Design and Development of a Hybrid Flexible Manufacturing System

A thesis presented in fulfillment of the requirements for the degree of Master of Technology at Massey University

Volume 1

By

Matthew J Jolly

1999

MASSEY

University

Abstract

The ability of a manufacturing environment to be able to modify itself and to incorporate a wide variety of heterogeneous multi-vendor devices is becoming a matter of increasing importance in the modern manufacturing enterprise. Many companies in the past have been forced to procure devices which are compatible with existing systems but are not as suitable as other less compatible devices. The inability to be able to integrate new devices into an existing company has made such enterprises dependent on one vendor and has decreased their ability to be able to respond to changes in the market. It is said that typically 60% of orders received in a company are new orders. Therefore the ability of a company to be able to reconfigure itself and respond to such demands and reintegrate itself with new equipment requirements is of paramount importance.

In the past much effort has been made towards the integration of shop floor devices in industry whereby such devices can communicate with each other so that certain tasks are able to be achieved in a single environment. Up until recently however much of this was carried out in a very much improvised fashion with no real structure existing within the factory. This meant that once the factory was set up it became a hard-wired entity and extensibility and modifiability were difficult indeed. When formalised Computer Integrated Manufacturing (CIM) system architectures were developed it was found that although they solved many existing shortcomings there were inherent problems associated with these as well. What became apparent was that a fresh approach was required that took the advantages of existing architectures and combined them into an new architecture that not only capitalised on these advantages but also nullified the weaknesses of the existing systems.

This thesis outlines the design of a new FMS architecture and its implementation in a factory environment on a PC based system.

Table of Contents

VOLUME 1:

VOLUME 2:

List of Tables

List of Figures

Acknowledgements

I would like to thank all the people who have been involved in the research and development of the Hybrid FMS system over the last few years.

In particular I would like to thank my two supervisors Professor Don Barnes and Dr Saeid Nahavandi for their invaluable support and their vision in the project guidance.
I would like to thank Saeid Nahavandi for the many hours of assistance he gave to the project and also the excellent guidance and morale support given in times of crisis.

The contribution given to the project by Mr Len Chisholm (Computer Systems Engineer) cannot be ignored. Len spent many hours of his own time voluntarily assisting with setting up the required computer network systems and also assisting in the many hours of system debugging, without his contribution the success of the project could not be guaranteed.

I would like to thank Jill Bracegirdle for her help in formatting and printing this document.

Lastly, but certainly not least, I would like to thank my wife for the inspiration and support that she provided and also for the burden she sustained due to the many lonely nights endured whilst I completed my studies.

An old man going along a lone highway came at the evening, cold and grey, to a chasm - vast, deep and wide.
The old man crossed in the twilight dim.
The depths below held no fear for him.
And he stopped when safe on the other side,
And built a bridge to span the tide.
"Old Man!" said a fellow pilgrim near,
"You are wasting your time with building here.
Your journey will end with the close of day,
And never again will you pass this way.
You've crossed the chasm both deep and wide,
Why build this bridge on the eventide?"
The old man lifted his grey old head.
"Good friend in the path I have come", he said
"There followeth after me today,
A youth whose feet must pass this way.
This chasm which now is as naught to me,
To that unlearned youth a pitfall may be.
He, too, must cross in the twilight dim.
Good friend ..... I am building this bridge for him".....

"The Builder" (Author Unknown)

Glossary

AGV	Automated Guided Vehicle
Capability	A short item of logic used to carry out a given item of function
Centralised Control	A system in which decisions are made at one location within the FMS/CIM System.
CIM	Computer Integrated Manufacturing
DFMS	Distributed Flexible Manufacturing System
Distributed Control	A system in which decisions are made by a global decisonmaking process involving all entities (or some of the entities) in the FMS/CIM system
DIT	Data Image Table
EFT	Earliest Finishing Time
Fault Tolerance	The ability of a software based system to cope with faults and/or unexpected events
FMS	Flexible Manufacturing System
GUI	Graphical User Interface
HFE	Human Functional Entity
Loose Coupling	The situation whereby units of modular software have minimal interaction in terms of data transfer
MFE	Machine Functional Entity
NC	Numeric Control
PIL	Program Invocation Loop.
Proprietary Controller	The controller used to control the actual physical behaviour of the manufacturing process using a Numeric Control script
SCADA	Supervisory Control and Data Acquisition
SFE	Specific Functional Entity
SPT	Shortest Processing Time
VMD	Virtual Manufacturing Device. A software model that generically emulates the behaviour of a real world device and can be mapped onto it thereby giving real control.

Chapter 1

Literature Review

1.1 Introduction

As indicated in the abstract the research being carried out is concerned with the design and development of a Hybrid Control System for a Flexible Manufacturing System (FMS). This chapter gives a review of all important subject matter relevant to the development of such an FMS. Areas that will be covered include the types of manufacturing systems currently being implemented, an overview of cellular manufacturing and flexible manufacturing, the architectures that are currently being implemented, an overview of the CIM-OSA project as well as current scheduling methods that are being implemented.

1.2 Types of Manufacturing Systems

Generally speaking a manufacturing system can be broadly classified as a series of processes each with a given sized queue of jobs required to be completed in front of them. The dynamics of the system are stochastic in nature[2] with the various features of the system such as job interarrival times, processing times and set up times being distributed about a mean value and this tends to be the case irrelevant of the system type.

There are four main classical types of manufacturing systems, these are

- Project shop
- Job Shop
- Flow Shop
- Continuous System

1.2.1 Project Shop

In a project shop the product being manufactured remains fixed and the parts that constitute the finished product are brought to it, such is the case in the ship building industry and aircraft industry.

1.2.2 Job Shop

A job is a work part or component to be manufactured, such manufacturing includes activities or operations such as turning, milling, grinding and forging. The job therefore consists of an operation or

number of operations to be executed on a number of distinct processes, when all operations are complete the job is complete.[1]

A job shop is characterised as one in which each one of a number of job's operations visit a sequence of machines in order that each job is completed fully.[2]
Typically a job shop exists in two forms:[2]

- **Open Shop** - one in which the job orders received are from outside concerns (make to order)
- **Closed Shop** - one in which the job orders are received internally (make to stock)

1.2.3 Flow Shop

The flow shop is one in which the processes are ordered in the sequence that the parts passing through them are to be processed, the idea being that this type of shop fosters high throughput. Typical examples of such a manufacturing system is the motor vehicle assembly line and in the meat processing industry where carcasses are transferred along a chain requiring various operations to be performed at various stations.

1.2.4 Continuous System

In contrast to the other types of flow shop which work on or produce discrete parts the continuous system involves the production of gases, liquids or slurries. As in the flow shop processes are arranged in a given order in which raw materials are fed in one end to produce the finished product at the other end.

1.2.5 Summary

The five systems as mentioned above are the classical types of manufacturing systems however they all have significant weaknesses associated with them. New developments in manufacturing systems have resulted in the development of cellular manufacturing systems that have benefits in the areas of improved product quality, reduced work-in-process, reduced lead times, reduced handling costs and improved operator efficiency and flexibility.[3]
ise the goals of the company.

1.3 CIM-OSA Computer Integrated Manufacturing

1.3.1 Introduction

Today worldwide competition between manufacturing enterprises has significantly increased. In order to survive, to grow and to maintain the margins of profit they are required to be able to compete in terms of quality, price and delivery time.[4] One of the weapons a manufacturing enterprise can use to consolidate its competitive position is the implementation of Computer Integrated Manufacturing (CIM).

The CIM-OSA is a total enterprise CIM design framework developed by a research project set up by the Commission of the European Communities in the framework of the Esprit program.

CIM-OSA provides a total enterprise modelling framework - in the framework the overall business requirements of the enterprise are captured and from this the physical system is derived. The system is the result of the recognition for the necessity of a structured top down approach from the derivation of business requirements to the design for those requirements and to the actual physical implementation of the system.[4] Other research carried out at Purdue University, USA, by the CIM Reference Committee has created an overall model of the enterprise using similar types of generic framework.[5]

1.3.2 Features a Quality CIM system should exhibit

- An optimal use of all the resources in an enterprise require that they be integrated into a CIM system, and that the result of the CIM integration is such that the flow of control and information is structured to model some proven heuristic.

  Structured control gives the following benefits.[4]

- Provides decision makers with up to date information allowing more accurate dynamic decisions to be made.
- Higher planning flexibility.

- Reduced work in progress and inventory.

Structured information flow results in a greater ability to access data and more predictable structured data results. This results in the following advantages:[4]

- Improved ability to exchange data between the stages involved in a product's life cycle allows better integration of each stage of the cycle, resulting in shorter time span between product design and marketing.

Having a structured information flow and control together results in the following benefits

- Improves product quality due to avoidance of errors
- Increased production flexibility
- Increased system flexibility

1.3.3 CIM-OSA Integration Requirements

There are two aspects of CIM related features that have to be considered in respect of CIM-OSA.

1.3.3.1 CIM Integration Aspects

CIM-OSA outlines a number of integration aspects that must be addressed in an enterprise wide CIM implementation in order for the developed system to reflect the integration in a comprehensive fashion. These aspects include:[4]

- Physical System Integration - Physical System Integration level is the lowest level and involves the physical interconnection of systems such as robots, CNC devices and computers and is one of the first aspects of integration that must be addressed as without the lowest level of integration successfully addressed, higher levels cannot be achieved.
- Application Integration - the Application Integration level deals with the information exchange between applications, the portability of applications between different devices and the distribution of applications and the use of standardised interfaces between applications. Layer 7 of the ISO

(International Standards Organisation) seven layer model deals with standards regarding the design and implementation of an application interface.[6-7]

- Business Integration - This is the highest level of all the aspects involved and addresses the integration of all the business processes at the corporate level, the business processes including concerns such as design, finance, accounting, marketing and production planning. It is the business level that is the driving force behind the requirements of the CIM design and implementation as it dictates the type of applications and physical systems that must be integrated.

1.3.3.2 CIM-OSA System Evolution

CIM-OSA treats the Business Integration needs as the motivation for the integration of the lower levels of the system to implement those needs. Therefore the CIM system is required to be updateable so that as the enterprise's business requirements change the system can change. This is achieved through the implementation of a generic structure and generic building blocks.

The combination of the three integration features are represented in the model[4] as shown in Figure 1.1

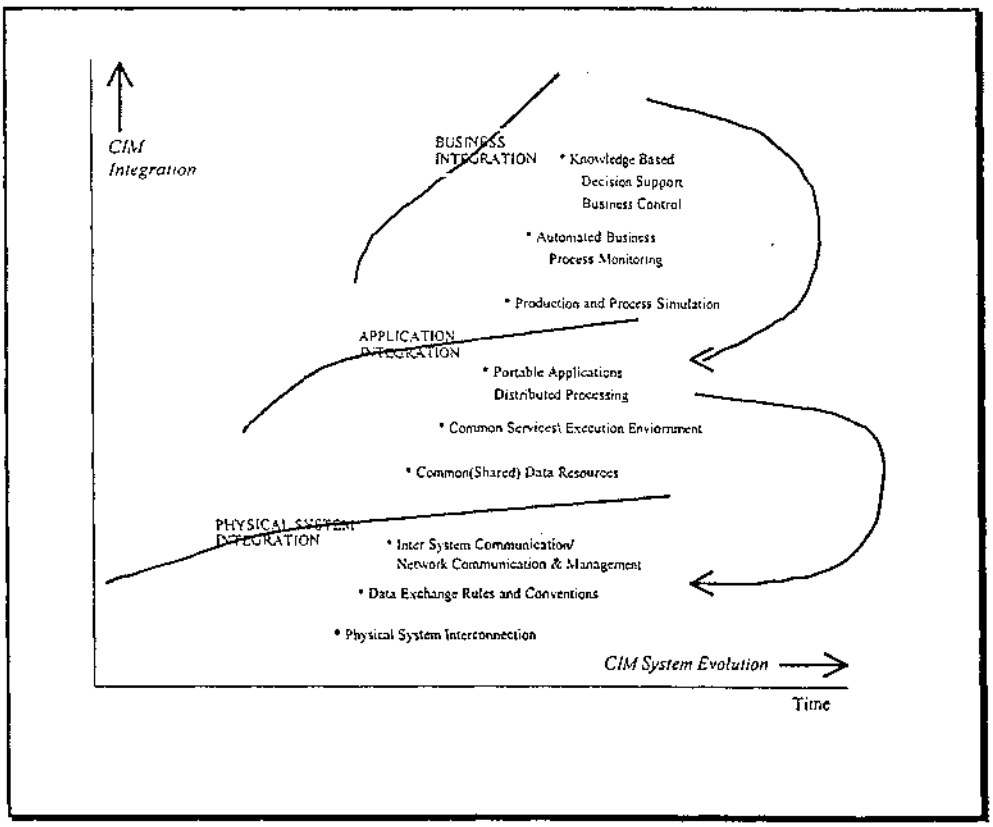


Figure 1.1 - Steps towards the integrated enterprise

The arrows in the model are representative of the derivation of the requirements of the lower level from the one above - this follows a logical derivation of the business requirements of the system, to the functional application requirements at the middle level to the derivation of the physical requirements at the lowest level.

1.3.4 Building a CIM-OSA Model

In section 1.4.3 we discuss the aspects which are required to be integrated in a CIM environment, however every business enterprise is different. Therefore we are required to construct a CIM-OSA implementation model for each enterprise considered and it is from the ensuing implementation model that the implementer finds solutions regarding what aspects are required to achieve the enterprise goals and how they are to be achieved.

The CIM-OSA model implements a top down approach[4,8-9] to CIM system design and implementation as shown in Figure 1.2. As can be seen from Figure 1.2 there are three levels of system modelling involved for the successful implementation of CIM for an enterprise under the CIM-OSA modelling paradigm.

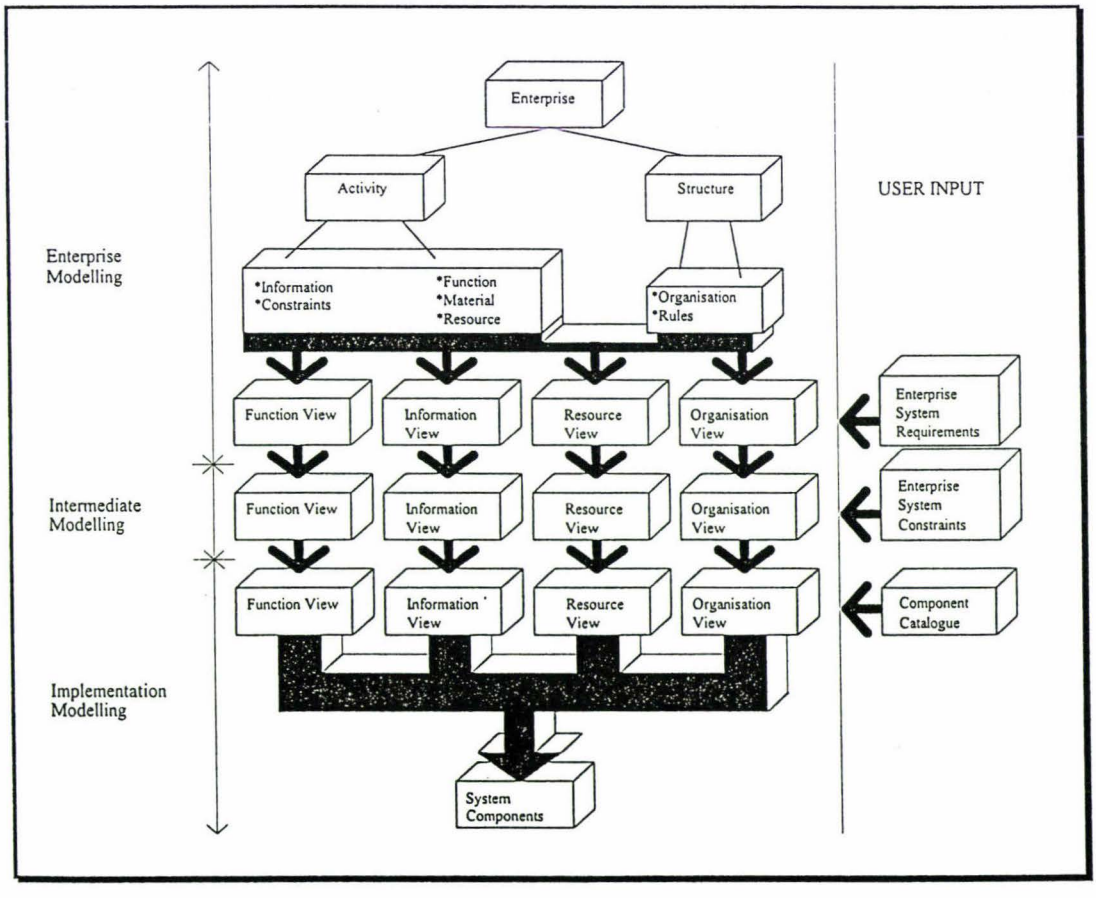


Figure 1.2 - Levels of Modelling under the CIM-OSA Paradigm

The system takes a top down approach to overall model formulation by modelling the goals of the enterprise at the top level (Enterprise Modelling) and then by using a system of derivation arriving at the implementation model at the bottom level which details how the specific implementation will be carried out.

Details of what is involved at each level of the modelling process is as follows:

1.3.4.1 Enterprise Modelling Level

The enterprise modelling level describes in a business sense and terminology what needs to be done within the enterprise. This is achieved in the following manner

- Identify all the business processes required to achieve the enterprise objectives and break each business process into a hierarchy of business processes required to achieve each main business process. Each business process is expanded into a hierarchy of spawned business processes required to produce results for the higher level business process, this is shown in Figure 1.3 as below

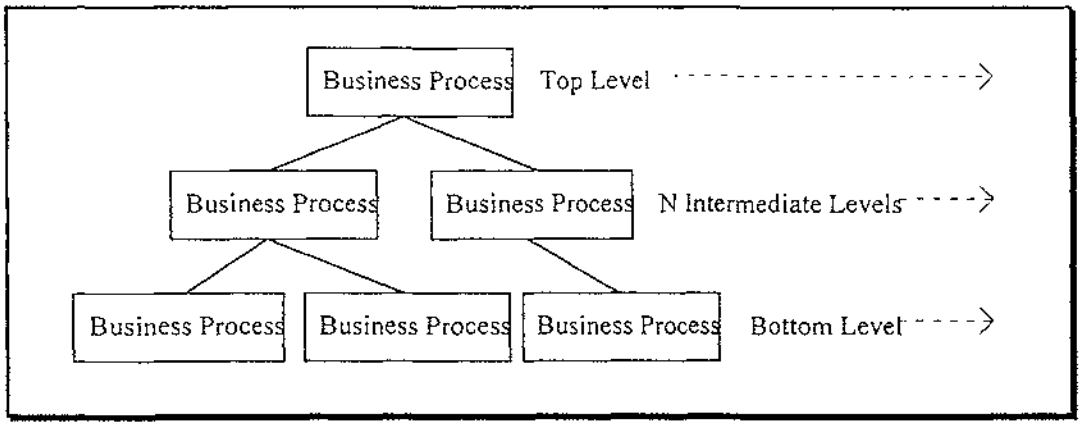


Figure 1.3 - Business Process Derivation

Therefore for each business process required to fulfil the goals of the enterprise we break the business process into a hierarchy of business processes. When the all the business processes has been derived we have all the business processes at the different levels of the hierarchy required to achieve the goals of the enterprise.

Each Business Process at each level is made up of a number of Enterprise Activities, each Enterprise Activity (EA) describing a given functionality of the enterprise, these EA's are not part of a given business

process but are used by it (as a function is), meaning that any business process can use any EA from the pool of EA's available.

Therefore looking at the lowest level of Figure 1.3 we have the situation as regards the relationship between two business processes as in Figure 1.4

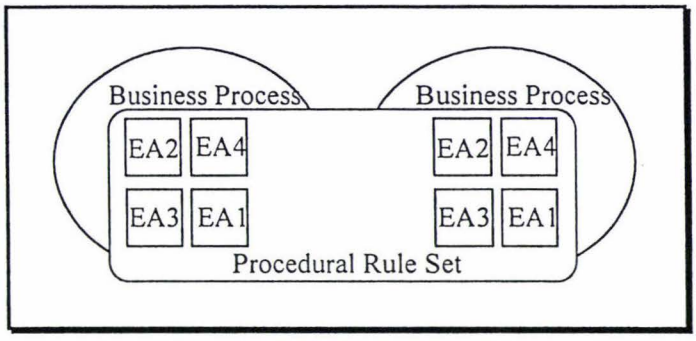


Figure 1.4 - Business Process Relationship Model

In Figure 1.4 the concept of the procedural rule set is introduced. Each business process has its own procedural rule set, the function of the procedural rule set is to define the flow of control between the EA's within a Business Process and also the flow of control between the Business Process housing the procedural rule set and the Business Process it is chained to.

This separation of functionality (EA's) and behaviour (Procedural Rule Set) means that it is easy to revise the overall CIM system design as depicted in Figure 1.1 because as the system is required to be changed to meet changing circumstances it is simply a matter of modifying the procedural rule set and constituent EA's for each Business Process.

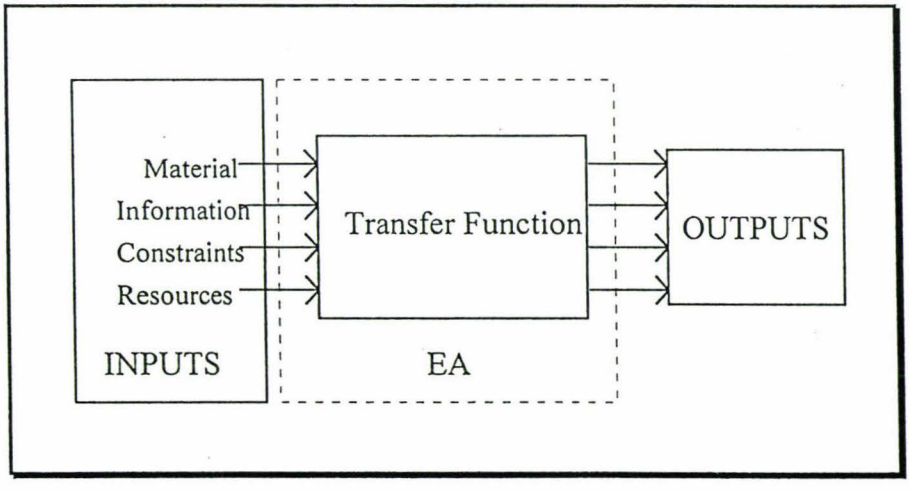


Figure 1.5 - Enterprise Activity Input Output Model

The enterprise model is finalised by breaking each EA down into the inputs required for each EA, the transfer function and the required outputs. The required types of inputs, the outputs and transfer function are shown in Figure 1.5.

The input types are:

- Material - types of materials required to input into the EA.
- Information - what type of information is required to be input to the EA
- Constraints - these are the types of constraints on the transfer function such as process definitions, work instructions and product information.
- Resources - these are the means required to carry out the transfer function such as the type of process required, equipment and tools and application programs.

For each EA required in the system the required inputs are defined in terms of the above input types and from this we can identify all the required materials, information, constraints and resources required to meet all the functions required for the enterprise. From this the Enterprise Modelling level has been completed as all the Enterprise System Requirements have been derived from the viewpoint of the Function, Information, Resource and Organisation aspects.

Therefore it is the implementation of these generic building blocks that give the system flexibility and extensibility. One important thing to note is that functionality (the EA's) and behaviour (Procedural Rule set) are clearly separated meaning that as the requirements of the enterprise change it is easy to change the system to meet those changes.

1.3.4.2 Intermediate Modelling Level

The enterprise modelling level describes what needs to be done within the enterprise in a business sense in terms of the function, information, resource and organisational view (Jorysz and Vernadat, 1990). The Intermediate Level of modelling deals with the design and organisation of the CIM-OSA system and focuses mainly on the information, resource and organisational view.[9]

- **The Information View**

All data used, processed or converted in the enterprise is represented at the intermediate level in terms of the Enterprise Object (EO), the EO being a generic handling system for any data being dealt with. The information view at the requirements level of the model gave a methodology for generically deriving the information needs for the specific CIM-OSA implementation for the enterprise. The Intermediate Modelling Level as explained by Klug and Tsichritzis provides a generic information modelling system defined along the three schema approach using Conceptual schemata[10]. The three schema approach classifies Enterprise Objects as consisting of three enterprise object views - internal, external and conceptual. The Conceptual view is considered at the intermediate level and deals with how the data derived at the requirements level (External View) is generically modelled (represented) in the system whereas the Internal view is developed at the Implementation Modelling level and deals with how the information is physically stored in the system.

- **The Resource View**

At the Enterprise Modelling level the types of resources required to meet the business requirements were derived by considering the inputs required to implement a Business Process. The Resource view at the Intermediate Modelling Level introduces a building block called the Logical Cell - the Logical Cell provides a mechanism to ensure the optimum grouping of equipment and resources to meet the objectives for the enterprise at the requirements modelling level. The nature of the resultant grouping may reflect a job orientated structure or business orientated one.

- **The Organisational View**

The Organisational View at the Intermediate modelling level describes the deployment of responsibility for the control and management of the resources, information and Business Processes. These responsibilities have to be arranged to fulfil the needs of the enterprise for decision making. The building block for the grouping of responsibilities within the enterprise is the Organisational Cell and provides a generic, structured decision making and control system on which decisions about the procurement of resources and subsequent execution of Enterprise Activities can be made.

1.3.4.3 Implementation Modelling Level

The implementation model is a description of the physical CIM system. It is created by examining the design requirements with respect to the different views of the Intermediate Model and making technical

choices about how the functionality is to be provided (functional view), how and where the information is to be stored (Informational View) and how the organisational structure is to be implemented.

The implementation level is mainly concerned with the development of the functional view. At the Enterprise Modelling level the Business Processes and Enterprise Activities were derived to meet the strategic needs of the enterprise, from this generic description which is implementation independent the implementation model is created.

As is shown in Figure 1.6 these Business Processes are controlled by an *executable* Procedural Rule set for each Implemented Business Process.

The Enterprise Activities (EA) are executable functions that can be used by a Business Process and are drawn from an EA Pool - the flow and action from one EA to another being dictated by the Procedural Rule set for that EA. Each Enterprise Activity describes the behaviour of part of the Business Process using a Transformation Function, the Transformation Function uses a set of generic Enterprise Objects as input describing the nature of the input required for the EA to carry out the required logical function.

The Transformation Function describes the behaviour of each required EA. However it is the implemented functional operations that model the behaviour of the required external physical components. Each of these modelled behaviours then interact with the external physical world to carry out the required actions to fulfil the Transformation Function.

The complex structure as shown in Figure 1.6 [9] has to be mapped onto a Physical Environment consisting of computers, machines, people and linked by communication systems. The physical world however involves a certain degree of required distributed functionality - the Implemented Functional Operation may have to communicate with a number of different Physical Devices, it does so using the Functional Entity Concept.

The functional Entity is an object able to transmit, receive and process data, it describes a physical world behaviour. The Functional Entity generically describes physical real world behaviour and not actual physical equipment - it uses a set of defined objects to describe and model the *generic behaviour* of actual equipment, this gives CIM-OSA the required flexibility at the implementation level. The functional entities are able to communicate with each other using transactions based on the MMS system. The MMS system uses a strict protocol of transmission of data packages using the client-server system, the data packages hold the messages being transmitted and each message is stored in predefined slots in the PDU (Protocol Data Unit) depending on the type of message being dealt with - this gives a common generic language between the Functional Entities as well as between the Functional Entities and the Physical

Devices. There has been considerable work already done on the benefits and implementation of such generic messaging systems.[7,11,15]

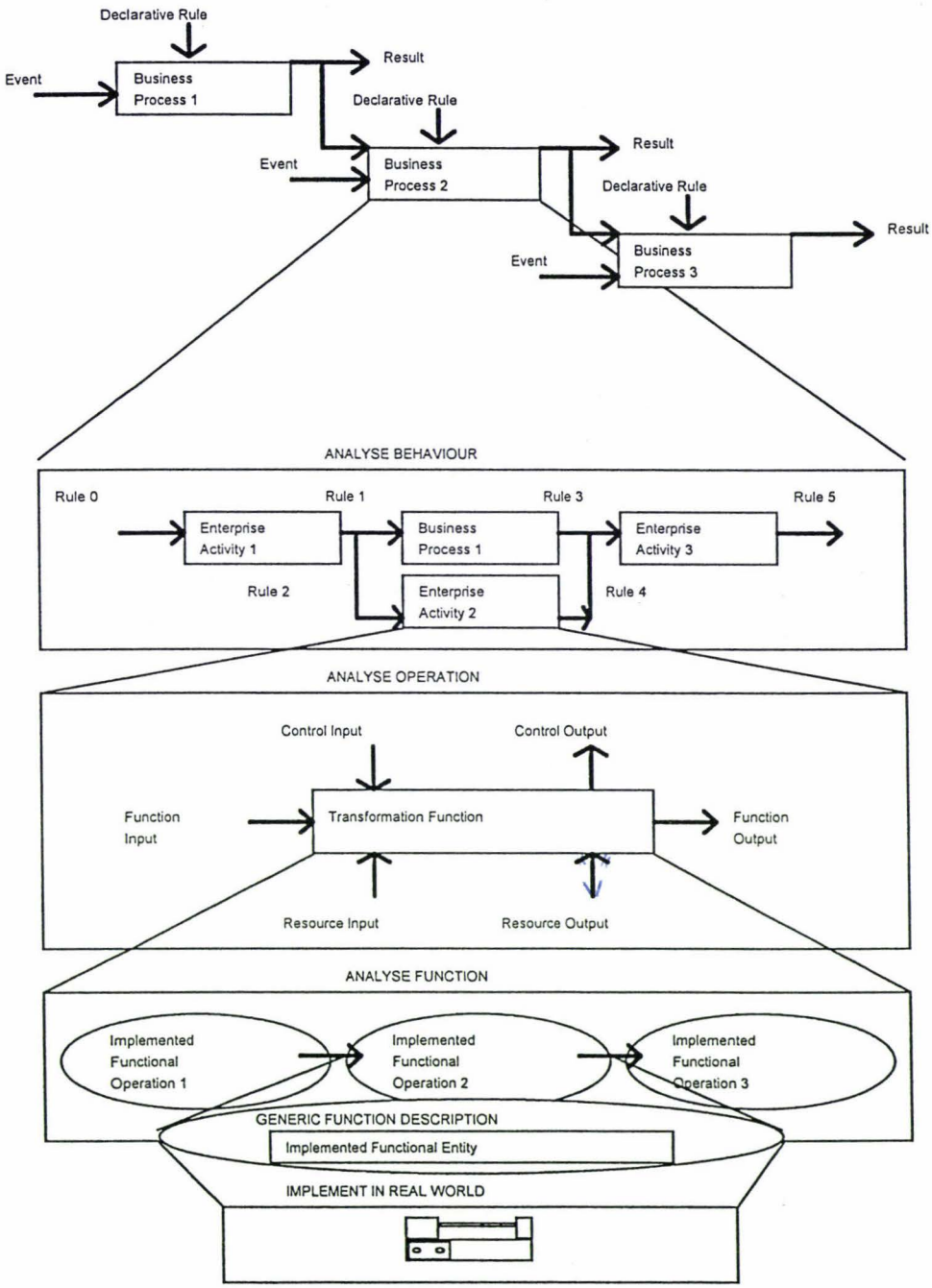


Figure 1.6 - Mapping CIM-OSA system onto Real World Environment

The physical environment is therefore described as a system of communicating Functional Entities that communicate with each other and drive the real physical devices in such a way as to meet the goals of the enterprise. CIM-OSA splits the Functional Entities into a number of sub categories:

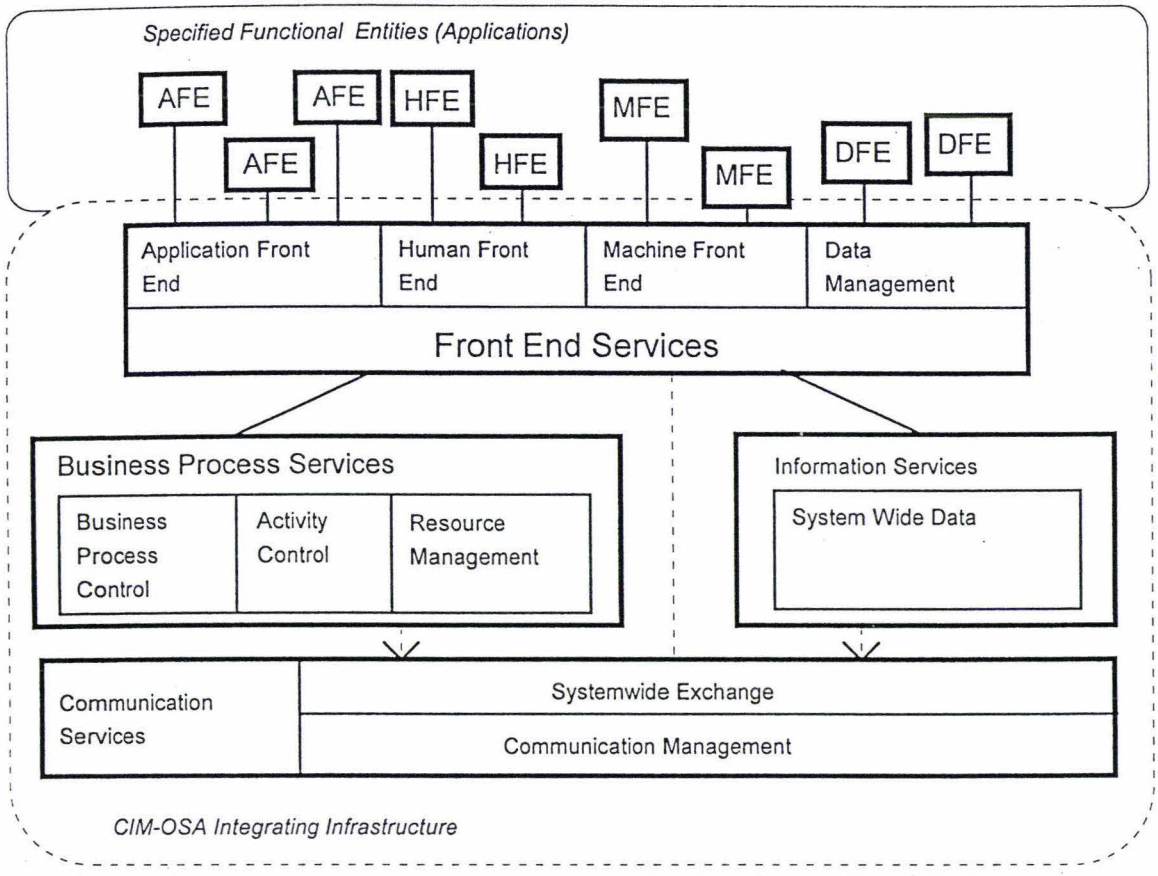
- Implemented Machine Functional Entities - describes generically the behaviour of a physical machine in such a way as this behaviour can be mapped onto a real machine to drive it as required.
- Implemented Human Functional Entities - model the behaviour of a human required to carry out the required Functional Operation in such a way as it can be mapped onto a real person.
- Implemented Application Functional Entities
- Implemented Data Storage Functional Entities
- Implemented Communication Functional Entities

Therefore it is possible in this way to model a complete organisation in terms of requirements to design to implementation by invoking such a top down approach, however it must be understood that the implementation of such a model involves a generic element - this is the role of the Integrating Infrastructure.

1.3.5 The Integrating Infrastructure

The paradigm of the Integrating Infrastructure is that there are a number of common services that are part of any corporate organisation and that there is a need not only to provide them but also implement them in a generic fashion - the ISO seven layer model[7,15] is a graphic realisation of this. The requirements of the model at the implementation level have been established, the integrating infrastructure binds the model together.

The model of the structure of the Integrating Infrastructure[8] is shown in Figure 1.7.



AFE: Application Functional Entities
MFE: Machine Functional Entities
DFE: Data Functional Entities
HFE: Human Functional Entities

Figure 1.7 - Integration of Applications through the services of the Integrating Infrastructure.

1.3.5.1 Front End Services

The Front End services provide a generic mechanism for interrogating the state of the specified functional entity and reacting to its state. The logic being along the lines of a 'generic' PLC - it provides the motor for the execution of physical actions or commands required of the machines, humans, applications and data management systems. The front end services use a Virtual Manufacturing Device[8,14] to emulate the status of the Specified Functional Entity and depending on the status a control decision is made using a generic control loop and a command is sent using generic MMS protocol to effect the behaviour of the real world device.

1.3.5.2 Business Process Services

There are three types of services under this umbrella, these being:

- Business Process Control Service - This service controls the execution of the actual business processes being executed by the SFE's by controlling the order of execution of the business processes and enterprise activities.[9]

- Activity Control Service - Controls execution of enterprise activities.

- Resource Management Service - Dynamically schedules all the resources required in the execution of the Enterprise Activities.

1.3.5.3 Information Services

Theses are generic services responsible for the addressing, storing , locating, retrieving and handling of data - the system wide data service also has to manage the context of the application-specific data according to the information model of the particular enterprise.

1.3.5.4 Communication Services

This service is generic and provides for system wide data handling and communications management providing access to OSI and other communication environments. The service has the following attributes:

- Location Transparency - physical location of services and Functional Entities are not important.

- Failure Transparency - Has mechanisms that provide strategies for the various failure modes that are not obvious and do not disrupt the system when they occur.

- Access Transparency - allows for the different types of access that are used by various communication systems.

- Performance Transparency - minimises the effect as much as possible of intermittent limitations in transmission performance.

1.4 CIM Control Architectures

Research into the design and implementation of computer integrated manufacturing systems has shown that a system is only as good as the architecture that it is developed under and that for a system to perform well it must be based on a sound structure.[1] The control architecture must be able to handle a complex and dynamic environment and must also be robust - this can be a difficult problem to solve but the selection of an effective architecture can go a long way. The challenge has been to design a system architecture that addresses the following performance characteristics:

- High Extensibility\Modifiability
- High Reliability\Fault Tolerance
- High Reconfigurability\Adaptability

By observing these various features it can be appreciated that different systems designed under various system architectures will exhibit different levels of performance in these areas and an understanding of these levels under various schemas is important so as to get an idea of the strengths and weaknesses of the different architectures. The design and implementation of the control architecture is most important as it breaks the overall control system into functional entities and the behaviour of these functional entities and their interactions determines the system's degree of overall performance in terms of the performance characteristics mentioned.

A number of different architectural structures are in existence and have been applied to CIM/FMS in the past and an evaluation of their form is required to gain an appreciation of why alternative control architectures may need to be considered.

Dilts *et.al* outlines four main types of control architecture are illustrated in Figure 1.8[1]

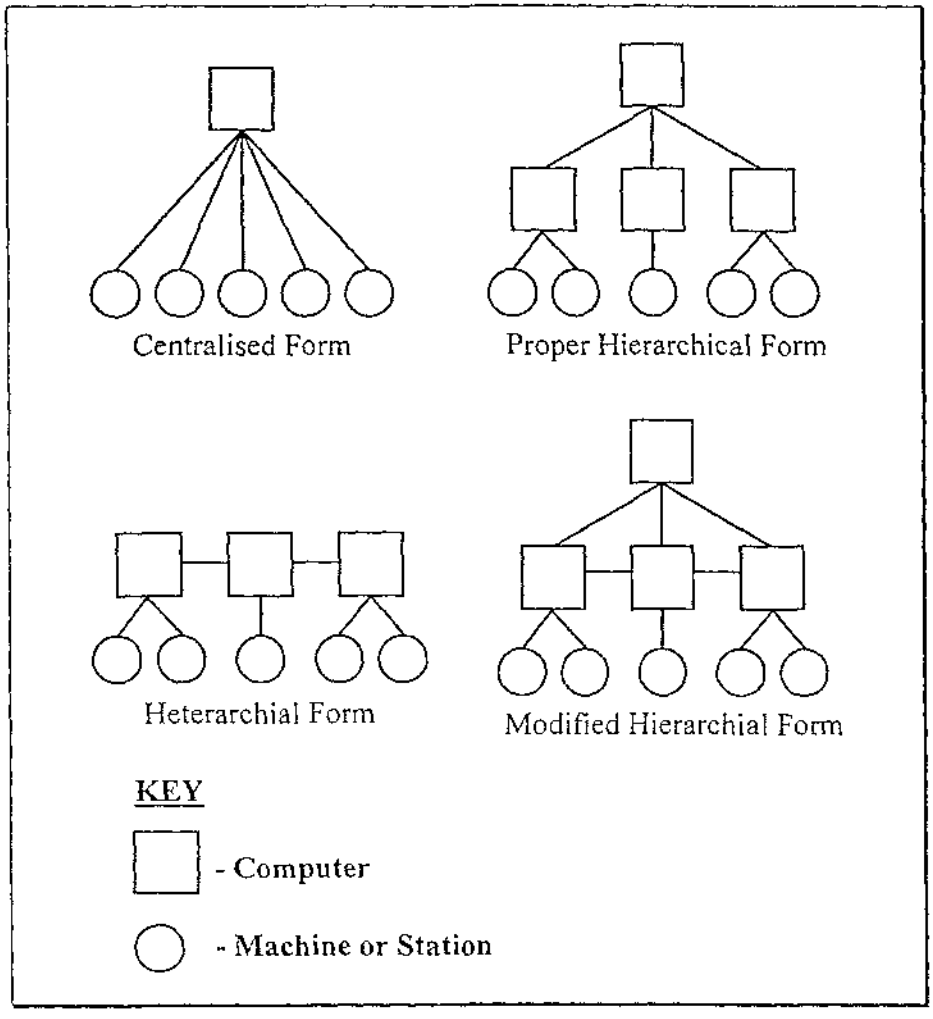


Figure 1.8 Common CIM System Architectures

1.4.1 The Centralised Form

The centralised architecture was the first attempt at CIM and generally involves the implementation of a centralised mainframe computer that acts as a manager and monitors the state of all processes in the factory.[1,16] The machine controllers execute commands received directly from the mainframe and it is these commands that dictate the manufacturing of the part directly from those controllers. In this environment the mainframe is responsible for the supervision of the overall system and the collection of data using shop floor sensors and uses this data to assess the overall enterprise and make decisions accordingly in a centralised fashion. Veeramani, Duffie and others point out that the major problem with the Centralised approach is the fact that failure of the centralised database can cripple the system.[1,16] It is

possible to design redundancy in the event of this failure however the centralised database would struggle in its ability to handle effectively all the database transactions in the CMS - thus "in all but the smallest organisations , performance and cost considerations will dictate distribution and function of data".[17]

1.4.2 The Proper Hierarchical Form

The shortcomings in the centralised control architecture coupled with advances in computer technology resulted in the development of the Proper Hierarchical form. The Proper Hierarchical (or more simply Hierarachial form) form of architecture was developed to emulate the Top-Down management structure that existed in many companies.[18,19] Strategic goals are set at the top most level of the hierarchy and the requirements are transmitted between levels top-down in a Master-Slave relationship, all activities of the subordinate level being dictated to by the supervisor level. The computer at the highest level of the hierarchy is responsible for the setting of the overall goals of the system and dictates and coordinates the lower levels in such a way as to achieve the required goals, the number of levels to the lower most level being undefined.[1,16,18,20-23] The only bottom up communication that is allowed in this architecture is for the reporting of the states of the various processes at the lowest level of the hierarchy to the highest level of the hierarchy.[1,16,18,21-23] Under the hierarchical control architecture the control information is input at the highest level of the structure and is broken down into a more detailed commands at the lower levels - modules at each level make decisions based on the commands that are received from higher levels and feedback received information from lower levels.[1,16,18,21-23] No information exchange occurs at the lower levels between the lower level modules rather communication only occurs with the level above the module concerned, the lower modules relying on supervision from the top to coordinate the required actions between them.[1,16,18,20-23] The lowest level of the architecture is responsible for shop floor activities such as machine control and acquiring sensory information, the top levels are responsible for control and coordination of the entire system.[20] Duffie, Chitturi and Mou state that a positive aspect resulting from the clear lines of communication obtained is that the response times of the system are fast but that the major problem with the architecture is that the structure of the system is fixed early in the design making unforeseen modifications difficult.[18,23] Duffie *et.al* and other researchers say that another problem with the hierarchical form is that once a system gets above a certain size and complexity it is impractical to define precise chains of command and operational detail - fault tolerance becomes expensive and difficult to allow for.[16,18,21-22,24] Hatvany himself made the statement that "no algorithms can be written which foresee every possible failure mode of a highly complex system, nor can remedial strategies be deterministically designed for every situation."[19]

1.4.3 The Modified Hierarchical Form

With the pure hierarchical form there was a major disadvantage - at the subordinate level there was no autonomy meaning that if a "break in the chain of command" occurred anywhere from the top of the structure to the bottom then the whole system would fail. The chain of command and the slave/master relationship is maintained with the Modified Hierarchical architecture but the degree of autonomy of the slaves is increased. When the first operation is carried out on the job at the first cell enough information is passed to it by the SCS (Shop Control System) so that it can arrange for the next operation at the second cell.[1] This allows the CCS (Cell Control System) the ability to supervise the operation and frees up the SCS to carry out more important tasks.

1.4.4 The Heterarchical Form

To overcome the limitations of the previous three architectures Dilts *et al.* proposed a new type of architecture - the Heterarchical architecture.[54] The Heterarchical form is a distributed type of architecture where the control of the overall system is done so in a distributed and opportunistic fashion. Many researchers have found that there are a number of limitations associated with hierarchical architecture, one of the major problems being increased complexity as the system expands and modifiability problems.[1,16,22,23] The core of this problem is in the inflexibility of the strict command protocol and master-slave relationships and because complicated command chains exist between modules.

With the Heterarchical form no Master\Slave relationships exist as inter-cell negotiation protocols such as The Contract Net Protocol (*Tilley et.al; Shaw; Smith; Veeramani; Ramasesh*) are used to distribute processing tasks between the cells in the processing environment.[39-40,57,55,2] Global information is eliminated or minimised in this architecture[1,22-23] which means a copy of all system required data is held and updated in each cell. This approach tends to aid in the designation and implementation of a fault tolerant system - this has been discussed by a number of authors.[1,23-24] A negotiation protocol is used between CCS's to decide where the operation required to be undertaken is best carried out, thus giving any CCS the capability of refusing to carry out an operation.[1] The design of the system is such that each CCS's required software is the same giving an implicitly modular system, each module within each CCS being able to be customised to suit the requirements of the system - this allows for reduced software complexity and implicit fault tolerance.[1] Comparisons of software complexity were made by Duffie and Piper and it was found that the number of lines of code required for a system with Heterarchical architecture was significantly less than the previous three architectures discussed.[26] Gremillion and Duffie *et.al* note that the number of lines of code are a good indicator of software complexity[21,27] while Dilts et.al states that the lower level of

complexity of the Heterarchical architecture coupled with its modularity will result in a system that is easy to extend and maintain.[54] Also the cooperative nature of the CCS's will continue to operate despite the failure of one or more companion CCS. The heterarchical architecture therefore allows the CCS's to redistribute work in accordance with the requirement of load balancing or in the event of machine or CCS failure.[1] Dilts et.al states that because of the weak couplings between CCS's and the modularity of the software within the system new cells can be easily added.[54] Other authors such as Pu *et.al* back up these statements saying that modularity and component reuse are important features that are required by a complete architecture regardless of the architecture used.[29]

Duffie notes that a major feature of hierarchical systems that causes problems is the fact that each level in the system relies on complex communication between levels due to the high level of global information, whereas heterarchical systems have loosely coupled interactions between the modules in the system.[23] Programmers need only be concerned with the inner workings of the module that they are developing and the types of messages required to be exchanged with other modules, they need not be concerned with the systems within other modules - this type of approach results in fewer logical programming errors and unforeseen interactions between modules.[23] Some researchers are of the opinion that the architecture is not based on a sound management philosophy that has been proven to be effective and that the system is still in its infancy and that more work is required before it becomes a recognised and proven architecture.[16,18]

Tables 1.1-1.4 below shows a summary of each architectures features, advantages and disadvantages.

Control Architecture - Features	Advantages	Disadvantages
Centralised • Single Mainframe Computer. • Control decisions made at a single location. • Global database records all system activities.	• Access to global information. • Global Optimisation Possible. • Single Source for System Status Information.	• Slow and Inconsistent Speed of Response. • Reliance on Single Control Unit. • Difficult to Modify Control Software.

Table 1.1 - Advantages and Disadvantages of Centralised Architectures[1]

Control Architecture - Features	Advantages	Disadvantages
Proper Hierarchical Form • A modified form of centralised control. • Based on traditional company management structures. • Supervisor controls activities of subordinates. • Rigid Master/Slave relationship between decisionmaking levels. • Integrated databases at each level. • Levels of control are arranged in a pyramid structure. • Control information flows down the hierarchy, feedback flows up.	• Reduced Software development time and subsequent problems. • Improved Fault Tolerance. • The rigid structure results in faster response times than the centralised design. • Emulates the corporate management structure therefore easy to map this structure onto the architecture.	• Unreliable inter-level links between layers of control. • Limitation of lower level controllers. • Failure at intermediate level of the hierarchy results in paralysis of the subordinate. • Difficult to extend/modify. • Global decisionmaking is based on obsolete data. • Failure recovery routines have to be programmed into the system. • Not all systems can be modelled hierarchically. Lack of availability of software. • Difficult to debug system. • No clear guidelines for system design currently available.

Table 1.2 - Advantages and Disadvantages of Proper Hierarchical Architectures[1,16,18,21-24,29,57]

Control Architecture - Features	Advantages	Disadvantages
Modified Hierarchical Form • Supervisor initiates sequence of activities in subordinates. • Loose Master/Slave relationships between decisionmaking levels. • Subordinates cooperate to complete required sequence of operations.	• All the advantages of proper hierarchical control. • Increased local autonomy means that the cells do not require supervision giving increased fault tolerance. • The overall system supervisor has more time available to carry out its tasks because some of its work is off loaded to subordinates.	• Control system design is more complicated. • Communication system between subordinates can pose connectivity problems. • Limitations of low level controllers.

Table 1.3 - Advantages and Disadvantages of Modified Hierarchical Architectures[1]

Control Architecture - Features	Advantages	Disadvantages
Heterarchical Form • A distributed form of Control. • No Master/Slave relationship • Full Local Autonomy • Distributed decision making for activity coordination. • Local databases only	• Reduced software complexity because of the system modularity and improved fault tolerance. • Shorter software development time than any others architecture • Improved fault tolerance due to local autonomy. • Improved system modifiability and extensibility. • Load balancing between cells easier to obtain. • Improved scheduling employing cooperative systems. Shaw found that distriduted scheduling performed better than centralised. • Amount of communication over LAN minimised because updating of ststem.	• Technical limitations of controllers. • No standards for communications, protocols or operating systems. • High likelihood of only local optimisation. • Requires a high network capacity. • Lack of availability of software. • The architecture is not based on a proven management philosophy and is therefore risky.

Table 1.4 - Advantages and Disadvantages of Heterarchical Architectures[1,23-25]

1.4.5 Discussion

The hierarchical architecture, Duffie et.al explains, is the most commonly applied architecture in the design and development of CIM systems.[23] Duffie et.al and other authors have explained that the reasons that this architecture was picked up quickly was due to the desire to emulate classical management structures.[1,18-19,24,30] Duffie and others authors such as Veeramani explain however that as the hierarchical systems grew they became more and more complicated with increasing associated costs, also because the structure of the systems was rigid the systems tended to become fixed and difficult to modify in the future.[18,21,24,57] Early pioneer of CIM systems Josef Hatvany also made similar comments in his early work stating that for large complex systems that the theology behind hierarchical systems of specifying each and every subsystem and operational detail was impractical and difficult to achieve.[30] Fault tolerance was achieved only at very high expense in this type of architecture and then it was not always reliable.[21,24]

Many researchers started to realise that the underlying architecture was the problem and started investigating other forms of architecture - the evolutionary approach to the development of system architectures is currently pointing to the heterarchical approach.

If a heterarchical architectural design is to be followed a systematic approach as suggested by Hatvany and Jorysz *et.al* is still required so that a comprehensive and complete system is the result, in fact other authors such as Veeramani *et.al* have noted that when the first systems were developed there were no systematic development guidelines to follow and therefore early systems were poor.[9,16,19] Hatvany suggested that one of the best ways to assure a successful design is to take a top-down integrated approach early in the design process, however he also warned that following the typical strict orderly hierarchical structure can lead to catastrophic collapse of the system in the event of failure.[19] The ESPRIT project takes this suggestion of Hatvany's and integrates it with the idea of a distributed non-hierarchical architecture and goal driven global optimisation to derive the CIM-OSA system[4,9] This paradigm helped to combat one of the main fears of the heterarchical architecture of local optimisation that was suggested by Duffie as being one of the possible pitfalls of this architecture and also negated the argument of a lack of system flexibility.

Researchers like Duffie and Hatvany realised that an architecture consisting of weakly coupled cooperating autonomous entities containing local information would promote better fault tolerance and a higher degree of modifiability.[21,24] Duffie stated that the elimination of Master\Slave relationships and Global Information enhances the attainment of implicit fault tolerance and significantly reduces the need for explicitly programmed fault tolerance.[18,21,24]

At the University of Madison-Wisconsin Duffie, Piper, Hartwick and Humphrey developed a flexible manufacturing cell to study hierarchical and heterarchical control strategies.[18] The FMS was identical for both strategies - it was only the control architecture that was modified. The study was set up to compare the differences in performance in a number of areas, the results of this study are shown below in Table 1.5. As can be seen the data from the table shows that the overall performance of the heterarchical system is far better and backs up the comments that have been made.
Much work has been done in the area of analysis of various architecture designs. From the experimental work and research and by gaining results as shown in Table 1.5 Duffie recognised that there were certain features of the architectures studied that led themselves to high levels of performance.

Recently and after extensive research Duffie, Chitturi and Mou presented the following set of design principles relating to a system with a high degree of modifiability and fault tolerance.[18,31]

- Master/Slave relationships should not exist.
- Entities should cooperate with other entities whenever possible.

Aspect	Hierarchical Cell Control	Heterarchical Cell Control
Lines of Source Code	2450	1235
Software Development Cost[1]	$61,250	$30,875
Expansion Software Cost[2]	$960	0
Machine Utilisation[3]	64%	60%
Memory Requirements[4]	50,608	39,2489
Average CPU Utilisation[5]	20%	60%
Complexity	High	Low
Flexibility	High	High
Modifiability	Average	High
Fault Tolerance	Low	High

[1] At $25 per line of Source Code

[2] 24 hours per machine added at $40 per hour

[3] From simulated fault free cell operation

[4] Scheduler bytes plus stack and date segments

[5] Four cell processors, scheduling 50%, control 10%.

Table 1.5: Comparison of Performance between Heterarchical and Hierarachial Architecture.[18]

- Entities should not take it for granted that other entities will cooperate with them.
- Entities should have minimal interaction with each other and where they do the interaction time should be minimised.
- Information generated locally should be retained locally rather than globally but should be available to other entities on request.

Using these rules it was thought that it was possible to develop a system that was highly modifiable, fault tolerant, did not become complicated when extended and was less likely to cause local optimisation. It is clear that the only practical means to achieve the requirements of the above guidelines is to adopt a distributed form of architecture.

1.5 Centralised and Distributed Scheduling Systems

1.5.1 Overview

CIM/FMS systems have as one of their components cells of manufacturing equipment the control of which is implemented by a shop floor controller (SFC) of some description. In the case of the Hierarchical architecture the cells have a centralised factory level controller (FC) that supervises the cells behaviour and gives the appropriate commands and distribute tasks as necessary. In the case of the Heterarchical system the SFC's communicate and negotiate between themselves to acquire and execute tasks. The common denominator of the systems is the need to communicate via Local Area Network Technology. The two different architectures implement two different scheduling paradigms - generally speaking the Heterarchical architecture uses a distributed method and Hierarchical uses centralised.

1.5.2 Centralised Scheduling and Distributed Scheduling: A Definition

Centralised scheduling as shown in Figure 1.9a was the traditional scheduling system and was reflective of the dominant manufacturing architecture at the time (Hierarchical).[31]

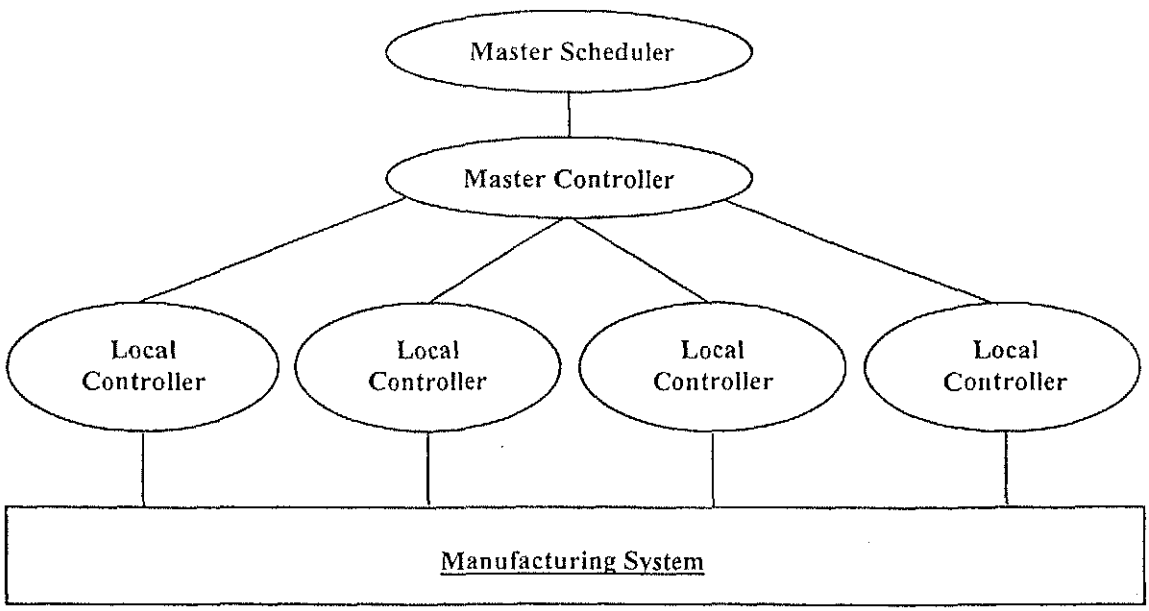


Figure 1.9a - Centralised Scheduling Architecture

As explained by Shaw, Duffie *et.al* and Kung *et.al* the implementation of a centralised scheme revolves around the scheduling of tasks at one location by the implementation of a master scheduler - shop floor

devices have no say in the task allocation process.[25,31-34] The master scheduler maintains a database of knowledge regarding the capability of devices at the shop floor cellular level and based on the devices capabilities assigns tasks to those devices. Whenever the processing status of a machine changes the Master Scheduler is informed - this requires the need for constant updating of the global database which tends to overload the network.[25,48] When tasks are introduced into the manufacturing environment for the first time the Master Scheduler looks at the state of the Manufacturing Environment and based on the state of the environment plans the path that the task will take through the system till completion. Duffie and other authors such as Bakker point out that this type of approach is complicated and may take several hours to execute meaning that by the time the schedule is generated it is obsolete and irrelevant.[31,35] Duffie also states that the simplification process required to make centralised scheduling tractable leads to ineffective or unrealistic schedules. Bakker says that the single attribute that most complicates scheduling is that what occurs in reality has little or no resemblance to that which was planned so that one event change can affect the whole plan.[35] This is the downfall of the centralised system. Kung in his research stated that "even for rather limited FMS, centralised optimal scheduling can be a complex problem defying direct solution".[34] Bakker also made negative comment regarding the centralised system stating that it lacked robustness, required considerable computing resources and is not reactive enough to the real situation.[35] It can be therefore be seen that the Centralised method of scheduling is rather limited in its scope of application.

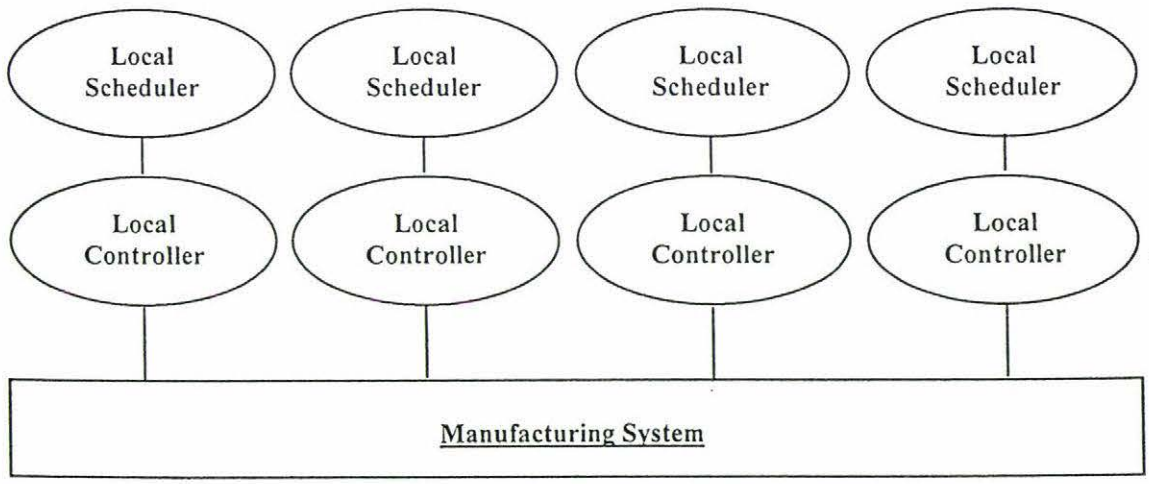


Figure 1.9b - Distributed Manufacturing System Scheduling and Control

Kung *et.al* describes Distributed control as being a means of data management by which "control and data are logically and geographically distributed".[36] The development of the Distributed Scheduling system as shown in Figure 1.9b coincided with the move away from the Hierarchical architectures to architectures such as the Heterarchical type. With the Distributed Scheduling paradigm all active cells have the

opportunity to compete for task assignment via a negotiation process. One form of this negotiation is by the implementation of a task auctioning system such as the Contract Net Protocol.[25,37-43,48] In the Contract Net Protocol approach the Master Scheduler role can be taken by any Local Scheduler - in this role it distributes information regarding the task to be carried out across the network to the cells within the environment. The cells on receiving the information formulate assessments of their current capability to execute the task, and transmit the information to the Local Scheduler for a decision. The Local Scheduler then makes a decision as to which cell receives the task and informs the winning cell.

Features of both Centralised and Distributed Control are summarised in Table 1.6[25,39,48]

	Centralised System	Distributed System
Control Structure	Centralised	Decentralised
Execution of Scheduling	A Master Scheduler	A Local Scheduler in each Cell can take the role as Master Scheduler
Scheduling Control System	Master-Slave control with uni-directional message-passing	Coordination through the exchange of messages between Cells
Vulnerability to failure of Scheduler	Entire system would collapse	Only that particular cell would fail
Manufacturing Database Management	A Global Database	Distributed databases
Maintenance of Dynamic System Information	Constant updating through communication messages	Local updating only required

Table 1.6: Features of Distributed and Centralised Scheduling Systems.

1.5.3 Centralised and Distributed Scheduling: Performance Evaluation

In 1986 Shaw set up a simulation to evaluate the performance of a Centralised Myopic - SPT (Shortest Processing Time) strategy with a Distributed negotiation-based approach using SPT as the basis - the results are presented in Table 1.7[25,32,40,48] The results are a culmination of twelve different simulation runs

based on using statistically generated input data, and on ten occasions the Distributed approach clearly out performed the Centralised paradigm. In his assessment of the results, Shaw stated that the Distributed scheduling method outperformed the Centralised system because of the fact that the Centralised system relied on constant database updating which overloads the network.[4,25,32,40,48] He also noted that the Distributed system scheduled more efficiently because the assessment process was based on more accurate up to date information due to the finite loading able to be placed on the network. Another point made by both Shaw and Smith is that because the Distributed scheduling system is carried out in parallel by a number of nodes on the network computational time is decreased.[4,25,39-40,48]

Shaw pointed out that other advantages that the distributed system had over the centralised system included:[39-40,48]

- Better Reliability because of the graceful degradation of the scheduler in the event of failure.
- Upward Extensibility - the control structure remains the same when new cells are added with minimal increase in network traffic.
- Scheduling is performed in parallel therefore the bottleneck produced by the global scheduler is eliminated.
- Cost Effectiveness -processing requirements are less therefore no necessity exists to buy more expensive computer equipment.

Smith also pointed out these potential advantages in pioneering work undertaken in the late 70's in the area of distributed scheduling using the auctioning system stating that distributed problem solving such as scheduling "offers the promise of speed, reliability and extensibility", Shaws simulation work showed these theories to be correct.[4]

Bakker carried out some as regards the creation of a new control structure based on the Distributed Control Structure and gave it the name DFMS (Distributed FMS).[35] In his work he identified the a number of advantages of the Distributed System including improved system reliability, better extensibility, conceptual simplicity and that it was more reactive to changes in the overall system schedule.

Zhang *et.al* made the point in a Robotics and Manufacturing Symposium in British Columbia in 1990 that another positive aspect with distributed control was that it had "the ability to handle applications with a natural spatial distribution".[36]

Williams pointed out advantages with Distributed system control when he stated in a paper in 1993 that they have "initiated a new era of ease of specification and installation".

In 1995 Ginting *et.al* developed a scheduling simulator as part of the research being undertaken at Massey University's Production Technology Department, the findings of trials undertaken during the research also supported the work carried out by Shaw and Smith and results were indicative of those in Table 1.7[44]

Bidding-SPT Results												
% of Jobs Late	15.85	21.80	32.56	22.05	23.25	28.78	41.20	23.75	43.28	47.43	67.34	39.55
Avg. waiting time	9.53	9.99	11.56	9.98	9.26	9.29	10.34	8.96	8.60	8.15	8.98	7.96
Avg. Lateness	4.90	5.23	6.38	5.24	2.32	2.54	3.41	2.78	1.42	1.44	1.91	1.31
Mean Flow Time	28.13	28.30	30.67	28.60	20.33	21.07	22.70	20.68	15.53	15.51	16.79	15.29

Myopic-SPT Results												
% of Jobs Late	24.11	29.89	37.27	27.28	25.56	29.15	43.46	26.54	44.13	46.95	66.83	39.93
Avg. waiting time	10.22	10.62	11.99	10.52	9.37	9.28	10.92	9.12	8.55	8.20	9.09	7.99
Avg. Lateness	6.18	6.35	7.17	6.89	2.92	3.47	5.00	3.65	1.65	1.87	2.32	1.53
Mean Flow Time	27.81	28.69	30.94	28.85	20.35	21.80	23.28	20.79	15.48	15.50	16.92	15.25

Table 1.7: Results of Simulation based Assessment of a Distributed and Centralised Scheduling Mechanism.

1.6 Contract Net Protocol - A Distributed Problem Solver

1.6.1 Introduction

The Contract Net Protocol is a distributed scheduling system used for scheduling everything from task scheduling in manufacturing companies[25], planning of freight transport[45] to the timely control of traffic light operation in the area of transport logistics. This very fact in itself shows the flexibility of use that the negotiation protocol has in that it can be applied in many varied instances. Smith *et.al* points out that the

framework is based on the human model of experts that cooperate to solve problems by the transfer of messages and use a contract negotiation process to distribute tasks to be executed concurrently.[41]

Shaw and others have already mentioned the benefits of the implementation of a distributed approach to FMS system control. The next step involves the choice of the model for such a system. The Contract Net Protocol and other negotiation type control systems have been used to achieve good system performance measures and have other positive attributes.[25,37-43,48]

Many authors such as Harris *et.al* and Smith *et.al* have shown that the bidding mechanism employed in the Contract Net Protocol can be used as an efficient way of allocating work and tasks within an organisation.[46-47] Cells in the networked environment use the negotiation protocol as a collective decisionmaking tool to resolve situations where there are a number of cells that are able to carry out a task they have in their possession.

When a cell (Manager Cell) has a job that it has worked on already and needs to allocate the next operation in that job (called a 'task') to a cell it initiates the negotiation based task assignment protocol by broadcasting a task announcement message over the network to other cells. Those cells that receive the message will in turn submit a price (called a 'bid') for carrying out the job - such a price maybe the earliest time that the cell can commence processing the task in the case that the Earliest Finishing Time heuristic is being used. If the cell is unable to carry out the task it does not submit a bid. When all bids have been received the manager cell then selects the cell with the cheapest price and the corresponding workpiece is then transported to the cell concerned. Shaw[25,32,48] constructed the information flow model for the which illustrates graphically the bidding process as shown in Figure 1.10.

If a manager cell possesses knowledge about other cells in the environment it can issue a *limited broadcast* to those cells and then the bid assessment by the Manager Cell can be based solely around responses from those target cells - the argument for this approach is that less overall computer time is tied up in the assessment process. Simulation work carried out by Tsukada *et.al* has shown that when the negotiation protocol is used to redistribute the work on a process which has broken down that the effect on the rest of the system is negligible in terms of the number of cells disrupted, the number of reschedulings required and the average makespan of tasks as compared to other recovery algorithms used.[49]

Shaw[25,32-39] defines the following features that are typical of systems that employ a network wide bidding protocol:

- Distributed control is used and no cell has greater importance, as far as control is concerned, than any other cell.

- The information used in the schedule is flexible and can take into account loading of processes, breakdowns and resource constraints in the bidding scheme.
- The system of bidding creates better schedules than traditional methods as they are created from real time data.
- Information processing requirements are mostly executed with the cells thus taking the load off the network.

There are a number of variations on the negotiation protocol theme, some of these are:

- Most negotiation protocols discussed in general literature assume that the system is implemented on a pure heterarchical structure but some authors such as Rabelo *et.al* have proposed implementation of the negotiation protocol on a semi-hierarchical platform.[25]

- In Shaw's system when a manager sends out a task announcement a given time period elapses before it assesses the bids received, if a cell has not prepared a bid in the meantime it misses out.[25]

- Task announcements can be made as general broadcasts (to all cells), as limited broadcasts (to a selection of cells) or as targeted broadcasts (to a cell).

- Tsukada believes that the best approach to scheduling is to preplan the schedule before any work is done and to only use the negotiation protocol as a task redistribution tool. He believes that the negotiation protocol is myopic and unpredictable and unproven in its ability to produce optimised solutions.[49]

- Bakker in his work suggests the use of the EFT (Earliest Finishing Time) heuristic as the best scheduling strategy as it has been shown to be the best performer compared to other heuristics such as the SPT (Shortest Processing Time) approach.[35]

- Fischer *et.al* in his work with truck scheduling heuristics has used the negotiation protocol in the scheduling of trucks. The trucks contain their own localised schedules and the auctioning of delivery orders occurs between the trucks - his work has shown that in the event of a truck breaking down the system allows for the next most appropriate truck to be selected without major global replanning being required.[45]

1.6.2 Advantages of the Negotiation Approach

Shaw has carried out extensive research in the area of negotiation protocols and has carried out much simulation work to verify his preconceptions about system features and performance. In work carried out by Shaw, Bakker and Rabelo *et.al* the following positive aspects of the negotiation protocol have been identified.[25,35,38-39]

- ***Minimal Network Traffic Needed*** - because the system is comprised of loosely coupled cells and communication is based on simple message passing minimal load is placed on the Network. If network capacity is at a premium the bidding system can be further improved using limited broadcasting.

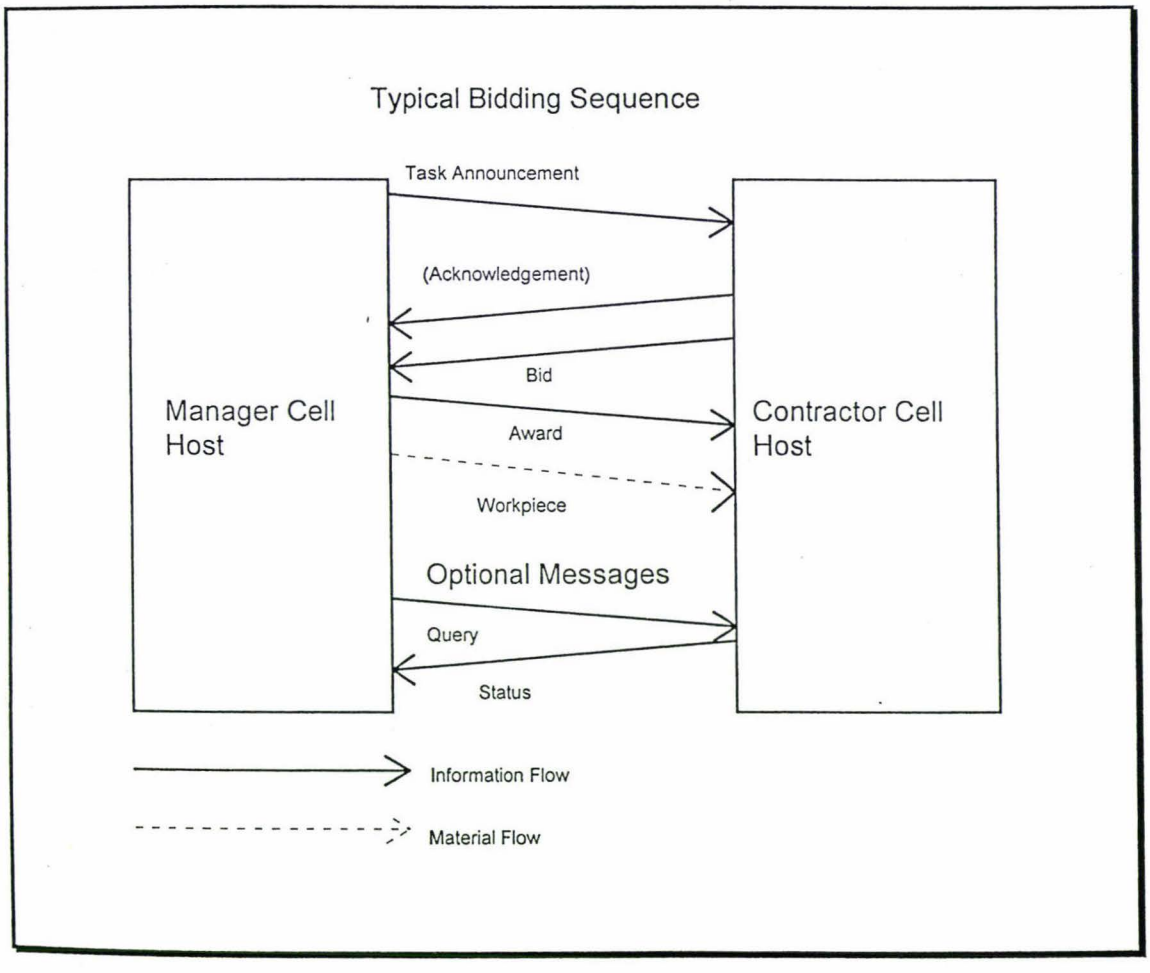


Figure 1.10 - Information Flow Model of the Bidding Process

- ***Implements Real Time Control*** - because the bid reflects dynamic status information near real time control can be achieved.

- ***Bottleneck Elimination*** - because data processing is carried out in parallel by all cells in the system the bottleneck at the centralised processor is eliminated.

- ***Flexibility*** - the bidding mechanism can be designed to reflect a wide variety of dynamic status information and desired levels of performance.
- ***High Reliability*** - Global databases are eliminated and bidder response is optional, therefore the auctioneer does not crash when a bidder does not respond to the auctioneer.

- ***High Extensibility*** - due to the highly localised data processing aspect and modular design it is easy to extend the system.

Simulation work carried out by Fischer found that a form of the negotiation protocol used for truck transportation scheduling gave levels of performance that were comparable to those of heuristic search OR (Operations Research) algorithms. He also discovered that his negotiation protocol had substantial advantages over those algorithms that agree with those findings of Shaw and Bakker such as the potential for reactibility to changes in system schedule, flexibility improvements and the ability to schedule in real time.[45]

Bakker however noted that there are a limited number of disadvantages but that the magnitude of these disadvantages outweighed the positive aspects.[35]

- *Slightly Limited Product Variety* - due to the finite numbers of different tools that can be used Bakker suggested that product variety is limited, but comments that "the impact of these limitations is small because most companies produce a limited set of products".

- *Limited Optimisation Possibilities* - Bakker suggests that the over simplification required using the negotiation protocol will not truly reflect true workplace dynamics.

1.6.3 Discussion

Auction based algorithms such as the Contract Net Protocol are very useful for the scheduling of tasks in that they provide a mechanism for real-time scheduling, are highly agile, easily extendable, and are highly adaptable to system disturbance. Shaw proved in his simulation work along with Ginting *et.al* that a properly designed job bidding system outperforms the centralised myopic approach in the areas such as job waiting time, job lateness and mean flow time[25,32,40,44,48] - this factor coupled with the other positive factors already mentioned provide us with an excellent alternative to traditional scheduling approaches which have proven to be very fragile.

1.7 Summary

This literature review has been presented to give an introduction into the latest theories regarding the modern FMS system so an appreciation of where this previous work fits in can be gained. The model of the Hybrid FMS presented in the next Chapter of this thesis will show how many of these paradigms are implemented. The literature review was written after carefull reading of up to 200 different papers relevant to modern FMS and CIM systems. As such a comprehensive and complete study has been undertaken to ensure an appropriate model of the proposed system can be created.