}(n)$ is a (point) estimate of E(X)[12]. Let $X_j$ be the value of X (for example, average flow times) from the $j^{th}$ of n observations (n = 12 here), then $\overline{X}(n)$ is (Kelton, ibid., p.79):

$$\overline{X}(12) = \frac{X_1 + X_2 + ... + X_{12}}{12} \tag{5.23}$$

and an *unbiased* estimate of the standard deviation of $\overline{X}(n)$ is:

$$s(n) = \sqrt{\frac{1}{n(n-1)} \sum_{j=1}^{n} [X_j - \overline{X}(n)]^2} \tag{5.24}$$

A 95% confidence interval for E(X) is then:

$$\overline{X}(n) \pm t_{n-1,0.975} s(n) \tag{5.25}$$

where $t_{n-1,0.975}$ is the upper 0.975 critical point from the t distribution with n - 1 degrees of freedom. Using these equations for the mean flow time from the "hybrid" part of Table 5.4, for example, leads to $\overline{F}(12)$ = 34.14 simulated hours, s(12) = 1.79, and $t_{11,0.975}$ = 2.201 leads to a 95% confidence interval of 34.14 ± 3.94 or, written as an interval, [30.20, 38.08].

---

11. The "full" 12 simulation observations may be done to study the Bidding-SPT. However, in order to obtain real results, the simulation input data regarding the setup and loading factors for each machine described in section 5.3 has to be reconstructed. Instead of determining them in advance, it may be better if they are generated - not before scheduling - but over simulated time.
12. Kelton (1986) states that "E(X) is the average observed value of X over *infinitely* many simulation runs; and since we can't make that many, we must settle for an interval which, with 95% confidence, contains E(X)"(p.79). A synonym for E(X) is mean (Kleijnen, 1987).

With reference to Figure 5.3, which is the graphing of the means in Table 5.4, it appears that the hybrid approach perfoms significantly better than the random approach. Table 5.4 also shows that the confidence intervals of the random approach for all performance criteria are wider than those with the hybrid approach. Moreover, under the random approach unbiased estimates of the standard deviation of all $\overline{X}(n)$ are generally higher than in the hybrid approach. Considering these figures, it can be said that the bidding EFT approach performed statistically "better" than the random approach.

| | **The Hybrid** | | | **Random** | | |
|---|---|---|---|---|---|---|
| Measures of Performance | $\overline{X}(n)$ | s(n) | Confidence Interval | $\overline{X}(n)$ | s(n) | Confidence Interval |
| Mean Waiting Time (MWT) | 20.25 | 1.80 | 20.25 ±3.96 | 43.82 | 4.68 | 43.82 ± 10.3 |
| Mean Processing Time (MPT) | 13.88 | 0.07 | 13.88 ± 0.16 | 14.31 | 0.12 | 14.31 ± 0.26 |
| Mean Flow Time (MFT) | 34.14 | 1.79 | 34.14 ± 3.94 | 58.13 | 4.66 | 58.13 ± 10.26 |
| Mean Lateness (ML) | 4.41 | 1.80 | 4.41 ± 3.97 | 29.41 | 5.34 | 29.41 ± 11.76 |
| Number of Late Jobs (LJ) | 98.42 | 11.21 | 98.42 ± 24.66 | 152.5 | 7.02 | 152.5 ± 15.46 |
| % Job Late (%JL) | 56.95 | 5 | 56.95 ± 12.05 | 86.57 | 3 | 86.57 ± 6.43 |
| Mean Tardiness (MT) | 7.63 | 1.40 | 7.63 ± 3.07 | 30.36 | 5.15 | 30.36 ± 11.32 |
| Mean Queue Length (jobs) | | | | | | |
| * Cell 1 | 4.97 | 0.52 | 4.97 ± 1.15 | 9.41 | 0.75 | 9.41 ± 1.64 |
| * Cell 2 | 3.78 | 0.40 | 3.77 ± 0.88 | 6.09 | 1.26 | 6.09 ± 2.78 |
| * Cell 3 | 3.80 | 0.54 | 3.80 ± 1.20 | 10.93 | 0.80 | 10.93 ± 1.76 |
| * Cell 4 | 1.80 | 0.37 | 1.79 ± 0.81 | 4.31 | 0.82 | 4.31 ± 1.80 |
| Jobs Completed (Jobs) | 169.42 | 3.77 | 169.42 ± 8.29 | 175.5 | 3.07 | 175.5 ± 6.75 |

Table 5.4: Measures of Performance

## *Differences Analysis*

To verify that there are statistically differences between the random and the hybrid approaches, the differences on the average flow times are calculated. The average flow time is chosen, since it is a representative of the measures of performance used.

Let $D_j$ be the difference between the $j^{th}$ observation of the random and the hybrid approaches. Thus, $D_1 = 43.11 - 30.84 = 12.27$, and so on (ref: Table 5.5).

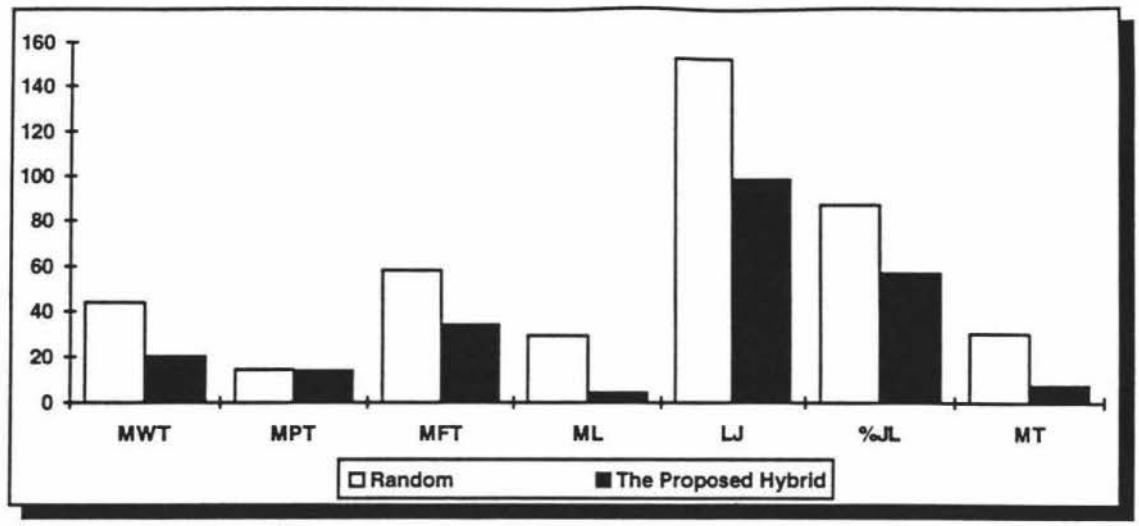


Figure 5.3: Measures of Performance (Mean)[13]

Table 5.5 shows the average of the $D_j$'s to be 23.99, with a standard deviation of 4.91. Proceeding as before (using $t_{11,0.975} = 2.201$) leads to a 95% confidence interval of 23.99 $\pm$10.80, or [13.19, 34.79]. This is a confidence interval for the difference between the expected average flow time for the random approach and that for the hybrid approach. The important thing to notice is that the interval does not cross zero. The interpretation is that there is statistically significant difference between these two approaches (at the 5% level). Thus the hybrid approach perfoms significantly better than the random approach.

Based on the availability of machines in each cell, the shop controller in the random approach just randomly selected the cell to perform a task, without considering the number of tasks already waiting for processing in the queues. Insensitivity of the shop controller to the real "workload" in each cell caused an uneven distribution of tasks in the queues and in turn created variability in the overall performance. Not suprisingly, the number of late jobs indicated in Table 5.4 was higher than in the hybrid approach[14].

13. See table 5.4 for the the meaning of MFT, MPT, MFT, etc
14. In addition, the "random" confidence interval for this parameter is lower than in the hybrid approach (ref: Table 5.4). This means less individual observations vary from the mean than in the hybrid approach.

| Observations | Random | The Hybrid | $D_i$ |
|---|---|---|---|
| 1 | 43.11 | 30.84 | 12.27 |
| 2 | 44.79 | 37.68 | 7.11 |
| 3 | 53.04 | 26.86 | 26.18 |
| 4 | 50.72 | 25.29 | 25.43 |
| 5 | 49.05 | 39.50 | 9.56 |
| 6 | 37.25 | 35.98 | 1.27 |
| 7 | 51.06 | 28.52 | 22.54 |
| 8 | 55.60 | 46.95 | 8.65 |
| 9 | 67.50 | 32.09 | 35.41 |
| 10 | 85.62 | 32.44 | 53.18 |
| 11 | 82.79 | 33.70 | 49.09 |
| 12 | 77.03 | 39.80 | 37.23 |
| | | $\overline{D}(12)$ | 23.99 |
| | | s(12) | 4.91 |
| | | $t_{n-12,0.975}s(n)$ | 10.80 |
| | | $\overline{D}(12) \pm t_{n-12,0.975}s(n)$ | [13.19, 34.79] |

Table 5.5: Differences between the "Hybrid" and "Random" for the Average Flow Time

The auction-based scheme performs better than the random scheduling method because, by executing the bidding mechanism, the scheduling decision was achieved by the shop controller based on the actual workload within each cell. The cell with the lowest workload (the queue with the shortest operation time) would always win the auction. In a broad sense, this in turn directly led to forming an even distribution of tasks waiting in the queues (see Figure 5.4).

Since the simulation results obtained for the hybrid approach are markedly better than the random approach, the next section focuses primarily on studying the behaviour of the hybrid control architecture and the characteristics required for actual implementation. Detailed simulation results obtained for both scheduling methods can be found in Appendices C and D.

### 5.6.2 *Data Analysis for the Hybrid Control Architecture*

#### *Utilisation Analysis*

Figure 5.5 shows the utilisation of each machine type, the highest was 87% and the lowest was 41%. The overall FMS shop utilisation ($U_{fms}$) was 74%[15] on a 95% confidence interval [71, 77] (ref: Table 5.6). Meanwhile, Table 5.4 indicates that the mean production was 169.42 jobs. This says, roughly, that the shop was utilised between 71% and 77%, with 95% confidence to complete 169.42 jobs during a 250-hour production. From simply looking at these figures, it can be seen that the mean production should apparently be increased to achieve greater overall shop utilisation, but this would add to the level of WIP.

Figure 5.4 indicates that the mean queue lengths for cells 1, 2, 3 and 4 were 4.97, 3.78, 3.80 and 1.80 jobs[16], respectively. Looking at the statistics in Table 5.6 and Figure 5.6, it appears that most of the tasks were waiting for processing at machine types 1, 2, 3 and 5. The mean time a task spent queuing for processing at machine type 4 was less than 1 hour. The mean utilisation of machine type 4 was only 0.41 (Figure 5.5) which affected the overall figure for $U_{fms}$. Without considering machine type 4, $U_{fms}$ is 82.50% as opposed to 74%.

Since CNC machines for an FMS are expensive, it is important that the overall shop utilisation high. From simply looking at these figures it can be said that the number of machines of type 4 should apparently be reduced to achieve greater overall shop utilisation.

---

15. $(0.87 + 0.85 + 0.75 + 0.41 + 0.83)/5 = 0.74$
16. Table 5.4 indicates that the 95% confidence intervals for the mean queue lengths are not wide.

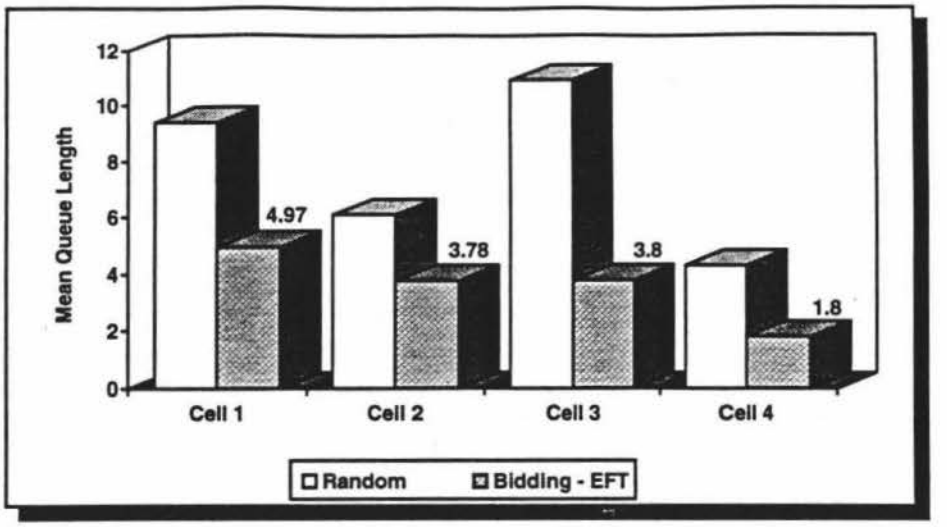


Figure 5.4: Mean Queue Lengths (Jobs)

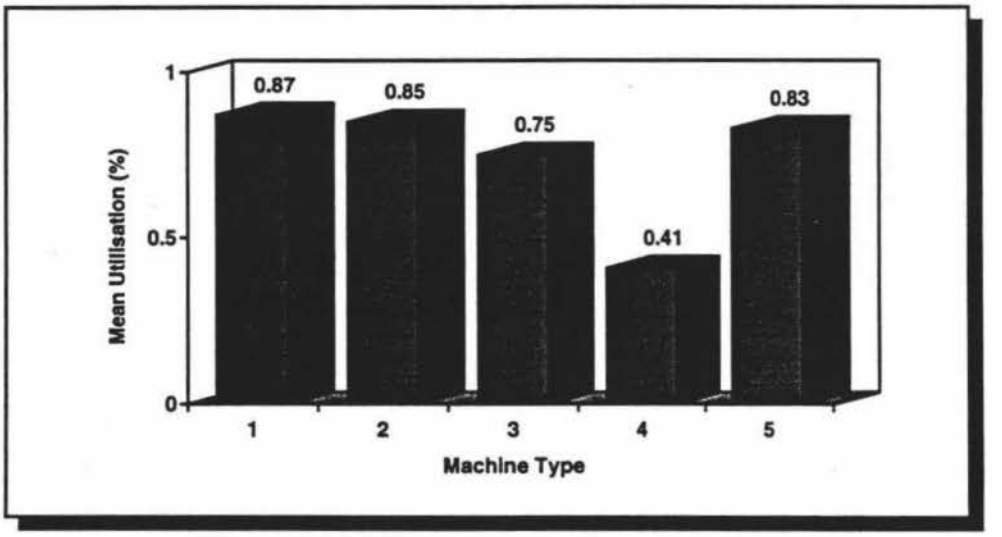


Figure 5.5: Individual Machine Type Utilisation

Thus there are two alternatives to improve the overall shop utilisation. First, decreasing the MTBA (increasing the arrival rate) and second, reducing the number of machines. Decreasing the MTBA is however rather tentative, since the random nature of the arrival rate makes it difficult to assign an exact rate and control the level of shop utilisation (Ramasesh, 1990; Jones, 1973). In addition, decreasing the MTBA would lead to the formation of large queues, particularly if the MTBA is substantially less than

the MTBC[17]. This in turn would cause higher flow times and work-in-process inventories and eventually lead the system to become unstable.

| Parameters | $\overline{X}(n)$ | s(n) | 95% Confidence Interval |
|---|---|---|---|
| **Machine Type Utilisation** | | | |
| Machine Type 1 | 87% | 1 | [ 84, 90 ] |
| Machine Type 2 | 85% | 2 | [ 81, 90 ] |
| Machine Type 3 | 75% | 2 | [ 71, 79 ] |
| Machine Type 4 | 41% | 1 | [ 39, 44 ] |
| Machine Type 5 | 83% | 2 | [ 79, 87 ] |
| Overall | 74% | 1 | [ 71, 77 ] |
| **Waiting Time at Machines** | | | |
| Cell 1 | | | |
| Machine Type 1 | 5.75 | 0.62 | [ 4.39, 7.11 ] |
| Machine Type 2 | 7.46 | 1.18 | [ 4.85, 10.06 ] |
| Machine Type 3 | 2.17 | 0.22 | [ 1.68, 2.66 ] |
| Machine Type 4 | 0.60 | 0.05 | [ 0.49, 0.71 ] |
| Cell 2 | | | |
| Machine Type 2 | 5.18 | 0.96 | [ 3.08, 7.29 ] |
| Machine Type 4 | 0.31 | 0.07 | [ 0.15, 0.47 ] |
| Machine Type 5 | 7.52 | 0.68 | [ 6.02, 9.03 ] |
| Cell 3 | | | |
| Machine Type 1 | 5.51 | 0.91 | [ 3.52, 7.50 ] |
| Machine Type 2 | 4.03 | 1.17 | [ 1.45, 6.60 ] |
| Machine Type 3 | 3.22 | 0.22 | [ 2.74, 3.71 ] |
| Machine Type 4 | 0.08 | 0.04 | [ -0.01, 0.16 ] |
| Cell 4 | | | |
| Machine Type 5 | 3.80 | 0.79 | [ 2.06, 5.54 ] |
| Machine Type 5 | 2.84 | 0.70 | [ 1.29, 4.38 ] |
| **Number of Tasks at Machines** | | | |
| Machine Type 1 | 159 | 3.43 | [ 151.44, 166.56 ] |
| Machine Type 2 | 191 | 4.98 | [ 180.04, 201.96 ] |
| Machine Type 3 | 142 | 3.39 | [ 134.54, 149.46 ] |
| Machine Type 4 | 153 | 3.73 | [ 144.80, 161.20 ] |
| Machine Type 5 | 174 | 4.52 | [ 164.06, 183.94 ] |
| Machine Type 4 (Without Cell 4) | 143 | 3.02 | [ 136.34, 149.66 ] |

Table 5.6: Measures of Performance - the Hybrid Approach

17. The greater the different between MTBA and MTBC (MTBA < MTBC), the larger the queues expected.

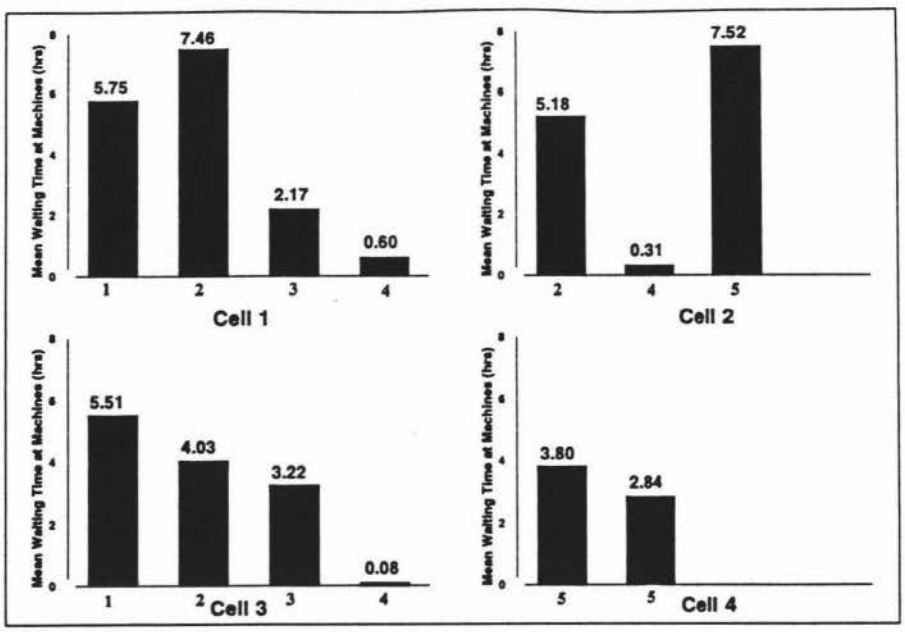


Figure 5.6: Mean Waiting Time of Jobs at Machines ($D_m$) in Each Cell

On the other hand, reducing the number of machines means reducing costs, since CNC machines are expensive. Additionally, Figure 5.7 indicates that the average number of tasks at machine type 4 in cell 3 was only 10. Overall, the average number of tasks required processing at machine type 4 was 153 (Figure 5.8) with a 95% confidence interval [144.80, 161.20] (ref: Table 5.6). Without considering machine type 4 in cell 3, Figure 5.8 shows that the average number of tasks at machine type 4 was 143 which is about 6.5% from the 153 figure. Since there is a fear that reducing the number of machines would cause higher waiting and flow times, the value indicated that such an effect would be insignificant. Based on this, reducing the number of machines is preferable to improve the overall shop utilisation, since this would reduce costs and not influence the overall performance of the shop.

### *Cell Layout Consideration*

Furthermore, Figure 5.4 shows that the lowest Mean Queue Length was at cell 4. This could indicate that arranging identical machines in a cell would not evenly distribute the number of jobs in the system. Looking at Figure 5.6, it is probably better if one of the

machine type 5 in cell 4 (the lowest $D_m$[18]) and the machine type 2 of cell 1 (the second highest $D_m$) are swapped. This would cause a more even distribution of jobs waiting in the queues at each cell. This is especially important, since each cell has limited queueing capacity.

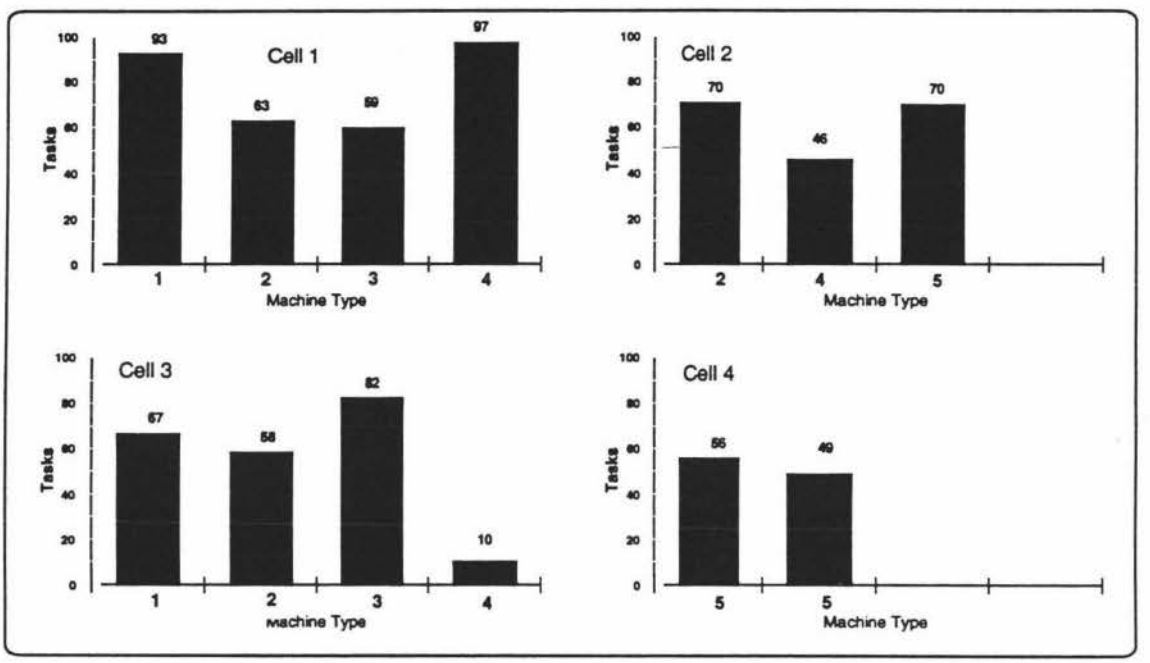


Figure 5.7: Average Number of Tasks Perfomed by Cell and Machine Type

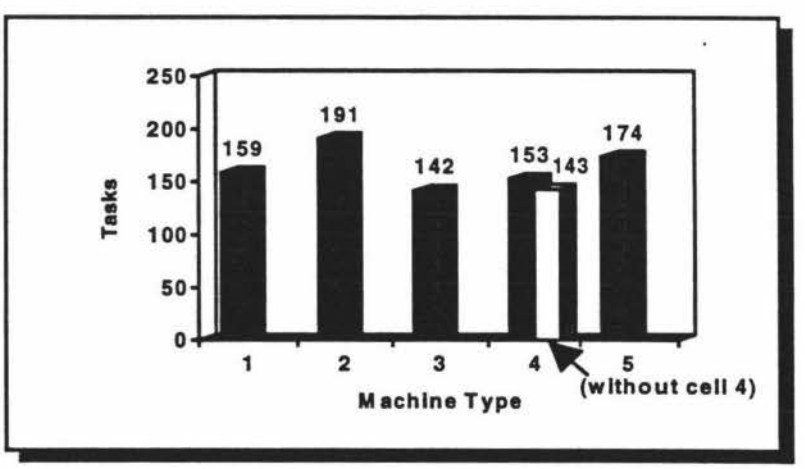


Figure 5.8: Average Number of Tasks at Each Machine Type

18. $D_m$ is the mean waiting times of tasks at machines

### *Flow Time Analysis*

Figures 5.9 shows that there was an interdependence between the average tardiness and the average flow times. The graphical patterns of the average tardiness followed the pattern of the average flow times. In other words, the fluctuation in the average flow times was the main cause of the unsteady pattern in the average tardiness.

Since the average flow times were related to the average waiting and processing times, Figure 5.9 clearly shows that the fluctuation in the average flow times was mainly affected by the variability in the average waiting times. The correlation coefficient between them is shown in Table 5.7. This coefficient is calculated as follows:

$$r = \frac{n\sum XY - \sum X \sum Y}{\sqrt{\left[n\sum X^2 - (\sum X)^2\right] - \left[n\sum Y^2 - (\sum Y)^2\right]}}, \tag{5.26}$$

where $r$ range from -1.0 to + 1.0.

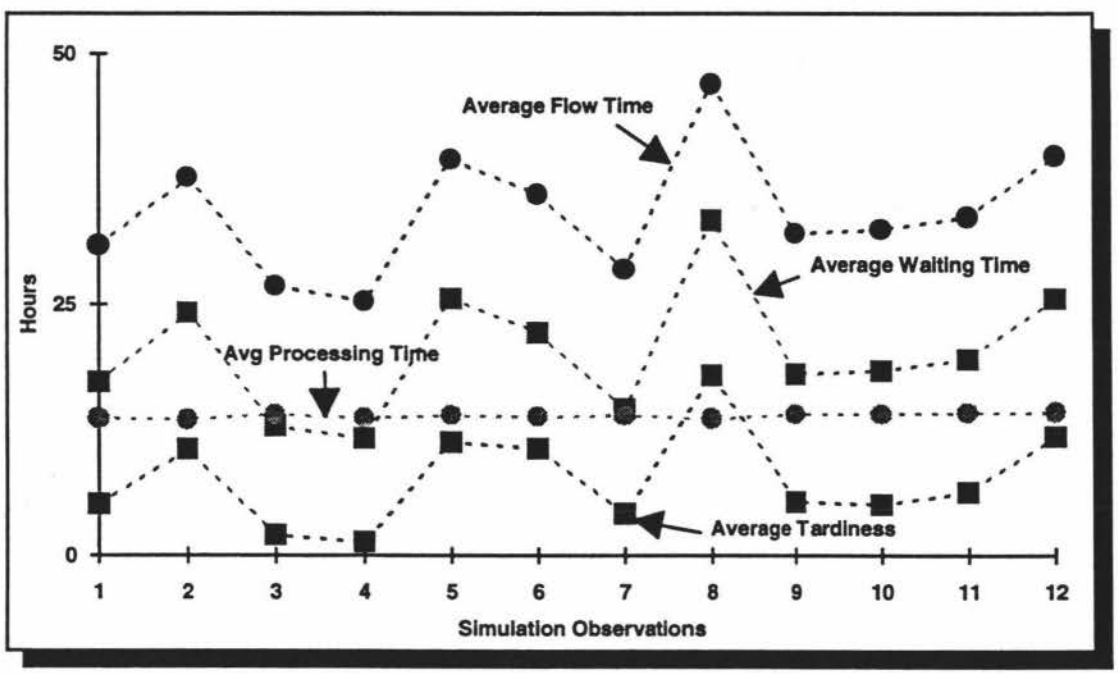


Figure 5.9: Average Flow Times ($\overline{F}$) & Average Tardiness ($\overline{T}$)

A value of *r* near or equal to 0 implies little or no linear relationship exists between the two lists of numbers (X and Y). A value of *r* near or equal to 1 or -1 indicates a very strong linear relationship. *r* is often referred to as *Pearson's* or the *Pearson product-moment correlation* (Bechtold et al., 1989).

Table 5.7 shows the correlation coefficient between the average waiting times and other parameters. Clearly indicated, there exists a very strong linear relationship between the average waiting times and other parameters, except the average processing times.

Waiting time indicates how long each operation of each job waits *before processing begins*. Total waiting time for a particular job is the sum of the waiting times for all operations of the job. For a problem consisting of *n* jobs, the average (total) waiting time indicates the length of time that all jobs spent *on average* in the shop, *excluding total processing times for all operations of the job* (see equation 5.10).

| Array of Independent Values | Array of Dependent Values | *r* |
|---|---|---|
| Average Waiting Times | Average Processing Times | - 0.1726 |
| | Average Flow Times | 0.9993 |
| | Average Lateness | 0.9980 |
| | Number of Late Jobs | 0.9563 |
| | % Late Jobs | 0.9559 |
| | Average Tardiness | 0.9886 |

Table 5.7: Correlation Coefficient between Parameters

There are two reasons why in the normal situation[19] an operation of a job has to wait, firstly, another task is being processed by a resource (in this case a CNC machine) and secondly, there are some tasks already waiting for processing in the queue while an operation is being performed. Obviously, the larger the queue facing an operation, the longer the time the operation has to spend waiting.

19. It is assumed that resources (machines) do not break down.

If it is assumed that setup and loading factors for each machine are 1.00, average time required by a machine to process a task is 0.63 hours and the batch size for each task is generated from a U[5,10] distribution. The lowest and the highest total times required to process the task are 3.15 (5 x 0.63) and 6.3 (10 x 0.63) hours, respectively. Thus the larger the batch size of a task, the longer the time needed to process the task.

Since the waiting time is mainly affected by the length of the queue, and the length of the queue is actually total processing times of tasks waiting in the queue, and the processing time required to complete a task is mainly determined by the batch size of the task (see equation 5.1), it can now be derived that in this research the variability in the average waiting times was mainly caused by the heterogenity of the batch sizes of the jobs arriving at the shop[20].

It seems intuitively obvious that to decrease total processing times required to complete a job, the maximum batch size allowed for each job has to be reduced. There are two alternatives that can be considered. First, the maximum batch sizes allowed are reduced to a certain level, and then varying the sizes of the jobs before they are released to the shop, ranging from the lowest amount permitted to that maximum quantity. Second, leave the range of the process batch size as it was (between 5 and 10 inclusive), but now the shop controller is allowed to split the job into two or more sub-jobs, with each sub-job of "equal" transfer batch size (for example, see Figure 5.10)[21].

Whatever alternatives are used to reduce the batch size, this would probably reduce the waiting times which in turn would reduce the flow times of jobs completed, decrease

20. Looking at the correlation coefficient between the average waiting times and the average processing times indicated in Table 5.7, it seems that, at first sight, this conclusion is contradictive. However, it should be noted that the average processing times considered in this research refers to total *processing* times (not including waiting times) for all operations of a job *on average* before leaving the shop.

21. Probably, this alternative is preferable, since the batch sizes of the jobs are generally determined before scheduling and can not be controlled by the shop controller. After the job arrives at the shop, however, the shop controller can split the job into sub-jobs. This is obviously future work requiring investigation.

the percentage of jobs late and eventually improve the average tardiness. Moreover if the shop is especially intended to operate in make-to-stock environment, relaxation in setting due dates might also be considered to reduce the number of jobs late.

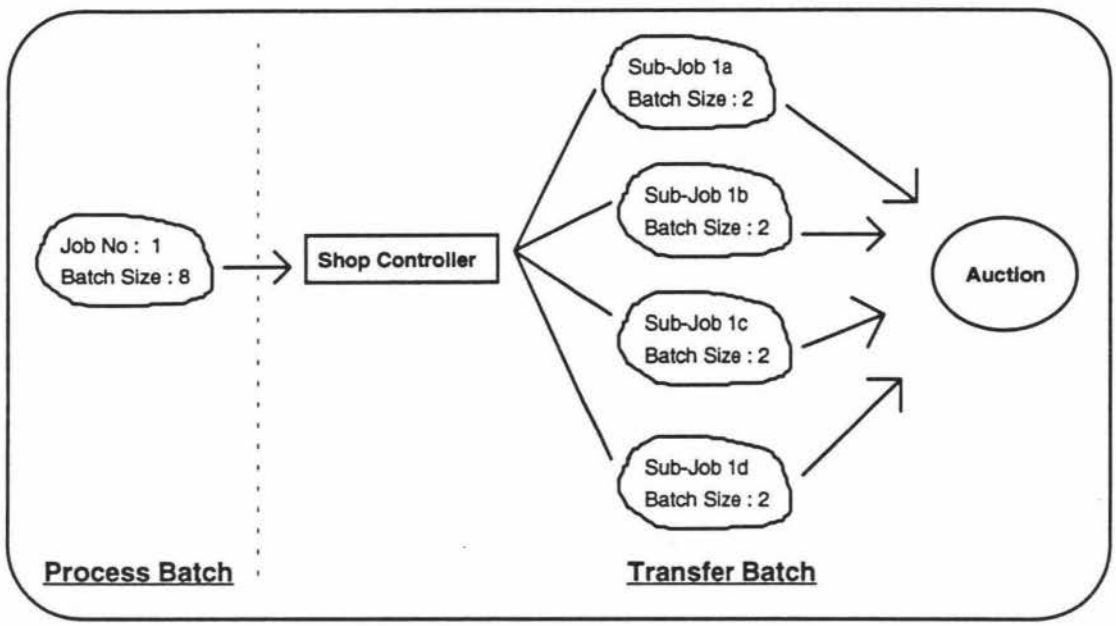


Figure 5.10: Process and Transfer Batches for the Hybrid Control Architecture

## 5.7 Summary

The test case analysis of the SIMCON package has been conducted: (1) to study the "hybrid" auction-based scheme, along with the "random" scheduling scheme for performance comparison, and; (2) to study and verify the operation of the implemented hybrid control architecture by considering the cell layout.

The SIMCON package was first started with empty and idle conditions. It was run until steady state conditions were reached, and then the statistical measures were reset. The time period for resetting the statistical measures in this research was after every 250 hours.

Based on the simulation results obtained, the auction-based scheme for scheduling jobs in the hybrid control architecture performs statistically better than the random approach to assigning jobs.

As far as scheduling is concerned, the simulation results also indicated that there are specific conditions required to implement successfully the hybrid control architecture to controlling the activities of an FMS shop. These conditions can be grouped into three categories, as follows:

(1) *Minimisation of the mean flow time of jobs completed.*

Based on the above analysis, in order to minimize the flow time of jobs completed, the batch sizes should be quite small.

(2) *Maximisation of the average cell utilisation over all batches.*

To increase the overall shop utilisation, the simulation results indicated that the number of the machines of the same type showing the lowest utilisation should be reduced, where possible.

(3) *Balancing work-loads at the cells.*

Since each cell has limited queueing capacity, two or more identical machines should not be found in each cell. This would balance the work-loads on the shop. Shimoyashiro, Isoda and Awane (1993) state that when the work load on the shop is balanced, the performance of the shop is improved under all scheduling rules.

In summary, the obvious result of the simulation study undertaken is to indicate the need for further extensive research. This is discussed in Chapter 6.

# CHAPTER 6

# FUTURE WORK

## 6.1 Introduction

Results obtained from the simulation study indicate that the hybrid control architecture along with its auction-based sequencing scheme has attractive attributes for the control of manufacturing systems. As mentioned in Chapter 5, there is the need for further research. The areas required to be investigated can be categorised into eight groups: (1) the auction-based sequencing scheme; (2) batching policy; (3) tool control system; (4) system disturbances; (5) cell layout consideration; (6) information system for the hybrid control architecture; (7) the role of human operators, and; (8) actual implementation.

## 6.2 The Auction-based Scheme

The "hybrid" auction-based scheme is part of a three-level scheduling framework (see Figure 6.1). The first level is task selection. The second-level is the bidding function to calculate bids and the third-level is the local scheduling problem within each cell. In this research, the selection of tasks to be auctioned is based on "First Come First Auctioned" (FIFA). The bidding function is based on the EFT and the rule used to load a task on a machine (the third level) is based on FCFS (First Come First Served).

The above feature along with a single supervisor provides an opportunity to employ the auction-based scheme, by incorporating different dispatching heuristics, to achieve a global goal. For FMSs operating on a make-to-order basis, for instance, due dates are of paramount importance. The global goal is evidently minimising the lateness of jobs.

EDD (earliest due date) heuristic may be employed by the shop controller to select tasks to be auctioned, along with the Bidding-EFT to calculate the bids and FCFS rule to load the tasks on machines. This example obviously indicates that further extensive research is required to investigate the benefits of the auction-based scheme.

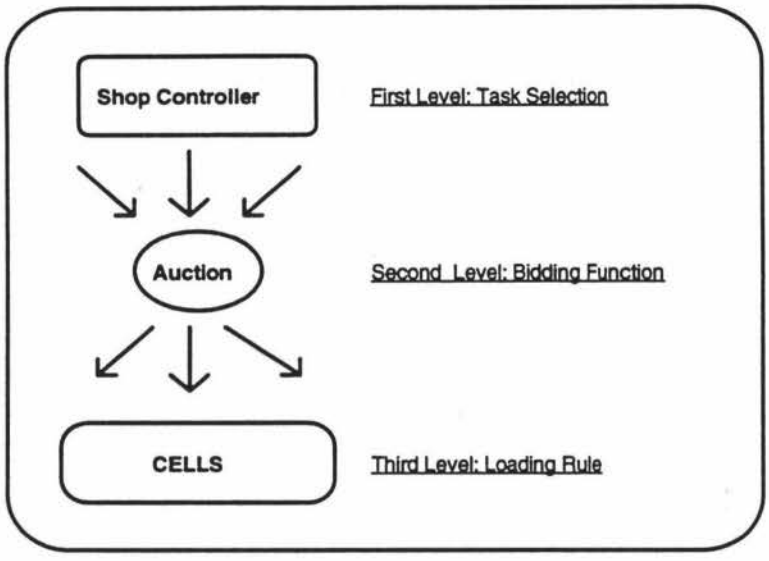


Figure 6.1: Three-Level Scheduling

Moreover, the optimality of the auction-based scheme developed needs to be studied to evaluate its long term performance in the hybrid control architecture. The fundamental objective of maintaining full local autonomy for subordinates, as Duffie and Piper (op.cit./c) state, contradicts the objective of *optimising* the overall system performance. Distributed optimisation algorithms need therefore to be developed.

## 6.3 Batching Policy

As suggested in Chapter 5, research regarding appropriate batching policy for the hybrid control architecture needs to be undertaken. A new function that facilitates the shop controller deciding whether the batch size of a task needs to be split or not, for example, should be developed. Since there exists a relationship between capacity level

and the batching policy, the batching policy should be studied along with the analysis of capacity levels in each cell (Karmarkar et al., 1987).

## 6.4 Tool Control System

There is currently no control over this matter. Each cell is assumed to be equipped with the required tool magazines to perform the processing operations. If the machine is available, then so are any tools required.

This assumption needs to be relaxed for further study. Each cell is equipped with a limited number of tools and the shop controller maintains a central tool crib. The procedure to request tools required from the central tool crib by cells could be as follows:

When the deadline for bid submission is due, the shop controller ranks the bids and awards the task to the best bidder. In case some of required tools are not locally available, the winning cell then requests the tools. This request is received and processed by the shop controller. A request consists of a set of tools, the earliest time when the requested tool-set is needed, and the expected return time of the tool-set. This gives the shop controller the possibility to perform actions to have the tools available at the time they are needed. Moreover the shop controller will "book" the tools specifically for the winning cell so that other cells would not be able to require the tools at the same time.

Obviously, the appropriate number of tools for different families of tasks should be provided at the central tool crib. New functions implementing this tool-requested procedure need also to be created. In addition, the shop controller requires a central tool database which downloads tool data on request of the cells.

## 6.5 System Disturbances

In this research, it is assumed that there are no breakdowns of machines, no tool breakages, and no alternative routing of jobs in case of machine and/or tool breakages. However, system disturbances do occur in actual production environments. The auction-based scheme needs therefore to be capable to respond to system disturbances. In order to study the hybrid control architecture under such a situation, these assumptions should be relaxed.

System disturbances can be categorised into two levels - major disturbances and minor disturbances (Kim et al., op.cit.). Major disturbances include arrivals of urgent jobs and major machine breakdowns, ones requiring long or unestimatable repair time whereas minor disturbances include material stockouts, worn tools, tool breakages, and machine breakdowns for which the estimated repair time is short (Kim et al., ibid.).

A new function has to be written that enables the shop controller to reroute tasks from a disabled machine.

## 6.6 Cell Layout Consideration

Essentially, the ultimate goal of the auction-based scheme is to evenly distribute the number of tasks in the queues. The simulation results obtained clearly show that in order to achieve this goal, cell layout (cell formation) has to be taken into account. Perhaps, a simulation model specifically to study the appropriate cell layout of the hybrid control architecture with respect to the auction-based scheme needs also to be constructed.

## 6.7 Information System Architecture

Veeramani, Bhargava and Barash (op.cit.) states that "the shop floor control system is intimately related to the information system" (p.90). This implies that an information system architecture suitable for supporting the hybrid control architecture is inevitably required to be developed. This includes:

(1) Identification of information requirements for workpieces, machine-tools, and transporters (AGVs);

(2) Cataloguing information - identification of information that should reside in the global database as well as in the local databases.

(3) Since the local databases only serve local users and access to data not available locally is made possible via the higher level, the framework that facilitates interaction (communication and integration) between the global and local databases should also be developed.

## 6.8 The Role of Human Operators

There is a question as to what the role of human operators should be in the hybrid control architecture. In an automated manufacturing, the human operators are required in substantial numbers to run the system. In addition, in the process of production, they also receive information from computer screens at the shop controller and cells to either performing supervisory, manual tasks in case of tools breakdowns or decision-making functions in abnormal situations. This indicates that their jobs are important and needs therefore to be clearly outlined, especially with regard to the hybrid control architecture.

## 6.9 Actual Implementation

A single supervisor also implies that the smoothness of the auction-based scheme is highly dependent on the shop controller. To know how high this dependence and also

to study the robustness the hybrid control architecture, it seems that an actual experiment employing real CNC machines, computers and LAN inevitably needs to be constructed.

## 6.10 Summary

The areas that should be addressed for future work have been outlined. Altogether there are 8 areas.

# CHAPTER 7

# CONCLUSION

## 7.1 Introduction

This thesis has detailed the development and the implementation of a hybrid control architecture as a way of controlling activities in an FMS shop floor. This chapter presents the conclusions drawn from the research.

## 7.2 The Concept of the Hybrid Control Architecture

1. This architecture incorporates the strengths of the four standard FMS control architectures: centralised, hierarchical, modified hierarchical and heterarchical. This includes levels of control, full autonomy for subordinates, full intelligent entities, simplicity in scheduling and global as well as local databases. The expected objective is to maximise the advantages associated with these existing architectures and minimise their problems.

2. The method used for scheduling jobs is the "hybrid" auction-based scheme with the top level (the shop controller) acting as the centralised auctioneer. The "hybrid" auction-based scheme is a part of a three-level scheduling framework: task selection, bidding function and local scheduling.

## 7.3 Method Used to Study The Hybrid Control Architecture

1. The method used is based on a simulation package serving two functions - the control facility and simulation tool. Technically, it involves two software packages - a real-time control software (FIX DMACS) and a windows programming language (Visual Basic). This approach is called SIMCON (Simulation Control).

2. By running the Visual Basic programs and the real-time control software simultaneously, the SIMCON package offers the following benefits:

   (a). The controls conducted and decisions made are the same as in the real time.

   (b). Rapid analysis can be done without actually constructing and operating the real system.

   (c). If it is required, the model can be reconstructed without risking disruption to production or to the control system.

   (d). Changes can be made when still in the development phase.

   (e). There exists a unique opportunity to develop the simulator that is initially used as a simulation tool and, later, as a supporting control software for the real-time control system.

3. In order to run simultaneously, an Interface containing a set of rules that both softwares can "understand" was developed. Creating an interface offers the following benefits:

   (1). the interface may easily be modified with minimal disruption to both the control and simulation programs.

   (2). Extensions to the functionality of the simulator, supporting functions, modelled FMS shop floor and auction-based control functions may easily be made.

   (3). Changing the real-time control software with another control software but also operating under the same environment (Windows) may also be possible.

## 7.4 The Results Obtained From The Simulation Study

1. As far as scheduling is concerned, there are three specific conditions required to implement the hybrid control architecture. Firstly, the batch size should be quite small to minimise the flow time of jobs completed, secondly, the machine of the same type showing the lowest utilisation should be reduced if possible, and thirdly, two or more identical machines should not be found in a cell.

2. The obvious result is to indicate the need for further extensive research. Eight areas have been identified for future work. They are: the auction-based scheme, batching policy, tool control system, system disturbances, cell layout consideration, information system architecture, the role of human operators and actual implementation.

## 7.5 The Future Potential of The Hybrid Control Architecture

1. The advantage the hybrid control architecture over the heterarchical architecture is that it has a single clear supervisor. This provides an opportunity to employ the "hybrid" auction-based scheme, by *incorporating different dispatching heuristics*, to achieve *a global goal.*

2. The use of the concept of levels of control, full autonomy for subordinates and no peer-to-peer interaction offers decomposition of the control problem, reduced control system complexity by localising information *without* eliminating global information, and elimination of connectivity problems. This leads to reduced software development problems and possible gradual implementation. Changes to existing elements of the system may easily be made. New elements may also easily be added to augment the existing level of functionality.

## 7.6 Publications From This Research

1. Ginting, D., Hurley, S., and Nahavandi, S. (1995). Simulation to aid manufacturing control. *Proceedings IPENZ Annual Conference 1995 "Innovative Technology" Volume 1,* Palmerston North, February 10-14, 1995, pp.26-31.

2. Ginting, D., Hurley, S., and Nahavandi, S. (1994). Development of a simulation module for near real-time scheduling of process activities in a flexible manufacturing system. *Proceedings of the Inaugural New Zealand Postgraduate Conference for Engineering and Technology Students,* Palmerston North, August 18-19, 1994, pp. 134-138.

## 7.7 Contribution of This Research

The research described in this thesis contributed the following:

1. Development and implementation of a new control architecture for controlling activities in an FMS shop floor.

2. The use of a unique approach incorporating two functions - control and simulation - to study the scheduling aspect of the new control architecture.

# REFERENCES


Arzi, Y., and Roll, Y. (1993). Dispatching procedures for a flexible manufacturing cell in constant production circumstances. *International Journal of Operations & Production Management, 13* (11), 35-51.

Aston, J.E., Johnson, M.D., and Cook, F.X. (1990). Shop floor control in a system job shop: Definitely not MRP. *Production and Inventory Management Journal, 31* (2), 27-31.

Baker, C.T., and Dzielinski, B.P. (1960). Simulation of a simplified job shop. *Management Science 6* (3), 311-323.

Baker, K.R. (1974). *Introduction to sequencing and scheduling*. New York: John Wiley & Sons.

Bakker, H. (1988). DFMS: A new control structure for FMS. *Computers in Industry, 10* (1), 1-9.

Bechtold, B., and Jonhson, R. (1989). *Statistics for business and economics*. Boston: PWS-Kent.

Bilberg, A., and Alting, L. (1991). When simulation takes control. *Journal of Manufacturing System, 10* (3), 179-193.

Black, J.T. (1983). Cellular manufacturing systems reduce setup time, make small lot production economical. *Industrial Engineering, 15* (11), 36-48.

Bona, B., Brandimarte, P., Greco, C., and Menga, G. (1990). Hybrid hierarchical scheduling and control systems in manufacturing. *IEEE Transactions on Robotics and Automation, 6* (6), 673-686.

Bourne, D.A., and Fox, M.S. (1984). Autonomous manufacturing: Automating the job-shop. *Computer, 17* (9), 76-86.

Browne, J., Dubois, D., Rathmill, K., Sethi, S.P., and Stecke, K.E. (1984). Classification of flexible manufacturing systems. *Industrial Engineering, 2* (2), 114-117.

Browne, J. (1988). Production activity control - A key aspect of production control. *International Journal of Production Research, 26* (3) 415-427.

Browne, J. and Jackson, S. (1989). An interactive scheduler for production activity control. *International Journal of Computer Integrated Manufacturing, 2* (1), 2-14.

Butler, M., and Nahavandi, S. (1994). Application of a SCADA package for the control of a flexible manufacturing system. *Proceedings of the Inaugural New Zealand Postgraduate Conference for Engineering and Technology Students* (pp.42-49). Palmerston North: Massey University.

Cassandras, C.G. (1986). Autonomous material handling in computer integrated manufacturing. In A.Kusiak (ed.), *Modelling and design of flexible manufacturing systems* (pp.67-98). Amsterdam: Elsevier.

Chen, I.J, Chung, C.S., and Gupta, A. (1994). The integration of JIT and FMS: Issues and decisions. *Integrated manufacturing systems, 5* (1), 4-13.

Chryssolouris, G. (1992). *Manufacturing systems: Theory and practice*. New York: Springer-Verlag.

Conway, R.W., Maxwell, W.L., and Miller, L.W. (1967). *Theory of scheduling*. New York: Addison-Wesley.

Cutkosky, M.R., Fussel, P.S., and Milligan, R.Jr. (1984). The design of a flexible machining cell for small batch production. *Journal of Manufacturing Systems, 3* (1), 39-59.

Dilts, D.M., Boyd, N.P., and Whorms, H.H. (1991). The evolution of control architectures for automated manufacturing systems. *Journal of manufacturing systems, 10* (1), 79-93.

Disney, R., Wysk, R., Peters, B., Smith, J., Kumaran, T.K., Lee, S., Hur, S. and Edlabadkar, A. (1994). Modelworld - an open system for manufacturing systems analysis. *Industrial Engineering, 26* (7), 49-54.

Duffie, N.A (1990). Synthesis of Heterarchical Manufacturing Systems. *Computers in Industry, 14* 167-174.

Duffie, N.A., Chitturi, R., and Mou, J.(1988a). Fault-tolerant heterarchical control of heterogeneous manufacturing system entities. *Journal of manufacturing systems, 7* (4), 315-328.

Duffie, N.A., and Piper, R.S. (1987b). Nonhierarchical control of manufacturing systems. *Journal of Manufacturing Systems, 5* (2), 137-139.

Duffie, N.A., and Piper, R.S. (1987c). Non-hierarchical control of a flexible manufacturing cell. *Robotics & Computer-Integrated Manufacturing, 3* (2), 175-179.

Duffie, N.A., and Prabhu, V.V. (1994d). Real-time distributed scheduling of heterarchical manufacturing systems. *Journal of manufacturing systems, 13* (2), 94-107.

El-Tamimi, A.M., Suliman, S.M.A, and Williams, D.F. (1989). A simulation study of part sequencing in a flexible assembly cell. *International Journal of Production Research, 27* (10), 1769-1793.

Emshoff, J.R., and Sisson, R.L. (1970). *Design and use of computer simulation models.* New York: The Macmillan company.

Erschler, J., Roubellat, F., and Thuriot, C. (1985). Steady state scheduling of a flexible manufacturing system with periodic releasing and flow time constraints. *Annals of Operations Research, 3* 333-353.

Ginting, D., Hurley, S., and Nahavandi, S. (1995). Simulation to aid manufacturing control. *Proceedings IPENZ Annual Conference 1995 "Innovative Technology" 1* (pp.26-31). Palmerston North: IPENZ

Ginting, D., Hurley, S., and Nahavandi, S. (1994). Development of a simulation module for near real-time scheduling of process activities in a flexible manufacturing system. *Proceedings of the Inaugural New Zealand Postgraduate Conference for Engineering and Technology Students* (pp.34-138). Palmerston North: Massey University.

Graybeal, W.J., and Pooch, U.W. (1980). *Simulation: Principles and Methods.* Cambridge: Winthrop.

Greenwood, N.R. (1988). *Implementing flexible manufacturing systems.* London: Macmillan.

Groover, M.P. (1980). *Automation, production systems, and computer-aided manufacturing.* New York: Prentice-Hall.

Gunasekaran, A., Goyal, S.K., Virtanen, I., and Yli-Olli, P. (1993). Production planning and control in automated manufacturing systems. *Production Planning & Control, 4* (1), 2-16.

Hastings, N.A.J., and Yeh, C. (1992). Bill of manufacture. *Production and Inventory Management Journal, 33* (4), 27-31.

Hitomi, K. (1994). Moving toward manufacturing excellence for future production perspectives. *Industrial Engineering, 26* (6), 48-50.

Hoover, S.V., and Perry, R.F. (1989). *Simulation: A problem-solving approach.* New York: Addison-Wesley.

Hurley, S.F. (1992). *Development of a generic simulation based critical resource scheduler for batch manufacturing environments.* (PhD Thesis). Liverpool: University of Liverpool.

Intellution. (1992-1994). *FIX DMACS for Windows Software v5.0.* (4 vols.). Norwood: Intellution

Jackson, R.H.F., and Jones, A.W.T. (1987). An architecture for decision making in the factory of the future. *Interfaces, 17* (6), 15-28.

Jackson, S., and Browne, J. (1989). An interactive scheduler for production activity control. *International Journal of Computer Integrated Manufacturing, 2* (1), 2-14.

Jones, C.H. (1973). An economic evaluation of job shop dispatching rules. *Management Science 20* (3), 293-307.

Jones, A., and Saleh, A. (1990). A multi-level/multi-layer architecture for intelligent shopfloor control. *International Journal Computer Integrated Manufacturing, 3* (1), 60-70.

Kamrani, A.K., and Parsaei, H.R. (1994). A methodology for the design of manufacturing systems using group technology. *Production Planning & Control, 5* (5), 450-464.

Kelton, W.D. (1986). Statistical analysis methods enhance usefulness, reliability of simulation models. *Industrial Engineering, 18* (9), 74-84.

Kim, M.H., and Kim, Y. (1994). Simulation-based real time scheduling in a flexible manufacturing system. *Journal of manufacturing system, 13* (2), 85-93.

Kleijnen, J.P.C. (1987). *Statistical tools for simulation practitioners.* New York: Marcel Dekker.

Kusiak, A., and Cheng, C. (1991). Group technology: Analysis of selected models and algorithms. In J.T. Black, B.C. Jiang and G.J. Wiens (eds.), *Design, analysis, and control of manufacturing cells* (pp.99-114). New York : The American Society of Mechanical Engineers.

Law, A.M., McComas, M.G., and Vincent, S.G. (1994). The crucial role of input modeling in successful simulation studies. *Industrial Engineering, 26* (7), 55-59.

Love, D., and Barekat, M.M. (1989). Decentralized, distributed MRP: Solving control problems in cellular manufacturing. *Production and Inventory Management Journal, 30* (3), 78-83.

Mellichamp, J.M., Kwon, O., and Wahab, A.F.A. (1990). FMS designer: An expert system for flexible manufacturing system design. *International Journal of Production Research, 28* (11), 2013-2024.

Mertins, K., and Wieneke-Toutaoui, B. (1991). State of the art in flexible manufacturing system. *Production Planning & Control, 2* (2), 155-159.

Microsoft Corporation. (1993). *Microsoft visual basic: Programming systems for windows v3.0.* (4 vols.). New York: Microsoft Corporation

Miltenburg, J.G., and Krisky, I. (1987, June). Evaluating flexible manufacturing systems. *IIE Transactions*, pp.79-92.

O'Grady, P.J. (1986). *Controlling automated manufacturing systems*. London: Chapman and Hall.

Pollacia, L.F. (1989). A survey of discrete event simulation and state-of-the-art discrete event languages. *Simulation Digest : IEEE Computer Society & ACM Press.*

Popovic, D., and Bhatkar, V.P. (1990). *Distributed computer control for industrial automation*. New York: Marcel Dekker.

Potter, B., and Scott, B. (1992). *Visual basic superbible*. (2nd ed.). Corte Madera: Waite Group Press.

Rachamadugu, R., and Stecke, K.E. (1994). Classification and review of FMS scheduling procedures. *Production Planning & Control, 5* (1), 2-20.

Ramasesh, R. (1990). Dynamic job shop scheduling: A survey of simulation research. *Omega 18* (1), 43-57.

Rao, P.P., and Mohanty, R.P. (1991). Searching for definitions and boundaries in flexible manufacturing systems. *Production Planning & Control, 2* (2), 142-154.

Robinson, S. (1993). The application of computer simulation in manufacturing. *Integrated manufacturing systems, 4* (4), 18-23.

Robinson, S. (1994). Simulation projects: building the right conceptual model. *Industrial Engineering, 26* (9), 34-36.

Schroer, B.J., Black, J.T., and Zhang, S.X. (1985). Just-in-time (JIT), with kanban, manufacturing system simulation on a microcomputer. *Simulation, 45* (2), 62-70.

Shaw, M.J. (1987). A distributed scheduling method for computer integrated manufacturing: the use of local area networks in cellular systems. *International Journal of Production Research, 25* (9), 1285-1303.

Shimoyashiro, S., Isoda, K., and Awane, H. (1984). Input scheduling and load balance control for a job shop. *International Journal of Production Research, 22* (4), 597-605.

Singhal, K., Fine, C.H., Meredith, J.R., and Suri, R. (1987). Research and models for automated manufacturing. *Interfaces, 17* (6), 5-14.

Stoeva, S. (1990). A due date-based dispatching rule for flexible manufacturing systems. *International Journal of Production Research, 28* (11), 1991-1999.

The Charles Stark Draper Laboratory. (1984). *Flexible manufacturing systems handbook*. Cambridge: Noyes Publications.

Tilley, K.J., and Williams, D.J. (1992, May). Modelling of communications and control in an auction-based manufacturing Control System. *Proceedings of the 1992 IEEE International Conference on Robotics and Automation* (pp 962-967). Nice: IEEE.

Veeramani, D., Bhargava, B., and Barash, M.M. (1993). Information system architecture for heterarchical control of large FMSs. *Computer Integrated Manufacturing Systems, 6* (2), 76-92.

Warnecke, H.J. (1983). New international developments for flexible automation. In Rathmill (ed.), *FMS* (pp.681-695). London:Chapman & Hall.

Wilczynski, D., and Wallace, D.K.(1993). OOPS in real time control applications. In S.Adiga (ed.), *Object oriented software for manufacturing systems* (pp.194.-226). London: Chapman & Hall.

Zhang, C., and Wang, H. (1992). Concurrent formation of part families and machine cells based on the fuzzy set theory. *Journal of Manufacturing Systems, 11* (1), 63-65.

# APPENDICES

# APPENDIX A

## THE SIMCON'S FIX DMACS DATABASE BLOCKS

The purpose of this appendix is to help future researchers who may wish to extend this work. A list of FIX DMACS database blocks used to implement the modelled FMS shop floor and the auction-based control functions of the shop, cells and machine controllers is oulined below[1].

## SHOP CONTROLLER - Auction Control

| No | Tag Name | Type | Description |
|---|---|---|---|
| 1 | LOADING | AO | loading time |
| 2 | MACHINE | AO | type of machine |
| 3 | PROCESSTM | AO | machining time |
| 4 | PRODUCT | AO | type of product |
| 5 | QUANTITY | AO | number of part |
| 6 | SETUPTIME | AO | setup time |
| 7 | JOBNO | AO | job number |
| 8 | TASK | AO | task number |
| 9 | TMARRIVAL | AO | arrival time to the shop controller |
| 10 | TRANSPORT | AO | transportation time |
| 11 | TOKEN | AO | Indicator for auction |
| 12 | AUCTION | AO | number of auction declared |
| 13 | BIDQUEUE | AO | signal for fc queueing program |
| 14 | RESPONSE | AO | response from cells |
| 15 | SORT1 | AO | the first eft |
| 16 | SORT2 | AO | the second eft |
| 17 | SORT3 | AO | the third eft |
| 18 | WINNER | AO | the winner |
| 19 | STORAGE | AO | storage of the shop controller |
| 20 | FCSTORAGE | AO | storage of the shop controller |
| 21 | INFCBUFFER | AO | in-buffer of the shop controller |
| 22 | FC | PG | shop controller |

1. Notes: AO = Analog Output Block
PG = Program Block
AI = Analog Input Block
CA = Calculation Block

## SHOP CONTROLLER - Bid-Evaluation Procedure

| No | Tag Name | Type | Description |
|---|---|---|---|
| 1 | WHOWINAC | PG | who is the winner (machine type 1 or 3) |
| 2 | WHOWINAC1 | PG | who is the winner (machine type 1 or 3) |
| 3 | WHOWINAC2 | PG | who is the winner (machine type 1 or 3) |
| 4 | WHOWINAC3 | PG | who is the winner (machine type 1 or 3) |
| 5 | WHOWINAC4 | PG | who is the winner (machine type 1 or 3) |
| 6 | WHOWINAC5 | PG | who is the winner (machine type 1 or 3) |
| 7 | WHOWINBDE | PG | who is the winner (machine type 2, 4, or 5) |
| 8 | WHOWINBDE1 | PG | who is the winner (machine type 2, 4, or 5) |
| 9 | WHOWINBDE2 | PG | who is the winner (machine type 2, 4, or 5) |
| 10 | WHOWINBDE3 | PG | who is the winner (machine type 2, 4, or 5) |
| 11 | WHOWINBDE4 | PG | who is the winner (machine type 2, 4, or 5) |
| 12 | WHOWINBDE5 | PG | who is the winner (machine type 2, 4, or 5) |
| 13 | WHOWINBDE6 | PG | who is the winner (machine type 2, 4, or 5) |
| 14 | WHOWINBDE7 | PG | who is the winner (machine type 2, 4, or 5) |
| 15 | WHOWINBDE8 | PG | who is the winner (machine type 2, 4, or 5) |
| 16 | WHOWINBDE9 | PG | who is the winner (machine type 2, 4, or 5) |
| 17 | WINNERBDE1 | PG | the winner (machine type 2, 4 or 5) |
| 18 | WINNERBDE2 | PG | the winner (machine type 2, 4 or 5) |
| 19 | WINNERBDE3 | PG | the winner (machine type 2, 4 or 5) |
| 20 | NOWINBDE | PG | no winner (machine type 2, 4 or 5) |

## SHOP CONTROLLER - Auction Initialisation

| No | Tag Name | Type | Description |
|---|---|---|---|
| 1 | INITIAL | PG | initialisation of setfac1,2 & 3 and C1,2 & 3 WAIT |
| 2 | INITIAL1 | PG | initial dummy |

# CELL CONTROLLERS - Auction Control

| No | Tag Name | Type | Description |
|---|---|---|---|
| 1 | CELL1 | PG | cell controller - 1 |
| 2 | CELL2 | PG | cell controller - 2 |
| 3 | CELL3 | PG | cell controller - 3 |
| 4 | CELL4 | PG | cell controller - 4 |
| 5 | QUELIMIT1 | AO | queue limit of cell 1 |
| 6 | QUELIMIT2 | AO | queue limit of cell 2 |
| 7 | QUELIMIT3 | AO | queue limit of cell 3 |
| 8 | QUELIMIT4 | AO | queue limit of cell 4 |
| 9 | QUEUE1 | AO | number of job in queue cell 1 |
| 10 | QUEUE2 | AO | number of job in queue cell 2 |
| 11 | QUEUE3 | AO | number of job in queue cell 3 |
| 12 | QUEUE4 | AO | number of job in queue cell 4 |
| 13 | C1WAIT | AI | waiting time of cell 1 |
| 14 | C2WAIT | AI | waiting time of cell 2 |
| 15 | C3WAIT | AI | waiting time of cell 3 |
| 16 | SETFAC1 | AI | setup factor cell 1 |
| 17 | SETFAC2 | AI | setup factor cell 2 |
| 18 | SETFAC3 | AI | setup factor cell 3 |
| 19 | C11EFT | AO | dummy of eft - cell 1 |
| 20 | C1EFT | AO | earliest finishing time cell 1 (also used by cell 4) |
| 21 | C1SPT | AO | shortest processing time cell 1 (also used by cell 4) |
| 22 | C22EFT | AO | dummy of eft - cell 2 |
| 23 | C2EFT | AO | earliest finishing time cell 2 |
| 24 | C2SPT | AO | shortest processing time cell 2 |
| 25 | C33EFT | AO | dummy of eft - cell 3 |
| 26 | C3EFT | AO | earliest finishing time cell 3 (also used by cell 4) |
| 27 | C3SPT | AO | shortest processing time cell 3 (also used by cell 4) |
| 28 | CAEFT1 | CA | calculation of eft - cell 1 (also used by cell 4) |
| 29 | CAEFT2 | CA | calculation of eft - cell 2 |
| 30 | CAEFT3 | CA | calculation of eft - cell 3 (also used by cell 4) |
| 31 | CASPT1 | CA | calculation of spt - cell 1 (also used by cell 4) |
| 32 | CASPT2 | CA | calculation of spt - cell 2 |
| 33 | CASPT3 | CA | calculation of spt - cell 3 (also used by cell 4) |

# MACHINE CONTROLLERS - Machine Parameters

| No | Tag Name | Type | Description |
|---|---|---|---|
| 1 | LOADFAC1a | AO | loading factor cell 1machine 1 |
| 2 | LOADFAC1b | AO | loading factor cell 1machine 2 |
| 3 | LOADFAC1c | AO | loading factor cell 1machine 3 |
| 4 | LOADFAC1d | AO | loading factor cell 1machine 4 |
| 5 | LOADFAC1 | AO | loading factor cell 1 |
| 6 | LOADFAC2b | AO | loading factor cell 2 machine 2 |
| 7 | LOADFAC2d | AO | loading factor cell 2 machine 4 |
| 8 | LOADFAC2e | AO | loading factor cell 2 machine 5 |
| 9 | LOADFAC2 | AO | loading factor cell 2 |
| 10 | LOADFAC3a | AO | loading factor cell 3 machine 1 |
| 11 | LOADFAC3b | AO | loading factor cell 3 machine 2 |
| 12 | LOADFAC3c | AO | loading factor cell 3 machine 3 |
| 13 | LOADFAC3d | AO | loading factor cell 3 machine 4 |
| 14 | LOADFAC3 | AO | loading factor cell 3 |
| 15 | LOADFAC41e | AO | loading factor cell 1 machine 5 (1) |
| 16 | LOADFAC42e | AO | loading factor cell 1 machine 5 (2) |
| 17 | SETFAC1A | AO | setup factor cell 1 machine 1 |
| 18 | SETFAC1B | AO | setup factor cell 1 machine 2 |
| 19 | SETFAC1C | AO | setup factor cell 1 machine 3 |
| 20 | SETFAC1D | AO | setup factor cell 1 machine 4 |
| 21 | SETFAC2B | AO | setup factor cell 2 machine 2 |
| 22 | SETFAC2D | AO | setup factor cell 2 machine 4 |
| 23 | SETFAC2E | AO | setup factor cell 2 machine 5 |
| 24 | SETFAC3A | AO | setup factor cell 3 machine 1 |
| 25 | SETFAC3B | AO | setup factor cell 3 machine 2 |
| 26 | SETFAC3C | AO | setup factor cell 3 machine 3 |
| 27 | SETFAC3D | AO | setup factor cell 3 machine 4 |
| 28 | SETFAC41 | AO | setup factor cell 4 machine 5 (1) |
| 29 | SETFAC42 | AO | setup factor cell 4 machine 5 (2) |

# MACHINE CONTROLLERS - Machine Control Algorithm

| No | Tag Name | Type | Description |
|---|---|---|---|
| 1 | MACHINE1a | PG | cell 1 machine 1 |
| 2 | MACHINE1b | PG | cell 1 machine 2 |
| 3 | MACHINE1c | PG | cell 1 machine 3 |
| 4 | MACHINE1d | PG | cell 1 machine 4 |
| 5 | MACHINE2b | PG | cell 2 machine 2 |
| 6 | MACHINE2d | PG | cell 2 machine 4 |
| 7 | MACHINE2e | PG | cell 2 machine 5 |
| 8 | MACHINE3a | PG | cell 3 machine 1 |
| 9 | MACHINE3b | PG | cell 3 machine 2 |
| 10 | MACHINE3c | PG | cell 3 machine 3 |
| 11 | MACHINE3d | PG | cell 3 machine 4 |
| 12 | MACHINE41e | PG | cell 4 machine 5 (1) |
| 13 | MACHINE42e | PG | cell 4 machine 5 (2) |
| 14 | C1AWAIT | AO | waiting time cell 1 machine 1 |
| 15 | C1BWAIT | AO | waiting time cell 1 machine 2 |
| 16 | C1CWAIT | AO | waiting time cell 1 machine 3 |
| 17 | C1DWAIT | AO | waiting time cell 1 machine 4 |
| 18 | C2BWAIT | AO | waiting time cell 2 machine 2 |
| 19 | C2DWAIT | AO | waiting time cell 2 machine 4 |
| 20 | C2EWAIT | AO | waiting time cell 2 machine 5 |
| 21 | C3AWAIT | AO | waiting time cell 3 machine 1 |
| 22 | C3BWAIT | AO | waiting time cell 3 machine 2 |
| 23 | C3CWAIT | AO | waiting time cell 3 machine 3 |
| 24 | C3DWAIT | AO | waiting time cell 3 machine 4 |
| 25 | C41EWAIT | AO | waiting time cell 4 machine 5 (1) |
| 26 | C42EWAIT | AO | waiting time cell 4 machine 5 (2) |

# APPENDIX B

## SETUP & LOADING FACTORS AND PROCESS ROUTING

B1: Setup And Loading Factors For Each Machine

B2: Process Routing - Product Information

## B1: SETUP AND LOADING FACTORS FOR EACH MACHINE

| Cell | Machine Type | Setup Factor | Loading Factor | Cell | Machine Type | Setup Factor | Loading Factor |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 0.85 | 0.85 | 3 | 1 | 1.13 | 1.24 |
| | 2 | 1.21 | 0.87 | | 2 | 0.77 | 1.00 |
| | 3 | 1.12 | 0.81 | | 3 | 0.96 | 0.75 |
| | 4 | 1.16 | 0.80 | | 4 | 1.13 | 1.25 |
| 2 | 2 | 1.08 | 0.89 | 4 | 5 | 1.13 | 0.98 |
| | 4 | 1.16 | 1.06 | | 5 | 0.89 | 1.12 |
| | 5 | 0.97 | 0.93 | | | | |

Note: Setup & Loading Factors were generated randomly between 0.75 and 1.25 inclusive

## B2: PROCESS ROUTING - PRODUCT INFORMATION

| Products | Operations | Machines | Setup Times | Loading Times | Machining Times | TPTo | TPTp |
|---|---|---|---|---|---|---|---|
| 10 | 5 | 1 | 0.05 | 0.03 | 0.08 | 0.16 | 2.07 |
| | 4 | 3 | 0.10 | 0.41 | 0.03 | 0.54 | |
| | 3 | 2 | 0.03 | 0.02 | 0.05 | 0.10 | |
| | 2 | 4 | 0.25 | 0.04 | 0.12 | 0.41 | |
| | 1 | 5 | 0.59 | 0.05 | 0.22 | 0.86 | |
| 9 | 5 | 1 | 0.11 | 0.06 | 0.12 | 0.29 | 3.43 |
| | 4 | 4 | 0.32 | 0.02 | 0.12 | 0.46 | |
| | 3 | 3 | 0.41 | 0.23 | 0.45 | 1.09 | |
| | 2 | 5 | 0.60 | 0.15 | 0.17 | 0.92 | |
| | 1 | 2 | 0.38 | 0.17 | 0.12 | 0.67 | |
| 8 | 5 | 4 | 0.05 | 0.03 | 0.03 | 0.11 | 3.43 |
| | 4 | 2 | 0.45 | 0.24 | 0.21 | 0.90 | |
| | 3 | 3 | 0.15 | 0.11 | 0.09 | 0.35 | |
| | 2 | 1 | 0.17 | 0.06 | 0.27 | 0.50 | |
| | 1 | 5 | 0.80 | 0.40 | 0.37 | 1.57 | |
| 7 | 6 | 3 | 0.02 | 0.18 | 0.05 | 0.25 | 3.69 |
| | 5 | 5 | 0.71 | 0.58 | 0.28 | 1.57 | |
| | 4 | 4 | 0.05 | 0.12 | 0.17 | 0.34 | |
| | 3 | 1 | 0.08 | 0.39 | 0.18 | 0.65 | |
| | 2 | 2 | 0.05 | 0.45 | 0.13 | 0.63 | |
| | 1 | 5 | 0.02 | 0.21 | 0.02 | 0.25 | |
| 6 | 3 | 5 | 0.71 | 0.21 | 0.28 | 1.20 | 1.92 |
| | 2 | 2 | 0.06 | 0.36 | 0.12 | 0.54 | |
| | 1 | 4 | 0.14 | 0.01 | 0.03 | 0.18 | |
| 5 | 5 | 2 | 0.25 | 0.14 | 0.12 | 0.51 | 3.33 |
| | 4 | 4 | 0.01 | 0.21 | 0.17 | 0.39 | |
| | 3 | 5 | 0.12 | 0.01 | 0.45 | 0.58 | |
| | 2 | 3 | 0.26 | 0.12 | 0.04 | 0.42 | |
| | 1 | 1 | 0.80 | 0.54 | 0.09 | 1.43 | |
| 4 | 4 | 1 | 0.18 | 0.10 | 0.07 | 0.35 | 3.08 |
| | 3 | 2 | 0.71 | 0.63 | 0.01 | 1.35 | |
| | 2 | 5 | 0.07 | 0.12 | 0.18 | 0.37 | |
| | 1 | 3 | 0.01 | 0.82 | 0.18 | 1.01 | |
| 3 | 5 | 4 | 0.36 | 0.11 | 0.06 | 0.53 | 2.45 |
| | 4 | 3 | 0.21 | 0.13 | 0.15 | 0.49 | |
| | 3 | 5 | 0.19 | 0.33 | 0.07 | 0.59 | |
| | 2 | 1 | 0.17 | 0.12 | 0.05 | 0.34 | |
| | 1 | 2 | 0.02 | 0.38 | 0.10 | 0.50 | |
| 2 | 6 | 2 | 0.03 | 0.30 | 0.24 | 0.57 | 3.35 |
| | 5 | 3 | 0.05 | 0.21 | 0.09 | 0.35 | |
| | 4 | 1 | 0.16 | 0.44 | 0.18 | 0.78 | |
| | 3 | 5 | 0.03 | 0.04 | 0.29 | 0.36 | |
| | 2 | 2 | 0.35 | 0.04 | 0.17 | 0.56 | |
| | 1 | 4 | 0.15 | 0.21 | 0.37 | 0.73 | |
| 1 | 3 | 1 | 0.51 | 0.03 | 0.27 | 0.81 | 2.89 |
| | 2 | 2 | 0.67 | 0.89 | 0.04 | 1.60 | |
| | 1 | 4 | 0.24 | 0.19 | 0.05 | 0.48 | |

Notes:
Mean loading time = 0.25 hours
Mean setup time = 0.20 hours
Mean machining time = 0.15 hours
TPTo = Processing Time per Operation (setup & loading factors = 1)
TPTp = Total Processing Time for a Product

# APPENDIX C

## TEST CASE RESULTS - THE RANDOM APPROACH

C1: Measures Of Performance

C2: Cells Utilisation

C3: Average Waiting Time Of Tasks At Machines

C4: Average Number of Tasks Arriving At Machines

**THE RANDOM APPROACH - C1**

**Number Of Shop Controllers : 1** **Capacity : 250**
**Number Of Cells : 4** **Capacity For Each Cell : 15**
**Number Of Machines In Each Cell : 4 3 4 2**

| | Start Condition | Simulation Observation | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | MEAN |
| **Cumulative Length (hrs)** | 2650 | 2900 | 3150 | 3400 | 3650 | 3900 | 4150 | 4400 | 4650 | 4900 | 5150 | 5400 | 5650 | |
| **Length (hrs)** | | 250 | 250 | 250 | 250 | 250 | 250 | 250 | 250 | 250 | 250 | 250 | 250 | 250 |
| **MTBA** | 1.49 | 1.49 | 1.49 | 1.49 | 1.49 | 1.49 | 1.49 | 1.49 | 1.49 | 1.49 | 1.49 | 1.49 | 1.49 | 1.49 |
| **MTBC** | 1.48 | 1.50 | 1.48 | 1.51 | 1.61 | 1.48 | 1.75 | 1.53 | 1.73 | 1.86 | 1.59 | 1.54 | 1.56 | 1.59 |
| **Cumulative Job Arrivals** | 1824 | 1995 | 2186 | 2343 | 2521 | 2674 | 2855 | 3036 | 3205 | 3387 | 3563 | 3743 | 3931 | |
| **Job Arrivals** | | 171 | 191 | 157 | 178 | 153 | 181 | 181 | 169 | 182 | 176 | 180 | 188 | 175.58 |
| **Cumulative Jobs Completed** | 1802 | 1976 | 2159 | 2330 | 2491 | 2667 | 2824 | 3009 | 3189 | 3353 | 3530 | 3720 | 3908 | |
| **Jobs Completed** | | 174 | 183 | 171 | 161 | 176 | 157 | 185 | 180 | 164 | 177 | 190 | 188 | 175.50 |
| **AQL** | | | | | | | | | | | | | | |
| **Cumulative AQL - Total** | 13.99 | 14.53 | 15.20 | 15.86 | 16.51 | 16.85 | 16.87 | 17.42 | 17.97 | 19.01 | 20.51 | 21.99 | 22.97 | 17.97 |
| **AQL per Observation (All Cells)** | | 20.25 | 22.95 | 24.20 | 25.46 | 21.81 | 17.21 | 26.59 | 27.57 | 38.35 | 50.11 | 52.44 | 44.24 | 30.93 |
| **Cell 1** | | 7.27 | 7.21 | 9.19 | 8.57 | 6.88 | 6.72 | 8.72 | 8.72 | 9.82 | 13.19 | 14.77 | 11.82 | 9.41 |
| **Cell 2** | | 3.43 | 2.82 | 3.64 | 2.61 | 2.06 | 2.72 | 4.93 | 3.88 | 8.69 | 13.18 | 13.37 | 11.70 | 6.09 |
| **Cell 3** | | 6.64 | 10.76 | 8.87 | 10.37 | 10.41 | 6.06 | 10.81 | 12.90 | 14.08 | 14.29 | 14.44 | 11.48 | 10.93 |
| **Cell 4** | | 2.85 | 2.17 | 2.54 | 3.75 | 2.47 | 2.00 | 2.17 | 2.24 | 5.19 | 8.64 | 8.90 | 8.81 | 4.31 |
| **Jobs in the Shop Controller** | 22 | 19 | 30 | 18 | 37 | 14 | 38 | 34 | 24 | 43 | 45 | 35 | 37 | 31.17 |
| **Jobs in Queues** | | | | | | | | | | | | | | |
| **Cell 1** | 10 | 9 | 7 | 7 | 11 | 2 | 14 | 7 | 11 | 7 | 8 | 14 | 13 | 9.17 |
| **Cell 2** | 4 | 0 | 11 | 1 | 5 | 0 | 7 | 3 | 3 | 13 | 12 | 14 | 15 | 7.00 |
| **Cell 3** | 7 | 2 | 12 | 6 | 9 | 3 | 15 | 14 | 13 | 14 | 13 | 15 | 15 | 10.92 |
| **Cell 4** | 1 | 5 | 2 | 2 | 3 | 1 | 2 | 8 | 4 | 13 | 14 | 4 | 0 | 4.83 |
| **Average Waiting Time** | | 28.81 | 30.83 | 38.24 | 35.79 | 34.84 | 22.92 | 37.03 | 41.57 | 52.71 | 71.10 | 69.35 | 62.68 | 43.82 |
| **Average Processing Time** | | 14.30 | 13.96 | 14.80 | 14.93 | 14.21 | 14.33 | 14.03 | 14.03 | 14.79 | 14.52 | 13.44 | 14.35 | 14.31 |
| **Average Flow Time (hrs)** | | 43.11 | 44.79 | 53.04 | 50.72 | 49.05 | 37.25 | 51.06 | 55.60 | 67.50 | 85.62 | 82.79 | 77.03 | 58.13 |
| **Average Lateness (hrs)** | | 13.59 | 15.03 | 22.12 | 19.73 | 19.37 | 6.67 | 22.35 | 25.17 | 37.47 | 60.16 | 62.41 | 48.89 | 29.41 |
| **Number of Late Jobs** | | 141 | 142 | 151 | 138 | 143 | 95 | 167 | 157 | 156 | 172 | 189 | 179 | 152.50 |
| **% Job Late** | | 0.81 | 0.78 | 0.88 | 0.86 | 0.81 | 0.61 | 0.90 | 0.87 | 0.95 | 0.97 | 0.99 | 0.95 | 0.87 |
| **Average Tardiness (hrs)** | | 14.80 | 16.76 | 22.67 | 20.53 | 20.68 | 10.16 | 22.93 | 26.08 | 37.68 | 60.37 | 62.42 | 49.21 | 30.36 |

**THE RANDOM APPROACH - C2**

**CELLS UTILISATION (%)**

| | | Simulation Observation | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | MEAN |
| **Cells** | **Machines** | *Um* | *Um* | *Um* | *Um* | *Um* | *Um* | *Um* | *Um* | *Um* | *Um* | *Um* | *Um* | |
| 1 | 1 | 0.93 | 0.96 | 0.98 | 0.96 | 0.94 | 0.94 | 0.92 | 0.98 | 0.99 | 1.00 | 1.00 | 0.95 | 0.96 |
| | 2 | 0.64 | 0.65 | 0.86 | 0.72 | 0.68 | 0.70 | 0.95 | 0.75 | 0.74 | 0.79 | 0.80 | 0.75 | 0.75 |
| | 3 | 0.77 | 0.88 | 0.87 | 0.92 | 0.77 | 0.72 | 0.67 | 0.83 | 0.90 | 0.99 | 0.96 | 0.94 | 0.85 |
| | 4 | 0.44 | 0.67 | 0.51 | 0.59 | 0.69 | 0.50 | 0.62 | 0.47 | 0.55 | 0.64 | 0.64 | 0.67 | 0.58 |
| | *Uc* | 0.70 | 0.79 | 0.80 | 0.80 | 0.77 | 0.72 | 0.79 | 0.76 | 0.80 | 0.60 | 0.85 | 0.83 | 0.77 |
| 2 | 2 | 0.75 | 0.70 | 0.74 | 0.74 | 0.70 | 0.68 | 0.83 | 0.87 | 0.92 | 1.00 | 1.00 | 1.00 | 0.83 |
| | 4 | 0.85 | 0.59 | 0.74 | 0.70 | 0.66 | 0.52 | 0.82 | 0.79 | 0.85 | 0.84 | 0.81 | 0.81 | 0.75 |
| | 5 | 0.69 | 0.75 | 0.62 | 0.75 | 0.59 | 0.69 | 0.59 | 0.63 | 0.63 | 0.44 | 0.41 | 0.66 | 0.62 |
| | *Uc* | 0.76 | 0.68 | 0.70 | 0.73 | 0.65 | 0.63 | 0.75 | 0.76 | 0.80 | 0.76 | 0.74 | 0.82 | 0.73 |
| 3 | 1 | 0.94 | 1.00 | 1.00 | 1.00 | 1.00 | 0.82 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.98 |
| | 2 | 0.81 | 0.91 | 0.79 | 0.77 | 0.61 | 0.87 | 0.56 | 0.67 | 0.51 | 0.43 | 0.66 | 0.81 | 0.70 |
| | 3 | 0.80 | 0.62 | 0.71 | 0.63 | 0.70 | 0.72 | 0.75 | 0.70 | 0.59 | 0.50 | 0.51 | 0.64 | 0.66 |
| | 4 | 0.76 | 0.78 | 0.50 | 0.71 | 0.65 | 0.84 | 0.88 | 0.61 | 0.66 | 0.51 | 0.54 | 0.72 | 0.68 |
| | *Uc* | 0.83 | 0.83 | 0.75 | 0.78 | 0.74 | 0.81 | 0.80 | 0.74 | 0.69 | 0.61 | 0.68 | 0.79 | 0.75 |
| 4 | 5 | 0.81 | 0.60 | 0.72 | 0.69 | 0.81 | 0.74 | 0.62 | 0.72 | 0.98 | 0.76 | 0.85 | 0.63 | 0.74 |
| | 5 | 0.83 | 0.83 | 0.62 | 0.84 | 0.67 | 0.63 | 0.78 | 0.74 | 0.77 | 1.00 | 1.00 | 0.93 | 0.80 |
| | *Uc* | 0.82 | 0.72 | 0.67 | 0.77 | 0.74 | 0.69 | 0.70 | 0.73 | 0.87 | 0.88 | 0.92 | 0.78 | 0.77 |
| | *Ufms* | 0.78 | 0.75 | 0.73 | 0.77 | 0.73 | 0.71 | 0.76 | 0.75 | 0.79 | 0.71 | 0.80 | 0.81 | 0.76 |

Notes : *Um = Machine Utilisation*
*Uc = Average Utilisation Of The Machines In A Cell*
*Ufms = Average Utilisation Of The Cells In The Modelled FMS*

**Average Utilisation of Each Type of Machines**

| | Run | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | MEAN |
| **Machine Type 1** | 0.94 | 0.98 | 0.99 | 0.98 | 0.97 | 0.88 | 0.96 | 0.99 | 0.99 | 1.00 | 1.00 | 0.98 | 0.97 |
| **Machine Type 2** | 0.73 | 0.75 | 0.80 | 0.74 | 0.66 | 0.75 | 0.78 | 0.76 | 0.72 | 0.74 | 0.82 | 0.85 | 0.76 |
| **Machine Type 3** | 0.78 | 0.75 | 0.79 | 0.78 | 0.73 | 0.72 | 0.71 | 0.76 | 0.75 | 0.74 | 0.74 | 0.79 | 0.75 |
| **Machine Type 4** | 0.68 | 0.68 | 0.59 | 0.67 | 0.67 | 0.62 | 0.77 | 0.62 | 0.68 | 0.66 | 0.66 | 0.73 | 0.67 |
| **Machine Type 5** | 0.77 | 0.73 | 0.65 | 0.76 | 0.69 | 0.69 | 0.66 | 0.70 | 0.79 | 0.73 | 0.75 | 0.74 | 0.72 |

**THE RANDOM APPROACH - C3**

**AVERAGE WAITING TIME OF TASKS AT MACHINES IN EACH CELL (hrs)**

| | | Simulation Observation | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | MEAN |
| Cells | Machines | *Dm* | *Dm* | *Dm* | *Dm* | *Dm* | *Dm* | *Dm* | *Dm* | *Dm* | *Dm* | *Dm* | *Dm* | |
| 1 | 1 | 11.90 | 8.49 | 13.43 | 16.76 | 9.52 | 10.05 | 11.04 | 15.55 | 11.44 | 18.41 | 24.82 | 13.01 | 13.70 |
| | 2 | 3.33 | 4.47 | 8.28 | 2.47 | 2.63 | 3.60 | 11.84 | 9.91 | 6.82 | 4.75 | 3.72 | 7.12 | 5.74 |
| | 3 | 5.88 | 7.18 | 4.42 | 4.09 | 6.69 | 8.28 | 3.19 | 1.90 | 6.88 | 10.69 | 8.98 | 10.61 | 6.57 |
| | 4 | 2.02 | 2.27 | 1.66 | 1.91 | 3.55 | 1.14 | 1.55 | 0.80 | 2.55 | 2.19 | 2.87 | 3.11 | 2.13 |
| 2 | 2 | 3.29 | 4.05 | 5.24 | 3.15 | 2.90 | 6.17 | 6.99 | 4.48 | 15.77 | 31.54 | 32.41 | 25.44 | 11.78 |
| | 4 | 7.01 | 1.11 | 5.19 | 3.37 | 3.10 | 0.65 | 8.11 | 6.19 | 6.51 | 5.06 | 6.55 | 5.55 | 4.87 |
| | 5 | 1.50 | 6.90 | 2.90 | 3.58 | 2.27 | 3.87 | 2.80 | 4.23 | 5.47 | 1.27 | 0.98 | 4.53 | 3.36 |
| 3 | 1 | 8.18 | 23.96 | 22.13 | 28.87 | 23.69 | 5.63 | 20.80 | 32.41 | 43.80 | 42.54 | 41.51 | 27.28 | 26.73 |
| | 2 | 7.07 | 6.76 | 2.43 | 3.63 | 2.50 | 5.24 | 3.23 | 1.73 | 1.35 | 1.88 | 2.70 | 4.44 | 3.58 |
| | 3 | 3.31 | 1.79 | 3.86 | 3.00 | 2.51 | 3.61 | 3.55 | 4.45 | 1.64 | 1.65 | 1.10 | 1.55 | 2.67 |
| | 4 | 4.14 | 3.49 | 2.32 | 2.90 | 2.38 | 10.64 | 6.87 | 5.03 | 5.20 | 2.58 | 1.40 | 2.00 | 4.08 |
| 4 | 5 | 6.55 | 3.43 | 6.23 | 3.93 | 3.93 | 3.05 | 2.98 | 3.75 | 12.67 | 7.99 | 10.70 | 2.36 | 5.63 |
| | 5 | 4.24 | 4.63 | 4.07 | 11.08 | 6.26 | 4.67 | 5.24 | 4.40 | 7.14 | 19.85 | 15.30 | 28.81 | 9.64 |

Note : *Dm = Average Time Spent By A Task Waiting In The Queue (hrs)*

**THE RANDOM APPROACH - C4**

**AVERAGE NUMBER OF TASKS ARRIVING AT MACHINES IN EACH CELL**

| | | Simulation Observation | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | MEAN |
| **Cells** | **Machines** | ***Am*** | ***Am*** | ***Am*** | ***Am*** | ***Am*** | ***Am*** | ***Am*** | ***Am*** | ***Am*** | ***Am*** | ***Am*** | ***Am*** | ***Am*** |
| 1 | 1 | 90 | 91 | 90 | 92 | 80 | 89 | 79 | 90 | 104 | 100 | 101 | 102 | 92 |
| | 2 | 53 | 67 | 69 | 60 | 62 | 57 | 81 | 63 | 63 | 84 | 68 | 62 | 66 |
| | 3 | 67 | 81 | 87 | 83 | 79 | 75 | 70 | 77 | 90 | 91 | 94 | 91 | 82 |
| | 4 | 41 | 62 | 53 | 61 | 55 | 52 | 54 | 51 | 59 | 64 | 57 | 58 | 56 |
| 2 | 2 | 68 | 61 | 61 | 69 | 58 | 69 | 79 | 76 | 102 | 93 | 94 | 89 | 77 |
| | 4 | 73 | 51 | 55 | 55 | 57 | 43 | 58 | 61 | 62 | 67 | 61 | 69 | 59 |
| | 5 | 59 | 67 | 58 | 65 | 50 | 61 | 51 | 57 | 55 | 40 | 36 | 57 | 55 |
| 3 | 1 | 77 | 80 | 74 | 74 | 87 | 77 | 89 | 90 | 73 | 74 | 87 | 85 | 81 |
| | 2 | 70 | 80 | 64 | 64 | 62 | 66 | 46 | 52 | 36 | 37 | 53 | 65 | 58 |
| | 3 | 76 | 62 | 59 | 56 | 68 | 66 | 75 | 62 | 51 | 46 | 60 | 61 | 62 |
| | 4 | 45 | 57 | 44 | 44 | 49 | 61 | 57 | 41 | 45 | 36 | 41 | 55 | 48 |
| 4 | 5 | 68 | 64 | 58 | 62 | 63 | 66 | 58 | 61 | 79 | 65 | 72 | 58 | 65 |
| | 5 | 59 | 60 | 58 | 60 | 52 | 49 | 68 | 53 | 60 | 88 | 83 | 70 | 63 |

Note : *Am = Number Of Tasks Arriving At Each Machine*

**Number Of Tasks Arriving At Each Machine Type**

| | Simulation Observation | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | MEAN |
| **Machine Type 1** | 167 | 171 | 164 | 166 | 167 | 166 | 168 | 180 | 177 | 174 | 188 | 187 | 173 |
| **Machine Type 2** | 191 | 208 | 194 | 193 | 182 | 192 | 206 | 191 | 201 | 214 | 215 | 216 | 200 |
| **Machine Type 3** | 143 | 143 | 146 | 139 | 147 | 141 | 145 | 139 | 141 | 137 | 154 | 152 | 144 |
| **Machine Type 4** | 159 | 170 | 152 | 160 | 161 | 156 | 169 | 153 | 166 | 167 | 159 | 182 | 163 |
| **Machine Type 5** | 186 | 191 | 174 | 187 | 165 | 176 | 177 | 171 | 194 | 193 | 191 | 185 | 183 |
| **Machine Type 4 (Without cell 4)** | 114 | 113 | 108 | 116 | 112 | 95 | 112 | 112 | 121 | 131 | 118 | 127 | 115 |

# APPENDIX D

## TEST CASE RESULTS - THE HYBRID APPROACH

D1: Measures Of Performance

D2: Cells Utilisation

D3: Average Waiting Time Of Tasks At Machines

D4: Average Number of Tasks Arriving At Machines

## THE "HYBRID" AUCTION-BASED SCHEME - D1

**Number Of Shop Controllers : 1** **Capacity : 250**
**Number Of Cells : 4** **Capacity For Each Cell : 15**
**Number Of Machines In Each Cell : 4 3 4 2**

| | Start Condition | Simulation Observation | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | MEAN |
| **Cumulative Length (hrs)** | 2650 | 2900 | 3150 | 3400 | 3650 | 3900 | 4150 | 4400 | 4650 | 4900 | 5150 | 5400 | 5650 | |
| **Length (hrs)** | | 250 | 250 | 250 | 250 | 250 | 250 | 250 | 250 | 250 | 250 | 250 | 250 | 250 |
| **MTBA** | 1.49 | 1.49 | 1.49 | 1.49 | 1.49 | 1.49 | 1.49 | 1.49 | 1.49 | 1.49 | 1.49 | 1.49 | 1.49 | 1.49 |
| **MTBC** | 1.48 | 1.65 | 1.39 | 1.57 | 1.68 | 1.47 | 1.50 | 1.69 | 1.39 | 1.58 | 1.47 | 1.69 | 1.53 | 1.55 |
| **Cumulative Job Arrivals** | 1824 | 1991 | 2167 | 2322 | 2487 | 2681 | 2825 | 3005 | 3186 | 3355 | 3528 | 3698 | 3843 | |
| **Job Arrivals** | | 167 | 176 | 155 | 165 | 194 | 144 | 180 | 181 | 169 | 173 | 170 | 145 | 168.25 |
| **Cumulative Jobs Completed** | 1802 | 1962 | 2150 | 2310 | 2463 | 2650 | 2817 | 2975 | 3168 | 3333 | 3506 | 3670 | 3835 | |
| **Jobs Completed** | | 160 | 188 | 160 | 153 | 187 | 167 | 158 | 193 | 165 | 173 | 164 | 165 | 169.42 |
| **AQL** | | | | | | | | | | | | | | |
| **Cumulative AQL - Total** | 13.99 | 13.76 | 14.05 | 13.61 | 13.21 | 13.75 | 13.63 | 13.46 | 14.09 | 13.98 | 13.92 | 13.89 | 13.95 | 13.78 |
| **AQL per Observation (All Cells)** | | 11.30 | 17.45 | 8.06 | 7.80 | 21.57 | 11.76 | 10.71 | 25.21 | 11.94 | 12.79 | 13.33 | 15.40 | 13.94 |
| **Cell 1** | | 4.87 | 6.35 | 2.51 | 2.80 | 5.50 | 4.35 | 3.76 | 9.20 | 3.97 | 4.40 | 5.81 | 6.13 | 4.97 |
| **Cell 2** | | 2.75 | 5.05 | 2.59 | 2.44 | 5.85 | 2.91 | 3.31 | 6.64 | 4.14 | 3.31 | 2.75 | 3.57 | 3.78 |
| **Cell 3** | | 2.68 | 4.94 | 1.88 | 1.81 | 4.47 | 3.55 | 2.63 | 8.40 | 2.59 | 2.97 | 4.00 | 5.70 | 3.80 |
| **Cell 4** | | 1.30 | 1.28 | 1.18 | 1.34 | 5.70 | 1.74 | 1.66 | 1.28 | 1.70 | 2.24 | 1.01 | 1.06 | 1.79 |
| **Jobs in the Shop Controller** | 22 | 29 | 17 | 12 | 24 | 31 | 8 | 30 | 18 | 22 | 22 | 28 | 8 | 20.75 |
| **Jobs in Queues** | | | | | | | | | | | | | | |
| **Cell 1** | 10 | 8 | 5 | 2 | 5 | 9 | 1 | 9 | 7 | 4 | 6 | 11 | 3 | 5.83 |
| **Cell 2** | 4 | 6 | 4 | 2 | 2 | 6 | 1 | 6 | 5 | 4 | 4 | 5 | 0 | 3.75 |
| **Cell 3** | 7 | 3 | 3 | 1 | 5 | 5 | 0 | 6 | 5 | 6 | 2 | 10 | 0 | 3.83 |
| **Cell 4** | 1 | 2 | 1 | 0 | 3 | 9 | 0 | 2 | 1 | 1 | 3 | 0 | 0 | 1.83 |
| **Average Waiting Time** | | 17.22 | 24.15 | 12.84 | 11.63 | 25.59 | 22.19 | 14.57 | 33.35 | 18.04 | 18.37 | 19.53 | 25.56 | 20.25 |
| **Average Processing Time** | | 13.62 | 13.53 | 14.02 | 13.66 | 13.91 | 13.79 | 13.95 | 13.6 | 14.05 | 14.07 | 14.17 | 14.24 | 13.88 |
| **Average Flow Time (hrs)** | | 30.84 | 37.68 | 26.86 | 25.29 | 39.5 | 35.98 | 28.52 | 46.95 | 32.09 | 32.44 | 33.7 | 39.8 | 34.14 |
| **Average Lateness (hrs)** | | 2.11 | 8.74 | -3.44 | -4.27 | 9.8 | 6.47 | -0.89 | 17.62 | 2.08 | 2.33 | 3.4 | 9.01 | 4.41 |
| **Number of Late Jobs** | | 85 | 137 | 44 | 43 | 137 | 92 | 66 | 176 | 89 | 104 | 97 | 111 | 98.42 |
| **% Job Late** | | 0.53 | 0.73 | 0.28 | 0.28 | 0.73 | 0.55 | 0.42 | 0.91 | 0.54 | 0.60 | 0.59 | 0.67 | 0.57 |
| **Average Tardiness (hrs)** | | 5.13 | 10.59 | 2.06 | 1.36 | 11.23 | 10.65 | 4.14 | 17.92 | 5.34 | 5.05 | 6.3 | 11.84 | 7.63 |

## THE "HYBRID" AUCTION-BASED SCHEME - D2

### CELLS UTILISATION (%)

| | | Simulation Observation | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | MEAN |
| Cells | Machines | *Um* | *Um* | *Um* | *Um* | *Um* | *Um* | *Um* | *Um* | *Um* | *Um* | *Um* | *Um* | |
| 1 | 1 | 0.96 | 0.96 | 0.91 | 0.91 | 0.99 | 0.90 | 0.94 | 0.98 | 0.96 | 1.00 | 0.95 | 0.94 | 0.95 |
| | 2 | 0.93 | 1.00 | 0.93 | 0.96 | 0.97 | 0.89 | 0.93 | 1.00 | 0.95 | 0.98 | 1.00 | 0.93 | 0.96 |
| | 3 | 0.48 | 0.64 | 0.57 | 0.57 | 0.71 | 0.65 | 0.56 | 0.79 | 0.65 | 0.66 | 0.61 | 0.51 | 0.62 |
| | 4 | 0.70 | 0.75 | 0.64 | 0.66 | 0.74 | 0.69 | 0.73 | 0.73 | 0.61 | 0.70 | 0.71 | 0.70 | 0.70 |
| | *Uc* | 0.77 | 0.84 | 0.76 | 0.77 | 0.85 | 0.78 | 0.79 | 0.88 | 0.79 | 0.84 | 0.82 | 0.77 | 0.81 |
| 2 | 2 | 0.86 | 0.98 | 0.81 | 0.77 | 0.98 | 0.91 | 0.84 | 1.00 | 0.96 | 0.90 | 0.88 | 0.88 | 0.90 |
| | 4 | 0.30 | 0.43 | 0.30 | 0.28 | 0.47 | 0.27 | 0.37 | 0.45 | 0.35 | 0.37 | 0.38 | 0.35 | 0.36 |
| | 5 | 0.99 | 0.99 | 1.00 | 1.00 | 1.00 | 0.94 | 0.99 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.99 |
| | *Uc* | 0.72 | 0.80 | 0.70 | 0.68 | 0.82 | 0.71 | 0.73 | 0.82 | 0.77 | 0.76 | 0.75 | 0.74 | 0.75 |
| 3 | 1 | 0.79 | 0.87 | 0.77 | 0.66 | 0.85 | 0.75 | 0.77 | 0.95 | 0.68 | 0.78 | 0.76 | 0.83 | 0.79 |
| | 2 | 0.57 | 0.87 | 0.48 | 0.66 | 0.89 | 0.62 | 0.65 | 1.00 | 0.63 | 0.68 | 0.73 | 0.74 | 0.71 |
| | 3 | 0.80 | 0.90 | 0.88 | 0.89 | 0.95 | 0.85 | 0.84 | 0.90 | 0.93 | 0.93 | 0.88 | 0.80 | 0.88 |
| | 4 | 0.32 | 0.10 | 0.31 | 0.12 | 0.16 | 0.05 | 0.10 | 0.24 | 0.26 | 0.14 | 0.12 | 0.22 | 0.18 |
| | *Uc* | 0.62 | 0.69 | 0.61 | 0.58 | 0.71 | 0.57 | 0.59 | 0.77 | 0.62 | 0.63 | 0.62 | 0.65 | 0.64 |
| 4 | 5 | 0.77 | 0.87 | 0.87 | 0.75 | 0.98 | 0.67 | 0.77 | 0.93 | 0.85 | 0.92 | 0.79 | 0.72 | 0.83 |
| | 5 | 0.64 | 0.73 | 0.62 | 0.57 | 0.90 | 0.63 | 0.68 | 0.66 | 0.66 | 0.80 | 0.58 | 0.61 | 0.67 |
| | *Uc* | 0.70 | 0.80 | 0.75 | 0.66 | 0.94 | 0.65 | 0.73 | 0.79 | 0.75 | 0.86 | 0.68 | 0.66 | 0.75 |
| | *Ufms* | 0.70 | 0.78 | 0.71 | 0.67 | 0.83 | 0.68 | 0.71 | 0.81 | 0.73 | 0.77 | 0.72 | 0.71 | 0.74 |

Notes : *Um* = *Machine Utilisation*
*Uc* = *Average Utilisation Of The Machines In A Cell*
*Ufms* = *Average Utilisation Of The Cells In The Modelled FMS*

### Average Utilisation Of Each Type Of Machines

| | Simulation Observation | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | MEAN |
| **Machine Type 1** | 0.88 | 0.92 | 0.84 | 0.78 | 0.92 | 0.82 | 0.85 | 0.96 | 0.82 | 0.89 | 0.85 | 0.89 | 0.87 |
| **Machine Type 2** | 0.79 | 0.95 | 0.74 | 0.80 | 0.95 | 0.81 | 0.81 | 1.00 | 0.84 | 0.85 | 0.87 | 0.85 | 0.85 |
| **Machine Type 3** | 0.64 | 0.77 | 0.72 | 0.73 | 0.83 | 0.75 | 0.70 | 0.84 | 0.79 | 0.80 | 0.75 | 0.65 | 0.75 |
| **Machine Type 4** | 0.44 | 0.43 | 0.42 | 0.35 | 0.45 | 0.34 | 0.40 | 0.47 | 0.41 | 0.40 | 0.40 | 0.42 | 0.41 |
| **Machine Type 5** | 0.80 | 0.87 | 0.83 | 0.77 | 0.96 | 0.75 | 0.81 | 0.86 | 0.83 | 0.91 | 0.79 | 0.78 | 0.83 |

## THE "HYBRID" AUCTION-BASED SCHEME - D3

### AVERAGE WAITING TIME OF TASKS AT MACHINES IN EACH CELL

| | | Simulation Observation | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | MEAN |
| Cells | Machines | *Dm* | *Dm* | *Dm* | *Dm* | *Dm* | *Dm* | *Dm* | *Dm* | *Dm* | *Dm* | *Dm* | *Dm* | |
| 1 | 1 | 6.35 | 5.68 | 2.93 | 2.99 | 6.61 | 4.27 | 4.19 | 7.38 | 4.54 | 5.79 | 8.08 | 10.20 | 5.75 |
| | 2 | 7.88 | 11.77 | 3.55 | 4.63 | 6.60 | 5.93 | 4.92 | 18.69 | 5.51 | 5.79 | 7.21 | 7.00 | 7.46 |
| | 3 | 0.72 | 1.34 | 1.75 | 1.75 | 2.41 | 2.84 | 3.55 | 2.91 | 2.50 | 2.15 | 2.41 | 1.71 | 2.17 |
| | 4 | 0.74 | 0.74 | 0.37 | 0.61 | 0.64 | 0.41 | 0.78 | 0.61 | 0.27 | 0.67 | 0.83 | 0.52 | 0.60 |
| 2 | 2 | 3.60 | 8.85 | 1.71 | 2.27 | 5.36 | 4.85 | 4.05 | 13.83 | 4.89 | 2.76 | 4.03 | 6.00 | 5.18 |
| | 4 | 0.13 | 0.40 | 0.11 | 0.11 | 0.87 | 0.11 | 0.38 | 0.32 | 0.09 | 0.68 | 0.41 | 0.13 | 0.31 |
| | 5 | 6.08 | 7.23 | 6.89 | 5.92 | 14.00 | 5.08 | 7.49 | 6.67 | 9.76 | 8.03 | 5.96 | 7.16 | 7.52 |
| 3 | 1 | 4.85 | 5.07 | 3.05 | 1.95 | 6.64 | 5.38 | 3.26 | 8.52 | 2.15 | 3.71 | 9.48 | 12.07 | 5.51 |
| | 2 | 1.88 | 10.37 | 1.07 | 1.03 | 4.67 | 3.21 | 2.78 | 13.93 | 2.44 | 0.78 | 2.04 | 4.12 | 4.03 |
| | 3 | 1.81 | 2.75 | 2.70 | 3.05 | 3.06 | 3.68 | 3.41 | 4.22 | 3.82 | 4.36 | 3.52 | 2.31 | 3.22 |
| | 4 | 0.04 | 0.00 | 0.00 | 0.00 | 0.44 | 0.00 | 0.00 | 0.12 | 0.23 | 0.00 | 0.00 | 0.07 | 0.08 |
| 4 | 5 | 2.89 | 2.48 | 3.51 | 2.38 | 11.87 | 2.98 | 3.38 | 3.39 | 3.06 | 5.83 | 2.01 | 1.80 | 3.80 |
| | 5 | 1.45 | 1.92 | 1.99 | 3.13 | 10.11 | 2.79 | 2.64 | 1.18 | 3.11 | 3.34 | 1.07 | 1.32 | 2.84 |

Note: *Dm = Average Time Spent By A Task Waiting In The Queue (hrs)*

## THE "HYBRID" AUCTION-BASED SCHEME - D4

### AVERAGE NUMBER OF TASK ARRIVING AT MACHINES IN EACH CELL

| | | Simulation Observation | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | MEAN |
| Cells | Machines | *Am* | *Am* | *Am* | *Am* | *Am* | *Am* | *Am* | *Am* | *Am* | *Am* | *Am* | *Am* | |
| 1 | 1 | 87 | 97 | 92 | 87 | 103 | 96 | 85 | 101 | 89 | 87 | 99 | 87 | 93 |
| | 2 | 58 | 70 | 53 | 57 | 69 | 59 | 69 | 65 | 64 | 63 | 63 | 60 | 63 |
| | 3 | 52 | 59 | 50 | 87 | 61 | 59 | 55 | 69 | 55 | 66 | 54 | 45 | 59 |
| | 4 | 92 | 107 | 96 | 87 | 104 | 102 | 91 | 101 | 92 | 99 | 94 | 101 | 97 |
| 2 | 2 | 76 | 79 | 71 | 60 | 78 | 63 | 68 | 81 | 71 | 71 | 65 | 61 | 70 |
| | 4 | 43 | 54 | 43 | 39 | 53 | 36 | 47 | 55 | 43 | 47 | 44 | 43 | 46 |
| | 5 | 68 | 71 | 70 | 74 | 72 | 65 | 72 | 75 | 66 | 72 | 69 | 62 | 70 |
| 3 | 1 | 70 | 73 | 56 | 57 | 76 | 60 | 59 | 77 | 65 | 74 | 65 | 66 | 67 |
| | 2 | 52 | 58 | 38 | 57 | 68 | 53 | 58 | 75 | 58 | 60 | 63 | 60 | 58 |
| | 3 | 77 | 86 | 81 | 78 | 95 | 74 | 85 | 77 | 83 | 85 | 86 | 80 | 82 |
| | 4 | 9 | 8 | 8 | 6 | 17 | 7 | 7 | 17 | 13 | 10 | 11 | 9 | 10 |
| 4 | 5 | 55 | 63 | 54 | 53 | 68 | 44 | 55 | 61 | 61 | 55 | 56 | 45 | 56 |
| | 5 | 46 | 51 | 40 | 59 | 66 | 47 | 47 | 42 | 45 | 60 | 43 | 40 | 49 |

Note : *Am = Number of Task Arriving At Each Machine*

### Number Of Tasks Arriving At Each Machine Type

| | Simulation Observation | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | MEAN |
| **Machine Type 1** | 157 | 170 | 148 | 144 | 179 | 156 | 144 | 178 | 154 | 161 | 164 | 153 | 159 |
| **Machine Type 2** | 186 | 207 | 162 | 174 | 215 | 175 | 195 | 221 | 193 | 194 | 191 | 181 | 191 |
| **Machine Type 3** | 129 | 145 | 131 | 165 | 156 | 133 | 140 | 146 | 138 | 151 | 140 | 125 | 142 |
| **Machine Type 4** | 144 | 169 | 147 | 132 | 174 | 145 | 145 | 173 | 148 | 156 | 149 | 153 | 153 |
| **Machine Type 5** | 169 | 185 | 164 | 186 | 206 | 156 | 174 | 178 | 172 | 187 | 168 | 147 | 174 |
| **Machine Type 4 (Without cell 4)** | 135 | 161 | 139 | 126 | 157 | 138 | 138 | 156 | 135 | 146 | 138 | 144 | 143 |