DFMS: Architecture and implementation of a distributed control system for FMS

J.J.A. Bakker

DFMS: Architecture and implementation of a distributed control system for FMS

Proefschrift ter verkrijging van de graad van doctor aan de Technische Universiteit Delft, op gezag van de Rector Magnificus, Prof. drs. P.A. Schenck, in het openbaar te verdedigen ten overstaan van een commissie door het College van Dekanen daartoe aangewezen, op 20 juni 1989 te 14.00 uur door:

Johannes Josephus Antonius Bakker,
geboren te Amsterdam,
werktuigkundig ingenieur.

Dit proefschrift is goedgekeurd door de promotor
prof. ir. L.N. Reijers

Abstract

The *Distributed Flexible Manufacturing System* (DFMS) is a system that can be used to control a range of flexible production systems such as job shops, DNC systems and traditional FMS. The control system is logically and physically distributed over a number of components that cooperate to perform the control functionality.

Each machine in the systen is equipped with a special component called a *Station Manager*. Station Managers have a dual functionality: they determine the sequence of operations on their machine and they guarantee that the resources, required to process the operations, are available.

To have resources available, Station Managers request services from *Function Modules*. The DFMS architecture provides Function Modules that are responsible for tool management, for pallet transports, for program management and for the introducing of new products into the system. The relation of Station Managers and Function Modules is as the relation between clients and servers in a client-server model.

An important characteristic of the DFMS architecture is that no schedule is made before the actual production starts. When an operation on a product is completed, the Station Managers confer with each other to determine which one is most suited to process the next operation on the product; a distributed dispatching method is applied.

The Station Manager functionality can easily be implemented on *programmable CNCs*. To implement a complete FMS control system a number of programmable CNCs, some small computer systems and a Local Area Network are required. These simple requirements make the application of DFMS possible in areas where previously flexible automation was considered too complicated and too expensive.

Some advantages of the DFMS system are: conceptual simplicity, high robustness, high reliability, high extensibility, broad application and low cost.

Acknowledgements

This research has been carried out in the Laboraratory of Manufacturing Systems of the Delft University of Technology. The DFMS project would not have been possible without the moral and actual support of the leader of the laboratory: Prof. Ir. L.N. Reijers. The author wishes to thank staff and students of the laboratory who contributed directly or indirectly. Many colleagues at Philips CFT and Philips I&E are thanked for their advice and help during the project. Special thanks are reserved for Joop Kaashoek who has contributed greatly during the specification and implementation phases of the DFMS prototype. Marcel Zeestraten is thanked for the many discussions we had concerning the basics of FMS and the required functionality of FMS controllers. The help of Jan Neve with the proofreading of the manuscript is thankfully acknowledged.

Samenvatting

DFMS: Architectuur en implementatie van een gedistribueerde besturing voor FFS

Flexibele Fabrikage Systemen (afgekort FFS) worden toegepast voor de produktie van *families* van produkten in kleine series. Produkten worden tot een familie gerekend, omdat zij bepaalde eigenschappen gemeen hebben die maken dat zij met soortgelijke produktiemiddelen vervaardigd kunnen worden.

De produkten ondergaan een aantal bewerkingen op de machines in het systeem. De volgorde en aard van de bewerkingen veschilt van produkt tot produkt. Bij het vervaardigen van produkten zijn *hulpmiddelen* noodzakelijk. *Pallets* worden gebruikt om produkten te transporteren, *stelstukken* worden gebruikt om produkten op pallets vast te spannen en *gereedschap* wordt gebruikt om de produkten vorm te geven. Pallets en stelstukken zijn hulpmiddelen die in principe onbeperkt bruikbaar zijn, gereedschap is in min of meerdere mate aan slijtage onderhevig.

Om de activiteiten in een FFS te besturen wordt een *systeembesturing* gebruikt. De systeembesturing zorgt er voor dat produkten in de juiste volgorde worden bewerkt op de diverse machines en dat de benodigde hulpmiddelen op tijd bij de machines aanwezig zijn.

De FFS besturing kan de efficiency waarmee het systeem produkten aflevert beinvloeden, door de volgorde van bewerkingen in het systeem te kiezen. Een goede systeembesturing zorgt er voor dat de eindprodukten beschikbaar zijn voordat zij nodig zijn en dat de hulpmiddelen zo efficient mogelijk worden gebruikt.

Efficiency is niet het enige criterium om een FFS besturing op te beoordelen. Andere belangrijke eigenschappen zijn onder meer de conceptuele eenvoud, de robuustheid, de uitbreidbaarheid en de kosten van de besturing.

De *Distributed Flexible Manufacturing System* (DFMS) architectuur is ontwikkeld om een FFS besturing te verkrijgen die geschikt zou moeten zijn om FFS technieken toegankelijk te maken voor een veel bredere markt dan gebruikelijk.

De DFMS besturing bestaat uit een aantal afzonderlijke componenten die met elkaar samenwerken om de standaard besturingstaak te verrichten. In de DFMS architectuur zijn twee groepen componenten gedefinieerd: *Station Managers* en *Function Modules*[1]. Station Managers en Function Modules werken samen volgens het *client-server model*: de Station Managers fungeren als clients, de Function Modules fungeren als server.

Iedere machine in het systeem is uitgerust met een Station Manager. De Station Managers overleggen met elkaar om te bepalen wie het meest geschikt is om een bepaalde bewerking te verrichten. Wanneer een bewerking is toegekend aan een Station Manager, is die Station Manager er voor verantwoordelijk dat de bewerking wordt uitgevoerd.

Bij het treffen van voorbereidingen voor de bewerking kan de Station Manager de hulp inroepen van Funtion Modules. De volgende Function Modules zijn gedefinieerd:

- Een *Program Module* is verantwoordelijk voor het bewaren van alle NC programma's. Wanneer een bewerking moet worden uitgevoerd op een machine waarvoor het programma nog niet aanwezig is, zal de Station Manager de Program Module verzoeken het programma beschikbaar te stellen.

- Een *Tool Module* is verantwoordelijk voor het toolmanagement in het systeem. De Tool Module beheert het centraal gereedschapmagazijn en kan gereedschap reserveren ten behoeve van Station Managers.

- Een *Pallet Transport Module* wordt gebruikt voor het uitvoeren van pallettransporten. Station Managers verzoeken de Pallet Transport Module pallets te halen of weg te brengen.

- Een *Load Module* is verantwoordelijk voor het introduceren van produkten in het systeem.

- Een *Operator Module* wordt gebruikt om de systeemoperator inzicht te geven in de werking van het systeem.

[1] De Engelstalige DFMS terminologie wordt aangehouden in deze samenvatting.

De Function Modules leveren *services* die door de *clients*, de Station Managers, worden aangevraagd.

Een belangrijk aspect van de DFMS architectuur is dat geen *schedule* wordt gemaakt voordat de eigenlijke produktie start. Normaal gesproken wordt een schedule gemaakt om vast te leggen welke bewerkingen op iedere machine verricht zullen worden. In DFMS wordt *tijdens* de produktie bepaald welke machine een bewerking zal verrichten. Wanneer een produkt een bewerking heeft ondergaan op een machine, bepaalt de Station Manager van die machine of een vervolgbewerking nodig is. De Station Manager inspecteert daartoe het *Product Script*, waarin per produkt de benodigde bewerkingen beschreven staan. Wanneer een vervolgbewerking nodig is, verzoekt de Station Manager zijn collega Station Managers die in staat zijn de bewerking te verrichten, een *offerte* uit te brengen. De Station Manager die de beste offerte uitbrengt, krijgt de bewerking toegewezen; de gekozen Station Manager voegt dan de bewerking toe aan zijn *Operation Queue*. De Operation Queue bevat de bewerkingen die door de Station Manager zijn geaccepteerd en die door de machine zullen worden uitgevoerd.

Bewerkingen die in het systeem kunnen worden uitgevoerd zijn ingedeeld in *Operation Classes*. Iedere Station Manager kan een of meerdere Operation Classes ondersteunen. Iedere Station Manager heeft een volledig overzicht over welke Operation Classes ondersteund worden door alle andere Station Managers in het systeem. Bij het bepalen welke Station Manager een bewerking uit zal gaan voeren worden alleen die Station Managers, die de gevraagde Operation Class ondersteunen, gevraagd een offerte uit te brengen.

De DFMS architectuur leent zich door zijn gedistribueerd karakter ervoor om gedistribueerd geimplementeerd te worden. Ten behoeve van de gedistribueerde implementatie kan goed gebruik worden gemaakt van een *programmeerbare* CNC. Een programmeerbare CNC is een CNC die is uitgebreid met de mogelijkheid om MS-DOS programma's te draaien. Om zinvolle interactie tussen het MS-DOS gedeelte en de traditionele CNC functionaliteit mogelijk te maken, wordt gebruik gemaakt van een uitgebreide DNC interface. De Station Manager functionaliteit kan nu eenvoudig, zonder dat speciale apparatuur nodig is, geimplementeerd worden op de CNC.

Wanneer gebruik wordt gemaakt van programmeerbare CNC's, kan een DFMS besturing systeem worden gerealiseerd door met behulp van een Local Area Network de programmeerbare CNC's met elkaar en met een aantal

Function Modules te verbinden. De Function Modules kunnen worden geimplementeerd op eenvoudige IBM-PC compatible computer systemen.

De DFMS architectuur kan in een aantal versies worden toegepast op een aantal terreinen, van simpel tot gecompliceerd:

- de DFMS-J versie als besturing voor een eenvoudige job shop;
- de DFMS-D versie als DNC systeem;
- de DFMS-F verie als besturing voor een volledig geautomatiseerd Flexibel fabrikage Systeem.

Dezelfde basis architectuur kan met slechts geringe wijzigingen worden aangewend voor de drie genoemde toepassingen. Het is daarbij eenvoudig om van de ene toepassing door te groeien naar een gecompliceerde toepassing.

Toepassing van de DFMS architectuur heeft een aantal belangrijke voordelen:

- De *conceptuele eenvoud* van de architectuur maakt dat voor de operators de handelingen in het systeem goed begrijpbaar zijn.
- De DFMS architectuur is inherent *robuust.* De besturings componenten zijn van elkaar gescheiden en kunnen elkaar slechts beinvloeden volgens nauwgezette protocollen.
- In DFMS worden geen dispatching beslissingen genomen die gebaseerd zijn op de beschikbaarheid van hulpmiddelen in de toekomst. In DFMS wordt het toekennen van een bewerking pas gedaan op het moment dat de bewerking uitgevoerd kan worden. De bewerking wordt gealloceerd aan een Station Manager die gegarandeerd geschikt is om de bewerking te verrichten.
- De DFMS architectuur kan op een breed terrein worden ingezet. Met één architectuur kan zowel de job shop, de DNC als de FFS applicatie worden bestreken.
- Door het modulaire karakter van de DFMS architectuur en implementatie is het eenvoudig mogelijk om een FFS besturing uit te breiden in kleine stapjes. Met DFMS is het mogelijk om precies zoveel besturingscapaciteit te installeren als nodig is.

- Door de eenvoud van het DFMS concept en door het toepassen van eenvoudige technische middelen is het mogelijk tegen lage kosten een DFMS besturing te realiseren.

- De DFMS architectuur maakt het door zijn eenvoud, lage kosten en uitbreidbaarheid mogelijk, om FFS-achtige techniek te introduceren in toepassingen waar voorheen FFS te gecompliceerd en te duur werd geacht.

Contents

List of DFMS definitions

Most definitions in this list are used exclusively in DFMS context. Some definitions have a more general application, but are explained because they are frequently used in this dissertation.

Clamp Station
A physical component in an FMS where products are clamped on pallets or are removed from pallets.

Computer Numerical Control
The controller of a machine. The CNC is responsible for performing the operation on a product.

Dispatching
The process of giving orders to machines and other components in an FMS.

Emergency Tool Delivery Request
A DFMS message that is sent by a Station Manager to the Tool Module when a tool has broken and urgently needs to be replaced.

Fixture
A mechanical device necessary to clamp products on a pallet. Fixtures are product specific.

Function Module
The group name of general modules in the DFMS architecture that perform services.

Load Module
The DFMS Function Module that is responsible for introducing products into the system.

Load Startup Information
A DFMS message that is sent by Station Managers to the Load Module. The message contains information about pallets in the local pallet buffer of the machine.

Machine Capacity Allocation
A DFMS message that is sent between Station Managers. The message is sent to the Station Manager that issued the best quotation and now is requested to perform the operation.

Machine Capacity Offer
A DFMS message that is sent between Station Managers. The message is a reply on a Machine Capacity Request and contains a quotation of the Station Manager that is requested to perform an operation.

Machine Capacity Request
A DFMS message that is sent between Station Managers. The message is a request to perform an operation.

Machine Down
A DFMS message that is sent to the Operator Module by a DFMS component that is to be removed from the system.

Machine Up
A DFMS message that is sent to the Operator Module by a new DFMS component. The message contains the Operation Classes that the new component supports.

Network Update
A Network message that contains the full information about network addresses and DFMS destinations. The message is sent by the Network Directory Server to the Network Layers of all DFMS components.

New Machine
A Network message that contains the network address and DFMS destination of a new DFMS component. The message is sent by the new component to the Network Directory Server.

Network Address
The network identification of a component in a DFMS Network.

Network Directory Server
The Network Module that is responsible for maintaining the Network Address information in the DFMS network.

Network Interface
The software layer that is responsible for sending and receiving messages between DFMS components.

Operation Class
A group of operations that can be performed with a common tool set on a machine.

Operator Module
The DFMS Function Module that allows operators to interact with the DFMS components.

Operation Queue
The structure that is maintained by each Station Managers that holds the operations that will be performed by the machine.

Operation Queue Request
A DFMS message that is sent by the Load Module to the Station Managers. The message is a request to upload the contents of the information in the Operation Queue.

Operation Queue Reply
A DFMS message that is sent by Station Managers to the Load Module that contains a summary of the Operation Queue.

Operation Reject
A DFMS message that is sent by the Station manager to the Operator Module when an operation cannot be performed by the machine.

Pallet Id
The name that is assigned to a pallet. The pallet is uniquely defined by its *Pallet Id*.

Pallet Startup Information
A DFMS message that is sent by the Station Managers (and Load Module) to the Pallet Module. The message contains information about the pallets in the local pallet buffer.

Pallet Transport Module
The DFMS Function Module that is responsible for transporting products between machines in a DFMS application.

Pallet Transport Request
A DFMS message from a Station Manager (or Load Module) to the Pallet Transport Module in which the Pallet Transport Module is requested to deliver or store a pallet.

Pallet Transport Error
A DFMS message sent by the Pallet Transport Module to a Station Manager (or Load Module) in which the DFMS component that requested a pallet transport is informed about the impossibility to perform the requested transport.

Product Script
A data structure that holds information on a product. In the Product Script the operations that need to be performed on the product are specified.

Program Module
The DFMS Function Module that maintains part programs that are required to perform the operations on the machines.

Programmable CNC
A hardware component that is used to implement both the traditional CNC functionality and the Station Manager functionality.

Program Delivery Reply
A DFMS message that is a reply on a Program Delivery Request. The message contains the program to be downloaded.

Program Delivery Request
A DFMS message from a Station Manager to the Program Module in which the downloading of a programs is requested.

Station Adapter
A hardware component that is used to implement the Station Manager functionality. It is used in combination with a traditional CNC.

Station Manager
The DFMS Function Module that is responsible for all operations on a machine.

Scheduling
The process of assigning operations to machines and operators before the actual production starts.

Tool Class
The name that is assigned to a *type* of tool.

Tool Id
The name that is assigned to a *specific* tool. The tool is uniquely defined by its Tool Id.

Tool Module
The DFMS Function Module that is responsible for the tool management in DFMS.

Tool Delivery Reply
A DFMS message that is the reply of the Tool Module to a Tool Delivery Request. The message specifies whether the requested tools can be delivered or not.

Tool Delivery Request
A DFMS message in which a Station Manager requests the Tool Module to deliver the tools that have been resevered for a particular order.

Tool Reservation Reply
A DFMS message that is the reply of the Tool Module to a Tool Reservation Request. The message specifies whether the requested tools can be reserved or not.

Tool Reservation Request
A DFMS message in which a Station Manager requests the Tool Module to reserve some tools.

Tool Startup Information
A DFMS message that is sent by a Station Manager to the Tool Module. The message contains information about the tools that are in the tool store of the machine.

Updated Machine Capabilities
A DFMS message that is sent by the Operator Module to all DFMS components. The message contains the Operation Classes that are supported by all DFMS components.

World Model
A list in which is specified which Operation Classes are supported by all DFMS components.

Chapter 1

Overview of FMS

1.1 Definition of FMS

1.1.1 The principle of FMS

Flexible Manufacturing Systems (FMS) are production systems used to manufacture a set of related products in small series. The products that are processed undergo one or more operations. The sequence of operations and the operations themselves vary for different products. The products can be grouped into one or more product-families. A product family is a set of products that have similar production characteristics. Examples of important characteristics are shape, size, material and required operations. An FMS will be able to produce only products that are members of the product family for which the system was designed.

The operations in the FMS are performed on general purpose machines that are able to perform a wide variety of operations. A large percentage of the machines used in FMS perform metal removing operations. Typical machines are machining centers to perform milling, drilling and boring operations, and lathes or turning centers to produce rotational workpieces. Apart from metal cutting machines, special machinery to wash or measure products is frequently used.

1.1.2 Systems, Cells and Modules

A very diverse terminology is used to describe systems that have FMS properties. A definition that first has been introduced by Yamazaki [Lacey 86] is presented here in a slightly modified form:

- **Flexible Manufacturing Module (FMM)**
  An FMM is an automated machine that is equipped to produce unattended for a prolonged period of time. The machine is able to change tools and products automatically. Simple process control equipment is used to detect problems such as tool failure or tool wear. In figure 1.1 a schematic overview of an FMM is shown;

- **Flexible Manufacturing Cell (FMC)**
  An FMC consists generally of two or three machines. The machines are linked: operations can only be performed in a *fixed* sequence. The products can be automatically transferred between the machines. In figure 1.2 a schematic overview of an FMC is shown;

- **Flexible Manufacturing System (FMS)**
  An FMS consists of a number of machines controlled by a computer system that is used to determine the sequence of operations. Products can be *routed freely* through the system. The products undergo operations on different machines. Automated systems are used for the transport of products and tools. Advanced process control methods are applied to control the operations on the machines. In figure 1.3 a schematic overview of an FMS is shown;

- **Flexible Manufacturing Factory (FMF)**
  An FMF is a completely automated production facility where products are processed from raw material to finished product. An elaborate computer system controls not only the technical system, but is also responsible for the financial and organizational aspects of production. A term that is used frequently for these kind of systems is *Computer Integrated Manufacturing*: CIM.

In the literature a large variety of definitions is used. Manufacturers of flexible equipment tend to over-classify their products. Single machines are sometimes called Flexible Manufacturing Cells or even Flexible Manufacturing Systems.

1.2 Productivity and flexibility

FMS systems allow for efficient production in small batches. In this respect they fill the gap between manual production of one-off products and highly automated mass production systems. In mass production highly specialized

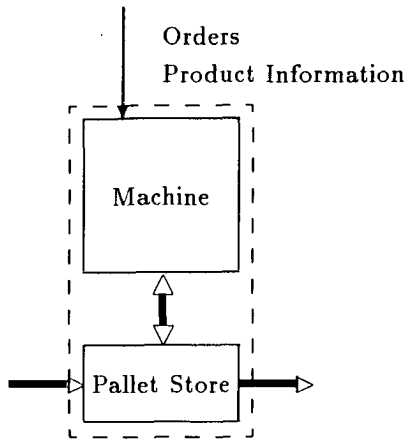


Figure 1.1: A schematic overview of a Flexible Manufacturing Module.

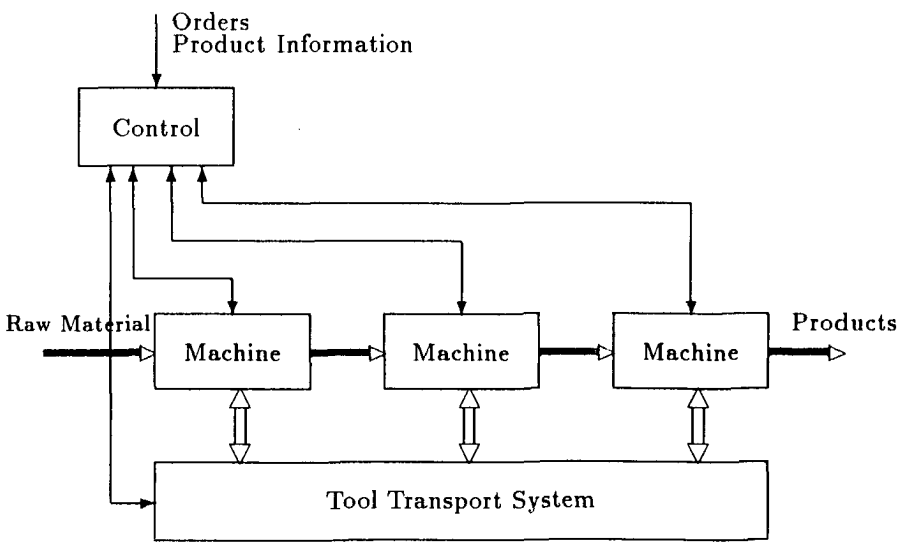


Figure 1.2: A schematic overview of a Flexible Manufacturing Cell.

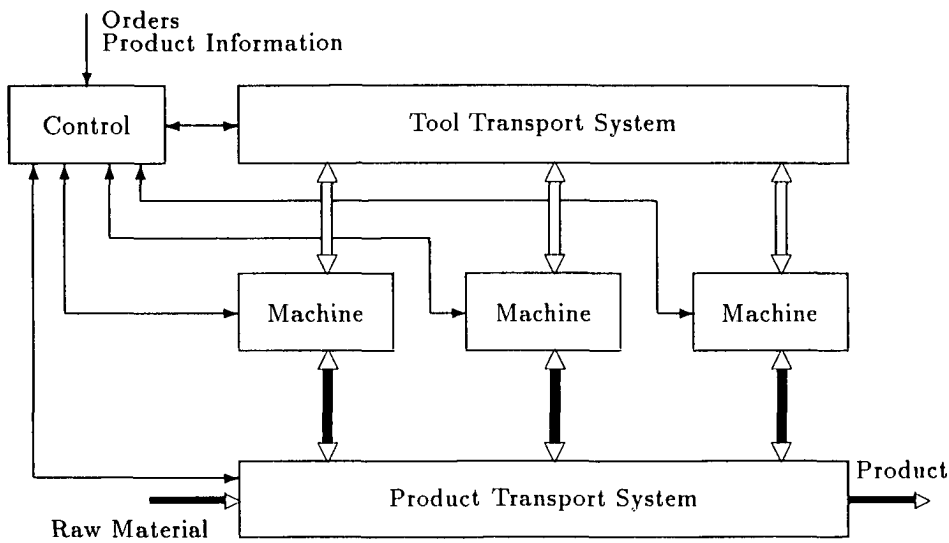


Figure 1.3: A schematic overview of a Flexible Manufacturing System.

machinery can be used to produce large numbers of similar products. For a smaller number of products the investment in the specialized, dedicated machinery is not feasible. Mass production systems are highly productive but very inflexible.

When single products have to be made, the use of automated systems will normally not be economically justifiable. Few automated system have the flexibility required to produce a very large number of different products. Therefore, for manufacturing of one-off products, manual production methods are mostly used. Manual production methods are very flexible, but not very productive.

For larger numbers of products the low flexibility of manual production methods is a serious problem. In general FMS will be applied when:

- a limited number of *related* products has to be made;
- the number of products to be made does not allow for mass production methods;
- the number of products to be made is too large to make the use of manual production methods acceptable.

Figure 1.4 shows the application of FMS in terms of productivity and flexibility. The term 'flexibility' has several meanings. A system is not just 'flexible', it is flexible with respect to a certain property. A number of flexibility properties are:

- **Product Family Flexibility**
  The product family flexibility expresses the ease with which the system is able to produce different products that belong to the product family for which the system was originally designed. This type of flexibility is crucial to Flexible Manufacturing Systems;
- **New-Product Flexibility**
  The system is said to have a high new-product flexibility, when it is relatively easy to adapt the system to produce new products that do not belong to the product family for which the system was originally designed;
- **Product Life Cycle Flexibility**
  Due to the product life cycle, the demand for the product is not constant over its life time, but varies. A system has a high product life

cycle flexibility if it is able to meet the varying demand due to changing market requirements;

- **Volume Flexibility**
  The volume flexibility determines whether it is simple to vary the total volume of products manufactured in the system;

- **Routing Flexibility**
  The routing flexibility determines whether it is necessary to perform the subsequent operations on a product on specific machines or that for each operation a number of machines can be chosen. A system has a low routing flexibility when each operation has to be performed by a specific machine;

- **Personnel Flexibility**
  The ease with which the system can be run with a varying number of operators determines the personnel flexibility. A system has for example a high personnel flexibility when it is possible to work at night with less operators than during the day.

It is important to understand that flexibility in isolation is not a very valuable property. It is not difficult to make a highly flexible system, in fact, the most flexible system that can ever be designed is a human operator with a universal machine. However, it is extremely difficult to design a system that is both flexible and productive. *For flexibility a price has* always *to be paid.* A well designed production system therefore should be just as flexible as is absolutely required. In general, all excess flexibility can be considered to be wasteful. In the next section some requirements for successful application of FMS will be discussed.

1.3 Requirements of FMS

FMS is an example of a production system in which the process has been automated. Automation of an industrial process is only possible when the process has been adapted properly: rationalization is required. In general this means that all aspects of the system to be automated have to be analyzed to evaluate their impact on the behavior of the automated system. Examples of important aspects to consider for FMS are:

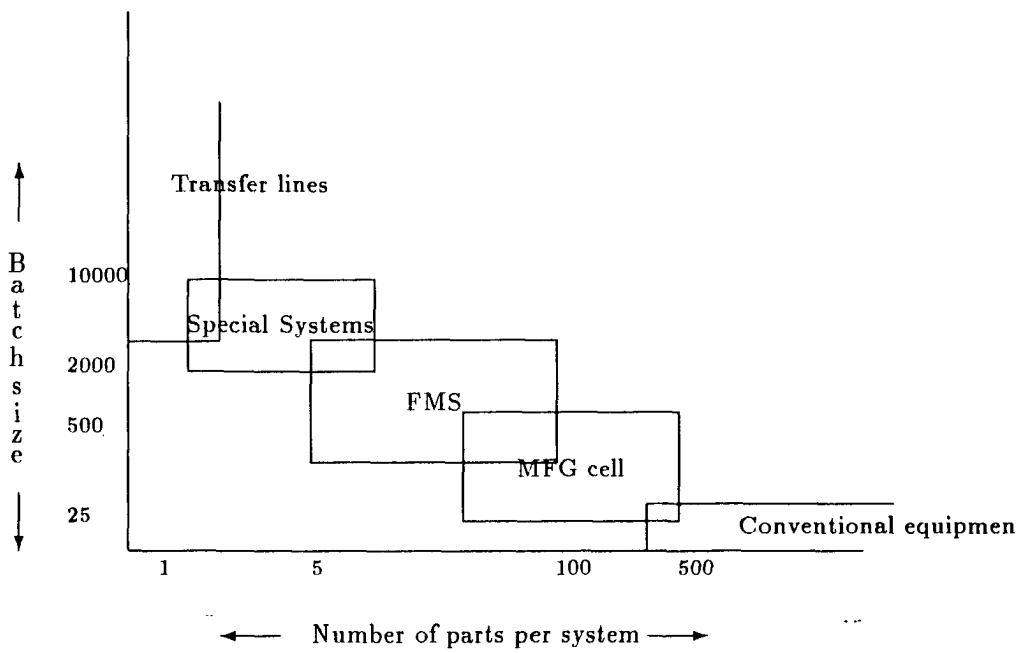


Figure 1.4: The relation between number of parts and productive capacity.
Source: Recent trends in Flexible manufacturing, United Nations 1986

- **The choice of products**
  Not all products are equally suited for automated production. Products that are known to cause many problems should be avoided. Material, shape and required operations determine the suitability of products;

- **The design of products**
  Elimination of unnecessary variety in product design in a product family lowers the number of tools needed to manufacture the family. This is an important advantage, because a very large set of required tools may render automated operation impossible;

- **The product planning**
  Elimination of unnecessary variety of product planning in a product family may lower the number of tools needed to manufacture the family;

- **The NC part program development**
  Unmanned production requires a high reliability, so NC part programs will have to be developed to maximize reliability, even at the cost of reduced efficiency.

1.4 Advantages of FMS

The necessity to manufacture small series has always been apparent, but has recently grown. The growing interest in flexible manufacturing systems can be explained by analyzing developments in the market:

- Consumers demand products in a larger number of varieties;
- The commercial life time of products decreases;
- Competition is becoming more vigorous;
- Quality requirements are increasing.

As a result of these developments manufacturers wish to be able to react without delay to changing demands and to be able to efficiently produce small series of products without having to invest in product specific production equipment. Flexible manufacturing systems seem to be well suited to this situation. There are two aspects of FMS that explain its suitability:

- the high level of integration;

- the high level of automation.

The integration referred to, is the integration between the control system and the system to be controlled. All important functions in FMS are connected: product design, product planning, process control, scheduling, tool management and system maintenance are all part of one system. Advantages of the high level of integration include:

- **Short lead times**
  The lead time of products in a production system is frequently not determined by the operation times but by the total time the products have to wait for operations. By applying advanced scheduling methods this wait is minimized;

- **Minimal work in process**
  The disadvantage of many products waiting to be processed is not only the resulting high average product lead time, but also the necessity to buffer all these products. A large number of products usually represents a considerable economic value. In addition, the technical requirements to store pallets in FMS systems, pallet buffers, pallets and fixtures, are usually very expensive;

- **Fast response times**
  Short lead times and low in process inventories enable the manufacturers to react quickly to market changes and opportunities;

- **Better integration with higher level scheduling systems**
  The application of advanced scheduling methods makes it possible to better predict when products can be processed and become available;

- **Higher machine utilization**
  The use of advanced scheduling methods minimizes the idle time of machines as much as possible.

Advantages of the high level of automation include:

- **The possibility to manufacture unmanned**
  When human operators are required, systems have to be closed down at night or during holidays. Unmanned systems do not have these disadvantages;

- **More consistent product quality**
  It is to be expected that products manufactured in automated systems

will be of higher quality, than when they are manufactured in systems where all kinds of uncontrollable factors influence product quality.

All these advantages combine to the most convincing reason to apply FMS: FMS is the most cost effective manufacturing method when small batches of different, but similar products have to be produced [Bat 85];

1.5 Restrictions of FMS

The application of FMS does not only have advantages: there are some drawbacks as well. An important threat to FMS is low productivity. Designing a flexible production system, without regard to its productivity is not a very complicated problem. To find a reasonable compromise between flexibility and productivity however, is. A number of different factors influence the productivity of an FMS:

- All products have to be clamped on pallets when they are introduced into the system. An interface between product and pallet, called a fixture, is required to accommodate the specific shape of the product. Fixtures are very expensive and it is usually not feasible to use a very large set. A number of alternatives exist:
  - the number of products processed in the system is limited;
  - a large number of different fixtures is to be maintained;
  - fixtures that can be adapted to a specific product are used.

  The last option has the disadvantage that adapting the fixtures may take considerable time; this leads to inefficiencies as well;

- The variety of tool usage in an FMS is another complication that may lead to low system productivity. The number of tools that can be contained in the local tool stores of the machines is limited. When many operations that require different tool sets have to be performed on a machine, tool changes will be inevitable. These tool changes complicate the operation of the FMS considerably and cause the machines to lose time;

- An FMS consists of a number of machines. The productivity of a FMS is the average of the productivity of its machines. When one or more of the machines is poorly used, this will reduce the system's

productivity[1]. For this reason it is important that the orders that have to be processed by the system allow an *even balancing of the workload*. So even when the products, from a technological point of view, are suitable to be processed in the system, the load balancing still poses limitations to the order mix that can be processed efficiently.

Of critical importance is the choice of products that have to be processed in the system. An FMS is designed to process products from a *product family*. When the products in the family are all very similar and use the same tools and same fixtures, it will be relatively easy to design an effective FMS. If however the product family is loosely defined and if a large number of tools and fixtures are required to manufacture the products, a much more complicated problem exists. In this case it will be more difficult to design a system that is able to manufacture all products and that is still productive enough.

1.6 Components of FMS

The definition of FMS mentioned in the beginning of this chapter does not describe how an FMS is implemented. Existing FMS tend to have similar implementations. In this section components that are frequently applied in FMS are described.

1.6.1 Machines

The machines in FMS are general purpose machines that are *process oriented*. They are able to perform operations on a wide variety of products and are generally controlled by a *Computerized Numerical Controller* (CNC). The CNC is responsible for performing the operation on the machine itself; in a *NC part program* the instructions to perform the operation are listed. The machines are equipped with a local tool store that contains the necessary tools; tools are changed automatically. A *process control system* is generally used to detect tool failure and tool wear.

In many cases machines are not able to run completely unattended. The processes that are performed on the machines are critical and errors during production that are not immediately detected can cause large damages.

[1] Not all machines are equally expensive, so poor machine utilization is not always a serious problem. Moreover, in a number of cases, poor machine utilization simply cannot be avoided, for example when a machine tool is needed for just a few products.

Only operations that have proven never to cause problems will be allowed to run unattended.

1.6.2 Product Transport

The transport of products is generally performed by an automated system. In many FMS an Automated Guided Vehicle (AGV) system is used [Mason 86][2]. Another transport system that is frequently applied, uses rail guided vehicles[3]. The transport system interfaces with each machine; pallets are transported automatically. The pallet transport system is controlled by the FMS control system.

1.6.3 Pallets and fixtures

Products are generally transported on *pallets*. The pallet is a simple mechanical device that functions as an interface between product and transport system. The transport vehicles and machines are all able to handle pallets; they are not aware that products with different shapes may be transported on the pallets. Each pallet may hold one or more products.

Products are clamped on pallets by means of *fixtures*. Fixtures are product specific mechanical devices that are able to clamp only one product shape on a pallet[4].

Products that require operations on *machining centers* never leave the pallet once they are clamped. In this case the fixture holds the product in position and is able to resist the forces exerted on the product during the operation.

Products that require operations on *turning centers* leave the pallet when the operation is to start. The turning centers are equipped with manipulators that remove the products from the pallet and places them back when the operation is completed. The function of the fixture in this case is just to position the product on the pallet: the product is not clamped.

[2] An automated guided vehicle is a vehicle that is guided by wires in the floor. Important advantages of these vehicles is that no rails are required on the floor and that the routing can be easily changed.

[3] These vehicles are generally faster and cheaper, but they have the disadvantage that rails are required.

[4] When different products have similar shapes, a fixture may be able to clamp more than one product type.

1.6.4 Tool Room

Tools are required for the operations that are performed by the machines. Each machine has a local tool store where tools are stored that are necessary for the current and expected operations. In a central tool room tools are prepared for operations on the machines. Tools are stored in a central tool store until they are required on the machines. A tool transport system transports the tools to the machines as soon as they are needed [Ber 85],[Mason 86].

In some systems it is possible that all tools that are required to manufacture a set of orders can be stored simultaneously in the local tool store. In those cases the machines are equipped with the necessary tools before a new set of orders is processed. When all orders are processed, worn tools will be removed and the contents of the local tool stores will be adapted to the new set of orders that will be processed next.

If it is not possible to have all tools simultaneously available in the local tool store, it will be necessary to exchange tools between central and local tool store whenever a tool is required that is not locally available.

In many cases an automated tool transport system is used to transfer the tools. Tool room and tool transport system are controlled by the FMS control system.

1.6.5 FMS control System

The operation of the system is controlled by a central computer system. The CNCs of all machines in the system are connected to this central computer system. NC part programs and tool data are maintained on the central computer and are downloaded when needed. In chapter 2 the functionality of the FMS control system is discussed in detail.

1.7 Examples of FMS

In this section a few examples of modern FMS will be discussed. Flexible Manufacturing Systems are generally installed by a machine tool manufacturer. Generally only machines originating from the manufacturer can be integrated. An example of a machine tool manufacturer that has designed and installed several FMS installations is the German company Werner [Rau 87], [Ham 87a], [Ham 87b]. Werner supports a range of flexible manufacturing systems, of which the 'Duplex Zellen' are the best known. These systems consist of two identical machining centers that are equipped with large tool

stores. A large system tool store functions as a backup for the local tool stores. A gantry type robot system that reaches over the whole system is able to exchange tools between the local tool stores and the system tool store. A tool preset station is used to prepare new tools for use in the system.

Two clamp stations are available to clamp and unclamp products. A system pallet buffer is used to store pallets that contain products that are waiting for operations or that are waiting to be unclamped. Pallets can be transported in the system by the same gantry robot that transports the tools. The pallets are equipped with a special mechanical interface to enable the gripper of the robot to handle them. An example of a system where the pallets are transported by the gantry robot is shown in figure 1.5.

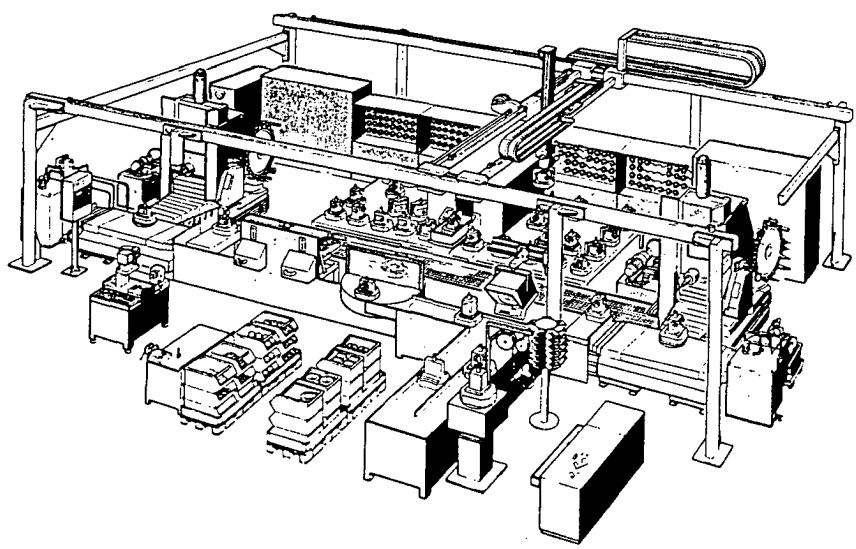


Figure 1.5: A Werner FMS type DFZ 400. Pallets in the system are transported by the gantry robot.

When large workpieces are manufactured in the system, it is not possible to let a robot handle pallets and products. In that case the pallets are transported on a rail guided vehicle. An example of this solution is shown in figure 1.6.

Recent developments of Werner are larger systems that can contain up to 12 machines. In these systems large numbers of pallets are used and special

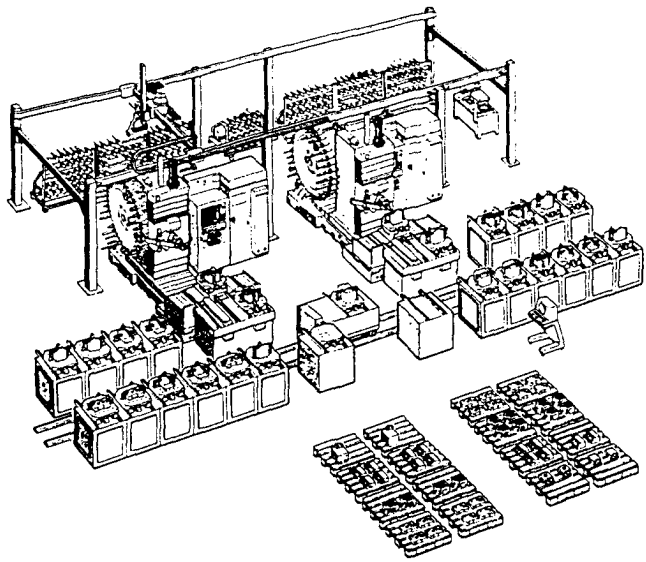


Figure 1.6: A Werner FMS type DFZ 630. Pallets in the system are transported by a rail guided vehicle.

automated warehouses can be applied to store these pallets. In general a more complicated transport system, where a number of Automated Guided Vehicles transfer the products between machines and warehouse, is used. A schematic overview of a large Werner system is shown in figure 1.7.

1.8 History of FMS

Although the growth of FMS applications in industry is a rather novel development, the concept of FMS is not new. Two examples of early FMS systems will be discussed.

In 1968 the American company Cincinnati introduced their *Variable Mission Machining System* (VMM) [Brow 68]. This system was designed to produce gear houses in batches from between 20 to 1000. The system consists of a number of machines that are connected by a roller track system. Pallets containing products circulate in the system waiting for their turn to be machined. The pallets are mechanically coded and can be recognized by the machines in the system. After an operation is performed, the code on the pallet is changed and the next operation can be performed. The sys-

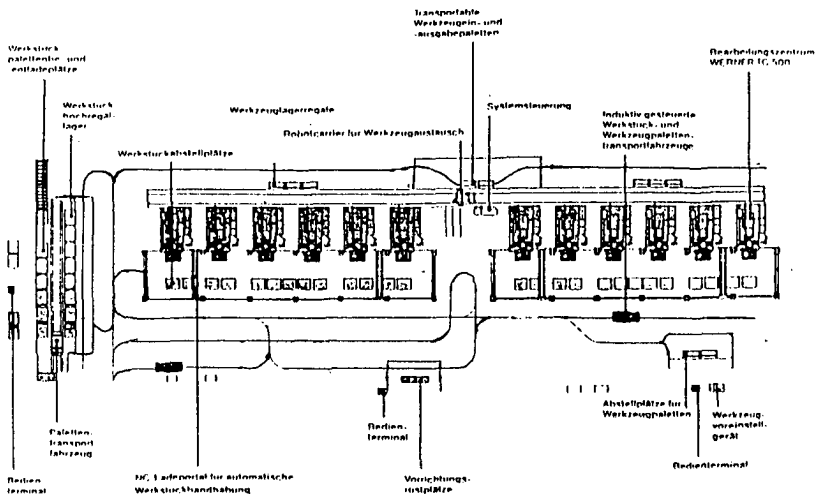


Figure 1.7: A Werner FMS type FFS 500-12. The system consists of 12 identical machines. Pallets in the system are transported by automated guided vehicles. Tools are transported by a robot that can reach all 12 machines. On the left of the picture an automated warehouse and the clamp and unclamp stations are shown.

tem does not require a computer control system. Products, once entered, circulate in the system until all operations are performed. When the last operation is ready the product is removed. The VMM system, as it was used in 1968 was not very flexible according to our current standards: only a very limited number of different products could be processed. In figure 1.8 a drawing of a VMM system is shown.

Another system developed at about the same time was the *Molins 24 system* [Wil 67]. This system was designed for the manufacture of a wide range of small aluminum products. The specially designed machines have tool stores that can hold up to 70 tools. The machine controllers can run NC part programs that are stored on magnetic tape. A tape 'juke-box' offers the possibility to choose part programs at random. Pallets are transferred automatically between the machines and an automated pallet warehouse. The system includes an automatic measuring station that can test the quality of the products. The high volume of produced metal chips by the machines is removed automatically. The whole system is controlled by an IBM 1130 computer.

None of the two systems described above were actual successes. A num-

ber of VMM systems have been built. Even today Cincinnati offers systems under the name VMM. The Molins 24 system was too complicated for its time and especially the control problems were never overcome. A limited prototype has been built for IBM. The specifications of the Molins 24 system were very advanced. It is remarkable that in 1988 the Molins system implemented with modern components would not have been obsolete.

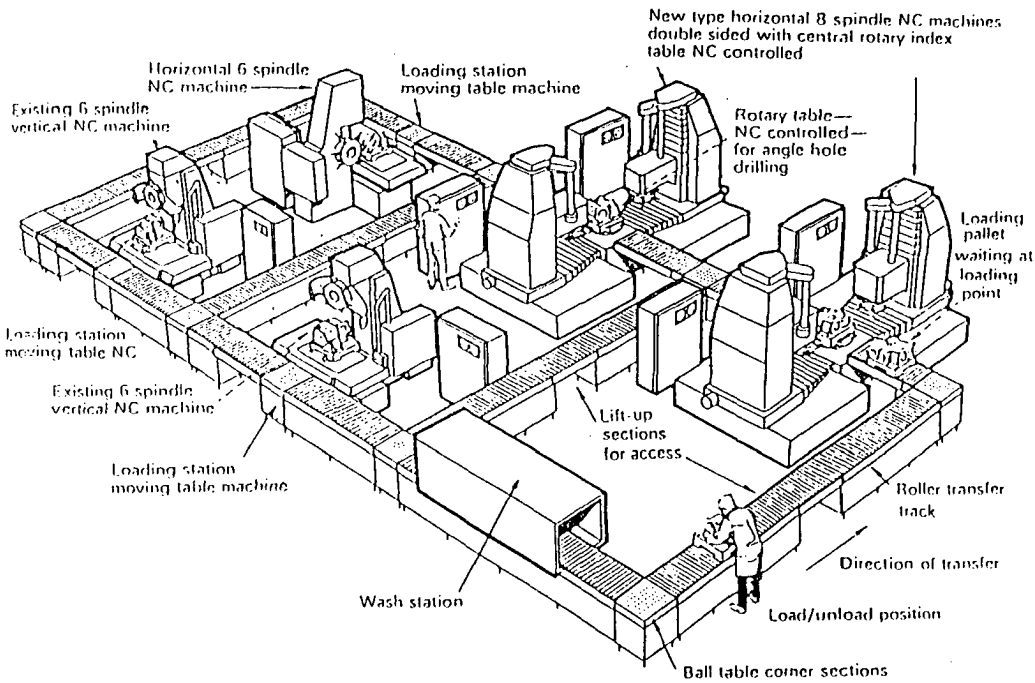


Figure 1.8: An overview of the Cincinnati VMM system.

1.9 Application of FMS

In recent years several institutions have tried to analyze the usage of FMS in the various countries of the world. Unfortunately, in many different countries different definitions of FMS system are used. The results of these investigations therefore have to be interpreted with some caution. In 1986 the United Nations published a report in which the application of FMS in Europe, America and in Japan is analyzed [UN 86]. Some results of this

investigation are presented in this section.

Three countries are currently the leading users of FMS: the USA, West Germany and Japan. America has always been a front runner in the machine tool industry; Numerical Control was developed in the USA and the first FMS applications were also American developments (see section 1.8). The need for FMS in the United States originated from the wish to make the large series production systems more flexible. In most American FMS the number of different products that can be produced is still less than 10.

West Germany is a more recent user of FMS. According to recent studies the major reason to apply FMS in West Germany is to *improve the productivity of systems* [Jut 86]. It is interesting that in Germany a large percentage of the systems consists of only two machines. The market for these small systems still seems to gain importance.

Japan is the largest user of FMS. The systems that are used in Japan are simpler than the systems used in Europe, but they are much cheaper. The machines used in European FMS are generally of higher quality. In table 1.1 the numbers of FMS and FMC in use in 1985 in the USA, West Germany and Japan are shown[5].

In table 1.2 the number of FMS and FMC in Europe is shown. The high number of FMC that several countries report suggests a confusion of definitions. Most probably these figures apply to *single* machines. In accordance with the definitions discussed earlier in this chapter these systems would be classified as Flexible Manufacturing Modules.

In table 1.3 the application of FMS in different sectors of industry is shown. As can be seen, 60% of all FMS is applied in automotive, aerospace and machine tool industries. None of these industries is of great importance in the Netherlands. This is one of the reasons why FMS is hardly applied in the Netherlands[6].

Table 1.4 shows the number of machines in FMS. The majority of the systems has between 2 and 10 machines. As has been said before, the importance of small systems tends to grow.

From table 1.5 it emerges that the flexibility of installed FMS systems is not very high. Almost 70% of the systems produce less than 50 different products.

The cost involved in FMS is generally very high. In table 1.6 the average

[5]The data are not up to date anymore and it can be expected that the number of systems currently in use is considerably larger than the figures suggest.

[6]In 1988 still no FMS according to the definition in section 1.1.2 is installed in the Netherlands.

costs of systems are categorized in three groups.

Germany	25-25
Italy	25
Japan	100
USSR	60
United States	47

Table 1.1: The number of installed FMS and FMC in the important FMS countries. Source: Recent trends in Flexible manufacturing, United Nations 1986.

Country	MetalCutting		Metalforming		Welding		Total	
	FMC	FMS	FMC	FMS	FMC	FMS	FMC	FMS
Belgium	-	-	4	-	-	-	4	-
Bulgaria	..	3	..	-	..	1	-	4
Canada	2	-	2	-	-	-	4	-
Czech.	54	6	43	2	31	1	128	9
France	15	6	5	2	3	1	23	9
Hungary	14	4	-	1	-	-	14	5
Netherlands	10	6	12	8	25	6	47	20
Norway	6	-	2	-	2	-	10	-
Sweden	50	10	-	-	10	-	60	10
Turkey	3	-	2	-	2	-	7	-
UK	..	3(21)	..	(5)	..	-	..	3(26)
Total	154	38(21)	70	13(5)	73	9	297	60(26)

Table 1.2: Installed FMS and FMC in ECE member countries. Source: Recent trends in Flexible manufacturing, United Nations 1986.

Industry sector	Percentage of installed FMS	
	by number	by value
Light automotive (cars, motor cycles)	20	26
Heavy automotive (tractors, trucks)	14	14
Aerospace	9	10
Machine tools	9	9
Automotive parts	9	9
Other sectors	39	32
	100	100

Table 1.3: Distribution of FMS by industry sector in western Europe in 1984. Source: Recent trends in Flexible manufacturing, United Nations 1986.

Number of machine tools	Number of systems (percentage)			
	Europe	North America	Japan	Total
2 or less	28	7	16	51
	(18)	(12)	(17)	(16)
3 - 5	62	19	34	115
	(40)	(31)	(37)	(37)
6 - 10	44	25	26	95
	(28)	(41)	(28)	(31)
11 or more	22	10	16	48
	(14)	(16)	(18)	(16)
Total	156	61	92	309
	(100)	(100)	(100)	(100)

Table 1.4: Number of machines per FMS. The number of systems and, in parentheses, the percentage distribution by size category are specified. Source: Recent trends in Flexible manufacturing, United Nations 1986.

Number of product variants	Europe	North America	Japan	Total
1 - 10	22	15	15	52
	(34)	(41)	(27)	(33)
11 - 50	32	8	13	53
	(49)	(22)	(24)	(34)
51 - 100	5	5	11	21
	(8)	(13)	(20)	(13)
101 -	6	9	16	31
	(9)	(24)	(29)	(20)
Total	65	37	55	157
	(100)	(100)	(100)	(100)

Table 1.5: Number of product variants for 157 FMS installations. The number of systems and, in parentheses, the percentage distribution by size category are specified. Source: Recent trends in Flexible manufacturing, United Nations 1986.

Investment costs ($US million)	Europe	North America	Japan	Total
Less than 3	32	5	6	43
	(54)	(18)	(55)	(44)
3 - 7	15	3	2	20
	(26)	(11)	(18)	(20)
More than 7	12	20	3	35
	(20)	(71)	(27)	(36)
Total	29	28	11	98
	(100)	(100)	(100)	(100)
Total investment costs	280	295	117	692
Average investment	4.7	10.5	10.6	7.1

Table 1.6: FMS investment costs. The number of systems and, in parentheses, the percentage distribution by size category are specified. Source: Recent trends in Flexible manufacturing, United Nations 1986.

Chapter 2

FMS Control Systems

2.1 Introduction

2.1.1 Layered design of control systems

Control systems for automated production systems such as Flexible Manufacturing Systems can be very complex. When complex systems are to be designed, it is advisable to distinguish four separate development steps [Phi 87]. The definitions of these four steps are discussed below.

- **Reference Model**
  A reference model specifies the general structure of the system and shows which tasks have to be executed; the relation between the tasks is shown. *The reference model shows an organization, a structure of tasks*;

- **Architecture**
  The architectural description of the system describes the functionality of each of the tasks. Allowable inputs and outputs are defined and the relations between the tasks are specified. *The architecture describes what the components do and describes the relations that exist between the components*;

- **Implementation**
  In the implementation or design phase the internal behavior of the system parts is defined. Whereas the architectural specification defines the *functionality* of the system parts, the implementation defines how that functionality can be achieved;

- **Realization**
  The last step in the design of a system is the realization. In this step the physical means with which the system will be implemented are specified.

The four design steps form a hierarchy. A reference model can lead to a number of different architectures, each architecture can be implemented in a number of ways and the implementations can be realized with different physical means. A reference model is described in very general terms and is valid for a range of systems, a realization description is very detailed and is applicable only to one specific system. A summary of the development steps is shown in figure 2.1.

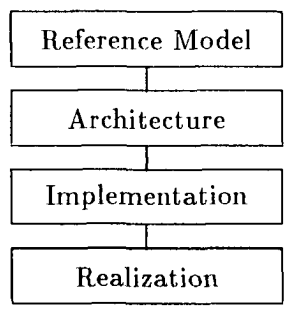


Figure 2.1: The four development steps.

In theory a design can be depicted as a straight line from reference model to realization: the four development steps are independent. However in practice interactions do exist between the development steps. Although it is for example possible to design an architecture without having to consider implementation issues, frequently implementation possibilities and impossibilities will have an influence on the architectural specification.

Two well known reference models are the *Open System Interconnect* reference model from the International Standards Organization (ISO-OSI) and the *reference model for production systems* (known as the NBS model) from the American National Bureau of Standards. The ISO-OSI model describes the required functionality for data communication systems [Tan 81]. Many data communication systems currently available have architectures that are

derived from this ISO-OSI model. The other reference model, the NBS model, will be discussed in some detail in the next section.

Currently an ESPRIT project aims at the development of a new reference model: the *CIM-OSA* model[1]. This reference model is intended to provide an overall framework for the development of CIM architectures [Huy 86].

2.1.2 NBS model

The American National Bureau of Standards (NBS) has defined a reference model for production systems that is widely used. The reference model describes a number of layers that will generally be needed to control a complete factory. Several authors suggested modifications and enhancements to the model [Bie 88a], [Bie 88b], [Phi 87]. An adapted version of the NBS reference model is shown in figure 2.2.

To understand the principle of the reference model, the basic concepts of *Command-Unit, Command-Unit Controller* and *Control Interface* should be understood. In figure 2.3 these concepts are illustrated.

Each Command-Unit consists of a Command-Unit controller and a number of lower level Command-Units. The Command-Unit Controller on level (N) coordinates the Command-Units on level (N-1). The Control Interface is the definition of the commands and statuses that can be exchanged between a Command-Unit and its superior controller. For example, in the NBS reference model a cell consists of a cell controller and of a number of workstations; a workstation consists of a workstation controller and of a number of automation modules.

The separation into levels of detail is important. The controllers of each layer are only aware of their superior controller and of the controllers immediately below them. Commands issued by the superior controller are decomposed into a set of commands that are processed by the lower controllers.

The functionality of the layers of the reference model have not been defined in great detail by the National Bureau of Standards. In general, controllers at the higher levels are responsible for long and medium term planning and resource optimization. The controllers at the lower levels are responsible for executing the plans that have been made at the higher levels. A summary of the functionality of each layer according to [Bie 88b] is

[1] ESPRIT is the abbreviation of European Strategic Programme for Research and Development in Information Technologies. CIM-OSA is the abbreviation of Computer Integrated Manufacturing - Open System Architecture.

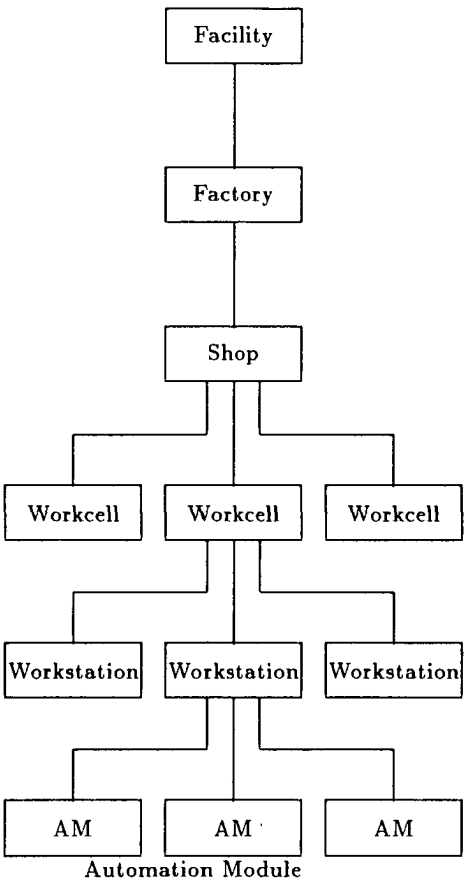


Figure 2.2: The NBS reference model for production systems.

discussed below:

- **Facility**
  The major goal of the facility controller is to let the facility survive in a changing economic and technological environment. The controller tries to achieve this by optimizing the usage of the production resources of the factory and by guaranteeing that the products demanded by the consumer markets are produced in sufficient quantity. The production facility controller is responsible for the (very) long term planning of the production system;

- **Factory layer**
  The factory layer contains a factory that receives its orders from the facility controller. The major goal of the factory is to realize profits by producing and selling products. The factory controller negotiates with consumers in order to determine production targets, the factory controller negotiates with suppliers in order to purchase raw material at acceptable prices. The factory orders the shop to manufacture the products that it agreed to deliver to consumers;

- **Shop layer**
  The shop is responsible for processing orders before their specified due dates. The shop allocates orders to the available workcells and prepares products to be processed by these workcells;

- **Workcell layer**
  The workcell controllers are responsible for coordinating the operations of the workstations in the cells. The sequence of operations on the workstations is determined by the workcell controller. The resources that are available to the workcells are allocated to the workstations that require them;

- **Workstation layer**
  The workstation layer consists of a set of workstations. The workstations are the command-units that are responsible for performing operations on products. Products are physically modified by the workstation. Products are delivered by the product transport system; an operation is performed on them and they are removed again by the product transport system;

- **Automation Module layer**
  The automation modules are the sub-systems of the workstations that

perform the actual operations that are necessary to physically modify the product. The sub-systems are coordinated by the workstation controller in such a way that the intended operation is performed on the product.

The NBS reference model as presented in this section is applied in a large variety of production systems (implicitly or explicitly). The NBS reference model provides a framework for the design of a controller for a flexible manufacturing system.

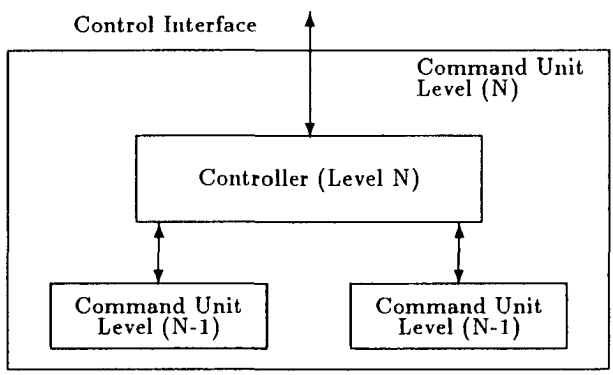


Figure 2.3: The relation between controller and command-unit.

2.1.3 The NBS model applied to FMS

For the design of an FMS controller not all parts of the NBS reference model are relevant. Only the workcell, the workstation and the automation module layer are of direct importance. The other layers fall outside the scope of FMS. In an FMS the machines can be regarded as workstations; the FMS control system functions as a Workcell controller. The shop layer is of some importance because, as will be shown, the FMS receives its orders from a shop controller.

The operations on the machines in an FMS are controlled by components called *Computerized Numerical Controls* (CNC). The functionality of these

controllers, match the functionality of workstation controllers: the CNC is responsible for performing a complete operation on a product. After the product has been processed, it is removed by the transport system.

The workstations receive orders from the workcell controller. Status messages are sent back to inform the workcell controller about the progress of the operations on the workstation. The Workcell controller guarantees that the operations on products are performed in the right order. Programs, tools and pallets are made available to the workstations when needed. The relation between the workcell controller and the workstations in an FMS is shown in figure 2.4.

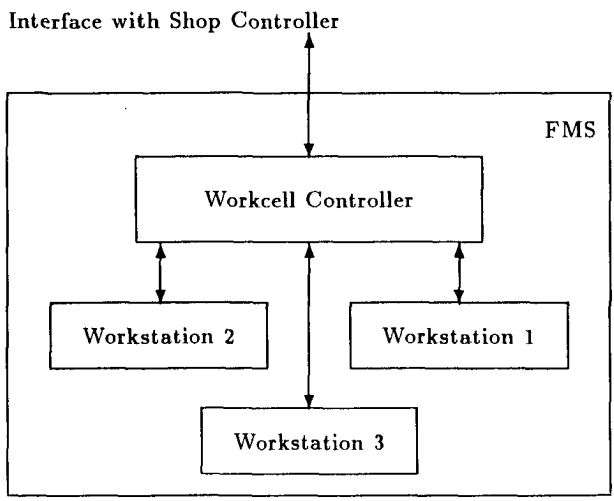


Figure 2.4: The relation between workstations and workcell controller.

The shop controller commands the cell: the FMS. The orders allocated to the cell by the shop controller, will be processed under the responsibility of the workcell controller.

2.2 Functionality of an FMS controller

In this section an overview will be given of the functionality of FMS control systems. Figure 2.5 sets out the components.

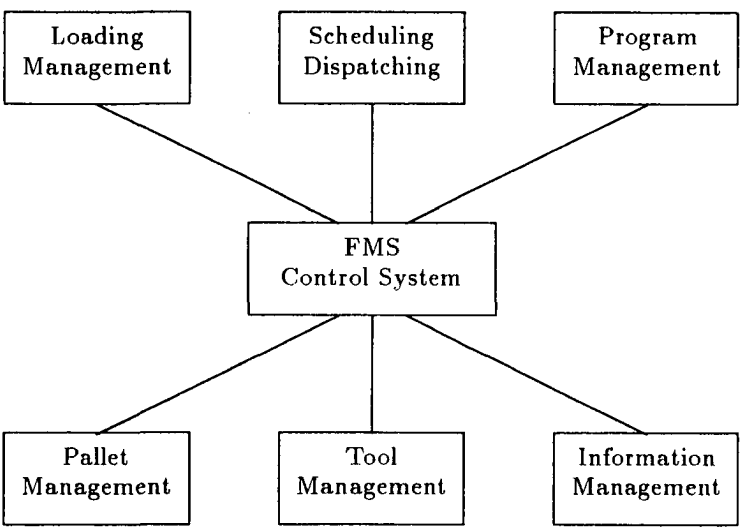


Figure 2.5: An overview of the tasks in an FMS control system.

2.2.1 Allocation of operations to machines

An FMS consists of a number of machines that are able to perform operations on products. Normally, an FMS system is able to process a large number of different parts, each part requiring different sequences of operations. To perform these operations only limited resources are available, so an important task of an FMS control system is to have the machines perform the required operations, in such a way that an optimal use is made of these resources. In many systems this task is performed in two steps: *scheduling* and *dispatching* [Sie 87], [Wer 87], [Acti 83].

Scheduling

Scheduling can be defined as the process of generating a sequence of operations for all machines *before* the actual production starts. The result of this activity is a *schedule* in which for each machine the sequence of operations is specified. The problem of finding the optimal schedule has been proven to be *NP hard*. NP hard problems are characterized by the fact that the amount of calculations required to solve the problem does not grow *polynomially* with the size of the problem, but *exponentially*[2]. For all but the smallest problems, it is therefore infeasible to solve the problems by using numerical methods.

A common approach to the scheduling problem that does not have the disadvantage of requiring enormous computational resources is scheduling by application of *priority dispatching rules*[3]. Such a schedule is made by *simulating* the operations in the system. Whenever a machine finishes an operation, a priority dispatching rule is used to determine which product has to be processed next on that machine. An example of a priority dispatching rule is the *Shortest Processing Time* (SPT) rule: 'from the products that are waiting for an operation that can be performed by the machine, choose one whose operation length is shortest'. Another example of a dispatching rule is the *First Come First Served* (FCFS) rule: 'from the products that are waiting for an operation that can be performed by the machine, choose one that has been waiting longest'. By applying these rules repeatedly, a sequence of operations can be determined for all machines simultaneously.

Although it is not possible to obtain a *globally optimal* schedule by applying these dispatching rules, it will be possible to generate an *acceptable*

[2]NP is the abbreviation of Non Polynomial.

[3]The use of the word *dispatching rule* in the context of the scheduling problem leads to confusion. A better name for these rules is "priority *scheduling* rules".

schedule. Examples of optimization criteria that are frequently applied in dispatching rules[Bla 82]:

- **Maximizing machine utilization**
  The machines used in FMS are expensive. A system in which the machines are frequently idle will not be very productive and hence will produce expensive products;

- **Minimizing makespan**
  The makespan is the total time it takes to process a complete set of orders. The shorter the makespan, the sooner the next set of orders can be processed;

- **Minimizing the maximum lateness**
  Products usually have to be available before a specified time: the *due date*. When the system does not succeed in processing the products before their due date, it is not possible to perform operations on the products in other departments of a factory. Availability before the due date is called *negative lateness*;

- **Minimizing the average lateness**
  Instead of minimizing the maximum lateness, the average lateness may be considered to be more important[1];

- **Minimizing Average tardiness**
  The tardiness criterion resembles the lateness criterion. Unlike lateness, tardiness can not have a negative value. If a product is early it has a tardiness of zero.

The schedule that is obtained by applying one of the priority dispatching rules, specifies for each machine which operations have to be executed. With this information it is possible to determine in advance when pallets and in which order have to be transported, when products have to be clamped, which tools are required, which fixtures will be used and so on. A complete list of all activities to be performed can be generated.

Schedules can be very detailed or they can be specified in more general terms. In a detailed schedule each activity and the exact start time is specified. For example in the activity list for the tool operator will be

[1] The average lateness is usually not a very interesting criterion. The fact that one product is very late can seldom be compensated by another product being very early.

specified at which time the operator will have to start preparing a new tool and at which time the preparation should be complete. A more general specification specifies that the tool operator must have a set of tools available before a specified time.

The advantage of a detailed schedule is that all required activities are very precisely known and that no surprises are to be expected. Everything has been checked, all required resources are available and the specified plan has been proven to work. An important disadvantage is that a very precise schedule is not very stable. If one of the assumptions that has been made during the scheduling proves to be wrong, the schedule will become invalid[5].

Dispatching

After a schedule has been made the actual production can start. The activities specified in the schedule will be executed under control of a *dispatcher*. The dispatcher issues orders to the machines and to operators in order to have each machine perform the operations that were specified. The dispatcher has two inputs:

- the actual system situation;
- the schedule.

The goal of the dispatcher is to take decisions in such a way that the schedule is executed as precisely as possible. Whenever the actual system situation shows a deviation from the schedule, the dispatcher will try to minimize the deviation. When the actual system situation differs too much from the predicted system situation, the dispatcher may request the scheduler to generate a new schedule. An overview of the relation between scheduler and dispatcher is shown in figure 2.6.
A number of events in the system cause the dispatcher to become active:

- An operation finishes on one of the machines. A new operation has to be started and preparations for the next operation have to be made;
- An operator completes an activity. A new operation may have to be performed by the operator;

[5] An example of a disruption is a tool failure. When a tool breaks, in the best case the operation will take longer than was planned. The product will become available later and the resources remain allocated longer than was foreseen. When the resources are required for other operations, those operations will be delayed and so on. A small disruption can have large consequences for a detailed schedule.

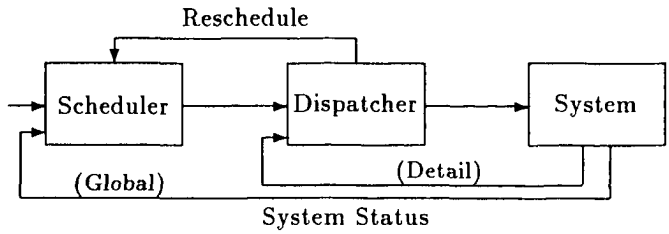


Figure 2.6: The relation between scheduler and dispatcher.

- An error situation is detected that causes an operation to take longer than was expected. The consequence of the delay has to be considered;
- An internal timer expires. An operator or machine that is idle has to start performing an operation.

The dispatcher analyzes the new situation, issues orders if necessary and waits for the next event to occur.

2.2.2 Loading management

The schedule that has been made specifies when products are to be processed in the system. Before products can be processed, they need to be clamped on pallets. The actual clamping is performed by human operators on *clamping stations*. The FMS control system orders the operators to clamp the products when needed. The operators signal the completion of the operation, when they finish clamping the product.

When products have had all their operations, they have to be removed from the pallet. The FMS control system orders the operators to unclamp products whenever finished products become available. The pallet and fixtures can be used to clamp a new product.

2.2.3 Program management

The controllers of the machines, the CNCs, require *part programs* to perform the operations. These programs are usually developed by programmers from a special *process planning department* and are maintained at a *part program database*. When an operation is to be executed on a machine and when the

program is not yet available, it is the task of the FMS control system to make the program available at the CNC. The capacity of the CNC to contain part programs is limited, so it may be necessary to delete part programs at the CNC to create space for the new program.

Only part programs that are tested and that are proven not to cause any problems are used in an FMS. It is imaginable however that a program containing an error is used and that an operator at the machine judges it necessary to adapt the program. In that case the changed program will be uploaded to the part program database in order to prevent having to make the change each time the program is downloaded again. The newly uploaded program will have to be authorized by the programmer before it is released for further usage.

2.2.4 Tool management

In an FMS tools can be stored in a *central tool store* or in the tool stores of the machines. A *tool room* facility is used to assemble and measure new tools. The new tools are stored in the central tool store until they are needed on a machine. The capacity of the tool stores of the machines is limited and the total number of tools that may have to be used on a machine is frequently much larger than the number that can be stored in the local tool store. The FMS control system will therefore have to ascertain that the tools required for the coming operations are available. A *tool transport system* is used to transport tools to the machines [Ber 85], [Mason 86].

The FMS control system is responsible for having the tools available at the time and place needed. The control system orders the tool room operators to prepare new tools when not enough tool capacity is available in the system. From the schedule, made by the scheduling system, can be derived when tools are required by the machines. The FMS control system orders the tool transport system to transport tools to where they are needed. Tools that are no longer needed at a machine are transported back to the central tool store.

When tools unexpectedly fail during an operation, and when no replacement tool is locally available, the FMS control system will try to replace the broken tool with a replacement from the central tool store in order to enable the operation to continue. If this is impossible, the operation is aborted.

When a part program is developed, the programmer assumes nominal tool dimensions. The nominal tool dimensions are fixed and are known for each tool type (Tool Class). The actual tools that are used however,

have dimensions that differ slightly from the nominal dimensions. Special equipment in the tool room is used to measure the actual dimension of each tool before it is actually used in the system. The CNC of the machine on which the tool is used, makes corrections for the actual tool dimensions.

In addition to the tool dimension data, in many systems the remaining tool life time is maintained for each tool. When a tool is used on a machine the CNC subtracts the time the tool has been used from the tool life time. Tools that do not have any tool life time left have to be removed from the system.

It is the responsibility of the FMS control system to maintain the tool data and to download the data to the machines where the tools will be used. The adapted tool data have to be uploaded from the CNCs when the tools are removed and sent back to the central tool store. Only tools that have sufficient tool time left are stored, the other tools are useless and can be disassembled or discarded.

2.2.5 Pallet management

In FMS systems products are clamped on pallets[6]. Fixtures are used as a mechanical interface between pallet and product. Pallets can be stored in a *central pallet buffer* or in the pallet buffers at the machines. The pallets are transported through the system by the *pallet transport system*. It is the responsibility of the FMS control system to ensure that pallets are transported to the machines, shortly before the operation on the products clamped on the pallets, is expected to start. In the schedule is specified when operations start at the machines. In very detailed schedules all pallet transports will already be specified. In less detailed schedules the dispatcher has to supply in the details.

2.2.6 Information management

In order to analyze the behavior of the system, it is important to have relevant data available. A *Management Information System* gathers this data, processes them and presents the management with reports containing data about the system performance. The importance of these reports is that they enable the management to improve the system performance. Examples of data that will be gathered are:

[6]Products are not *clamped* on pallets when lathes or turning centers perform the operations. In those cases, pallets are used just to *transport* the products.

- the operations that are performed by each machine;
- the problems that occurred on each machine:
  - tool failures;
  - tools not available;
  - pallet not available;
  - erroneous part programs;
  - problems during machining.
- the tools that are used for each operation;
- the average remaining tool life of all tools in the tool stores;
- the frequency of pallet transports.

The data that have been gathered should be processed to be presented in a meaningful way. Examples of questions that may be asked by the management are:

- Why is the utilization of machine A lower than expected?
- What causes the frequent errors on machine B?
- Is there reason to assume that the low performance of the system is caused by too little pallet transport capacity?
- Are there products that cause problems more frequently than other products?

It is important that the data necessary to answer these questions are gathered. Some standard reports will be generated routinely. Specific processing is required for each particular question that may be asked.

2.2.7 Exception and error handling

No matter how well the operations in an FMS are planned or how well the processes are controlled, exceptions and errors will always occur. All kinds of possible exceptions need to be considered. An *exception* is a non-standard situation that can be handled within the standard system procedures. During the design of the system, these situations have been foreseen and appropriate procedures have been specified. Very severe exceptions for which

no appropriate procedures have been defined, are called *errors*. When these situations occur, it is important to minimize the 'damage'. Special error recovery procedures specify what has to be done when these situations occur.

Sometimes the cause of an exception is understood, sometimes it is not. However, in all cases, it is of the utmost importance that the FMS control system be able to handle the exceptions in a safe way that allows the system to continue working.

If a disturbance has occurred in the system, it is important that the consequences for other system elements are kept minimal. In the list below a number of exceptions are discussed. The list contains just a few situations that could occur in a real system. The required response of the FMS control system is briefly discussed. Examples of exceptions are:

- **Tool fails**
  When a tool breaks, the CNC tries to select a replacement tool of the same tool type. If that tool is not locally available, the FMS control system will be requested to deliver a replacement tool;

- **Tools are not available locally**
  An operation is started on a machine for which all required tools are not available. This situation can be handled as if a tool failure occurred;

- **Tools are not available at all**
  A tool, requested at the central tool room, cannot be delivered. This will make it impossible to perform the operation for which the tool was requested. The schedule will have to be adapted;

- **Part program not available**
  An operation is attempted while the required program is not available. The program will have to be downloaded from the program database;

- **Pallet cannot be transported**
  When a pallet buffer failure or a pallet transporter failure makes the transport of a pallet impossible, an operation on a machine cannot start as planned. The FMS control system will allocate a new operation to the machine and the product on the pallet that could not be transported will be processed as soon as this problem is solved;

- **Operation takes longer than planned**
  During the generation of a schedule, assumptions have been made

about the length of operations. If for some reason operations take longer than expected, the schedule cannot be executed as planned. The dispatcher will have to deal with this problem;

- **Part program contains error**
  In the case that a part program contains an error, the operation cannot continue as planned. If possible, the machine operator will try to correct the error. Delay is the least serious problem; the error in the part program may have damaged the product in which case the operation on the product will be aborted. In the worst case the machine is damaged and cannot be used for some time: a completely new schedule will have to be made;

- **Machine breaks down**
  When a machine failure occurs and when the problem cannot be solved immediately, the schedule will be disrupted severely. The dispatcher will request the scheduler to generate a new schedule;

- **CNC has to be reset**
  If the CNC has to be reset due to some internal error, the FMS control system must be able to handle the interruption of the data connection without problems.

When *real* errors occur it has to be accepted that the system does not have a very smooth solution. To some problems there just are no very good solutions. However, in these cases the FMS control system should be able to recover gracefully from the problem. The most serious error situation that can occur is an error of the FMS control system itself. Both software and hardware errors can disrupt the operations in the system completely.

Internally the FMS control system has to be able to handle error situations. The control system has to have facilities to check its internal integrity. When problems are detected error recovery procedures with operator intervention may be applied or (parts of) the control system may have to be re-initialized.

It is important that the FMS control system can be restarted without having to *physically initialize* the whole system. For example, having to remove all pallets from the machines to restart the control system, would make the handling of an FMS after error situations rather difficult. The system status should be easily recoverable to enable a smooth restart. Even more important than an easy restart is a *safe* restart. It should be possible

to guarantee that the representation of the system status in the FMS control system is correct.

2.3 Requirements for FMS controllers

In this section, properties of FMS controllers that are to be considered of crucial importance are discussed. FMS control systems will be judged on the following criteria:

- **Efficient utilization of resources**
  The equipment used in FMS systems is expensive. Resources like pallets, fixtures and tools add greatly to the cost of the system. Making optimal use of equipment and resources leads to efficient operation of the system;

- **Simplicity of concept**
  In order for people to have confidence in a system they are working with, they have to understand its general operation. A simple control architecture helps them to understand the functionality of all components;

- **Simplicity of operation**
  The human interface of a system is of crucial importance. User-friendly facilities have to be available for all activities that have to be performed by operators in the system;

- **Operational reliability**
  The FMS control system is the component that integrates separate components into a system. Failure of the FMS control system disrupts all system operations completely;

- **Scheduling predictability**
  Products manufactured in the FMS are usually required for subsequent process steps in the company. In order to avoid disrupting these, it is important that products become available at their scheduled times;

- **Robustness**
  Once a minor irregularity has occurred, the control system must be able to remain functioning. A small problem must never be allowed to paralyse the whole system; the system has to be able to recover automatically;

- **Modularity**
  The FMS control system has to be constructed from several modules, with each module having a well defined functionality. The functionality of the FMS system can then be easily adapted when the need arises;

- **Extensibility**
  The FMS control system has to allow easy extensions. It should be possible to simply add more capacity to the control system when the system to be controlled is extended;

- **Generality**
  The FMS control system has to be able to control components from different manufacturers;

- **Cost**
  The cost of hardware and software of the control system have to be commensurate with the cost of the complete system.

2.4 Examples of FMS control systems

In this section two examples of control systems by important German FMS manufacturers will be discussed.

2.4.1 Werner SC1

The German company Werner has been active in the FMS market for some years[7]. In chapter 1 some systems of Werner have been shown. In this section the control system that is applied by Werner is discussed. The Werner SC1 control system is an FMS control system that can handle up to 12 machines, a tool presetting unit, a tool supply system, a central tool store and a pallet transport system [Wer 87]. In figure 2.7 an overview of the Werner SC1 is shown.

The system has a menu driven operator interface. An overview of the menu options of the SC1 is shown in figure 2.8. A summary of the menu options is given below:

- **Jobs**
  The jobs option is used to edit orders for each machine. Orders can be

[7] At the EMO in Paris in 1983 Werner presented a small Flexible Manufacturing System.

added, changed or removed. The number of products of each product type and the program that is to be used are specified;

- **Pallets**
  The pallet option enables the operator to inspect the status of the pallets in the system;

- **Tools**
  The tools option is used to inspect the data on tools in the system. The available tools per machine, the remaining tool life of those tools, the tools that are allocated to the machine and the tools that will have to be removed, can be shown;

- **NC Programs**
  The NC programs option enables the operator to download, upload, edit, add and delete NC programs. A total amount of 15 MB of memory is available for the storage of NC programs.

- **Operation Data**
  The operation data option can be used to:

  - display and analyze data concerning each machine;
  - show a chronological list of events;
  - show the current system status;
  - show job-related operating data.

  The information can be presented in plain text or graphically as charts.

- **Installation**
  The installation option is used to set and edit system parameters. This option is not used on a regular basis.

The Werner control system stresses the optimization of tool use. The criterion to allocate jobs to machines is to minimize tool changes. The tool requirements are calculated and tools are transported automatically (if the proper equipment is available). The tool data are maintained and downloaded by the FMS control system. Programs are downloaded when they are not available at the machines. The control system has knowledge of the status and controls all operations.

A single central computer is used. The system components are connected to the computer by means of serial lines or by means of a small Local Area

Network. An overview of the hardware configuration of the SC1 system is shown in figure 2.9.

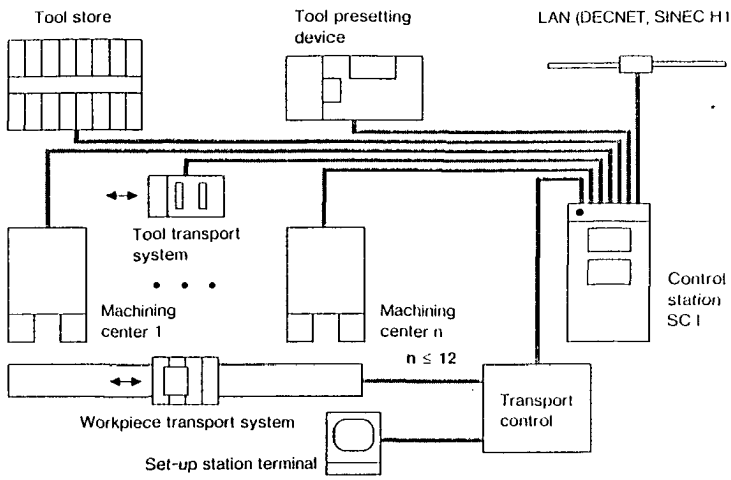


Figure 2.7: A Werner SC1 control system controlling an FMS.

2.4.2 Siemens FMS-M

The German manufacturer Siemens has developed an FMS control system called FMS-M [Hen 86], [Sie 87]. The control system is designed to control FMS's in which all machines are controlled by Siemens controllers. Four versions of FMS-M are available to control systems of 4, 8, 12 or 16 machines. A diverse range of equipment can be included in the system: AGVs, measuring machines, tool setting devices, as long as they are controlled by Siemens controllers. An overview of the functionality of the FMS-M system is shown in figure 2.10.

The FMS-M system uses a very simple scheduler that tries to optimize the machine utilization. The schedule is generated interactively: the operator takes the scheduling decisions and the system presents the operator with the consequences of his decisions. The following restrictions are considered when generating the schedule:

- the due date of the products;

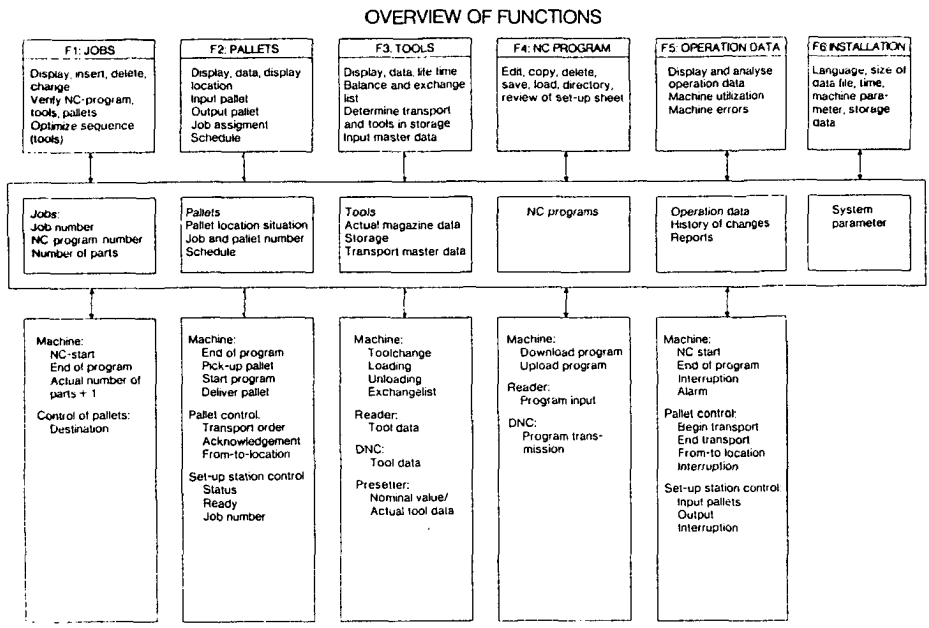


Figure 2.8: The menu options of the Werner SC1 control system.

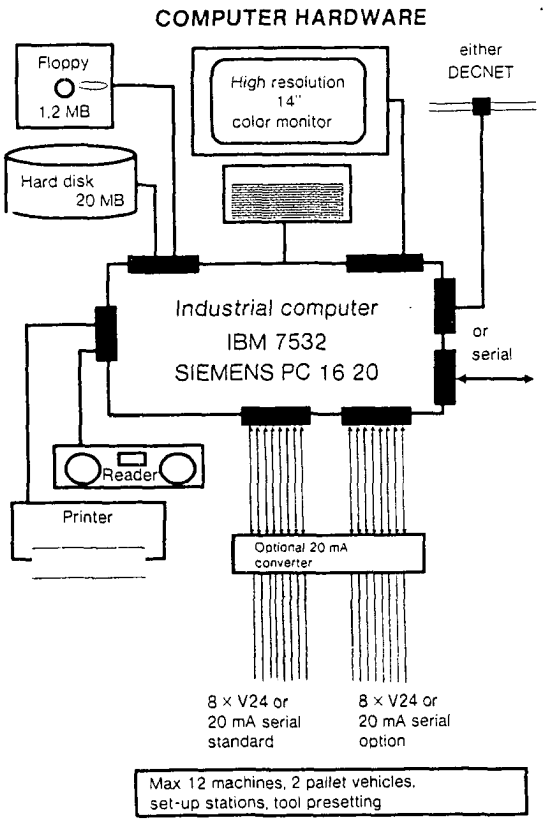


Figure 2.9: The hardware of the Werner SC1 control system.

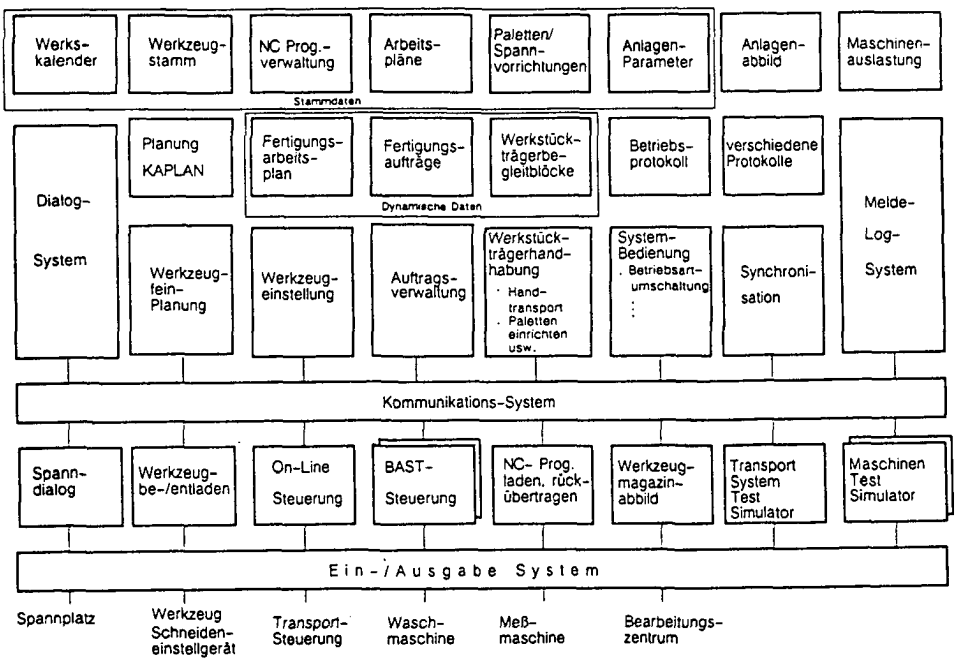


Figure 2.10: The functionality of the Siemens FMS-M control system.

- the available pallets and fixtures;
- the available tools and tool storage capacity.

When the schedule has been generated, the system generates lists of required actions for the tool operators and for the operators responsible for clamping and unclamping the products.

A dispatcher is responsible for executing the schedule. Machines send messages when they are ready processing a product. The dispatcher then allocates a new product.

Like the Werner system, the FMS-M system has a menu driven operator interface. Information can be requested about tools, pallets, jobs, NC programs and the status of the system. The FMS-M system can be parameterized to control a particular system. Examples of parameters that can be set are shown in the following list:

- the number and type of machines;
- the layout of the system;
- the number of central buffer positions;
- the number of shifts;
- the sizes of the NC part program memories;
- the sizes of machine pallet buffers;
- the number of transporters;
- the size of the local tool stores;
- the number of pallets in the system.

2.5 Evaluation of FMS control systems

Quite a large number of different control architectures have been developed for FMS control systems [Wer 87], [Sie 87], [Mol 83], [Hea 86], [Tor 82], [Pur 82], [Acti 83] and [Ono 87]. Although control systems based on these architectures are currently used to control real FMS, they are not without disadvantages. Some of these disadvantages are:

- Most systems apply a *scheduler* to determine the sequence of operations on the machines. In order to create the schedule, *assumptions* concerning the duration of operations have to be made. Disruptions in the system operations cause the assumptions to become invalid. It is difficult to create a schedule that is not sensitive to these disruptions.

A *dispatcher* is required to execute the schedule. The dispatcher is continuously confronted with disruptions of system operations. The need to cope with these disruptions complicates the dispatcher considerably. When the dispatcher judges the deviations from the predicted system status to be too large, the scheduler is requested to create a new schedule.

- Both the scheduler and dispatcher are complicated components of the FMS control system. For human operators it is frequently not clear why the system behaves as it does.

- The traditional control systems cannot be easily extended. To allow for future growth of the system, the control system usually has to be prepared to deal with the maximum number of machines that may have to be connected.

- The high costs of the traditional control systems added to the high costs of the FMS components, prohibits application of Flexible Manufacturing Systems in small companies.

The control systems mentioned above are all traditional systems based on hierarchical architectures.

In this thesis a control architecture will be presented that does not have any of the disadvantages discussed above. The *Distributed Flexible Manufacturing System* (DFMS) architecture is not hierarchically oriented, like traditional control architectures, but is defined as a set of equally important, cooperating components.

Parallel to the DFMS project several authors have investigated non-hierarchical architectures. A non-hierarchical control structure for a FMC Cell is described in [Duf 87]. Examples of distributed architectures for FMS are discussed in [Hat 85], [Shaw 87] and [Shaw 88].

In the next chapters the DFMS architecture will be discussed.

Chapter 3

The DFMS architecture

In this chapter a new architecture for an FMS control system will be discussed[1]. The new control architecture does not have the disadvantages that are normally attributed to traditional control systems. The architecture is based on a non-hierarchical organization and can easily be implemented distributed, without having to apply a central computer system. The proposed system will be called *Distributed Flexible Manufacturing System* (DFMS).

3.1 Overview of the architecture

In DFMS the sequences of operations on each machine are controlled by components called *Station Managers*. Each machine has its own Station Manager. The Station Managers *negotiate* with each other to determine which operation will be performed next by each machine. The Station Manager behaves as an *agent* for the machine. It makes sure that the machine gets orders allocated, and is responsible for the availability of the required tools, part programs and products.

In the DFMS concept no schedule is made before the actual production starts. The system is scheduled by handling an *Operation Queue* for each separate machine. An Operation Queue is a list of operations that will be performed by the machine[2]. Each machine performs the operations from its queue in the sequence in which they were received by the Station Manager.

[1] In this architectural description of DFMS, only functions and functional relations are defined. Implementation of the DFMS architecture is addressed in chapter 4.

[2] The Operation Queue is *not a physical* queue of parts.

No attempt is made to optimize the productivity of the machine by changing the sequence of operations.

When a machine finishes an operation on a product, the Station Manager checks the product to see if more operations are needed. To each product a list is attached that describes the sequence of operations that have to be performed. This list is inspected by the Station Manager[3]. If another operation needs to be performed, the Station Manager negotiates with other Station Managers to decide which machine is best suited to perform the operation: i.e. the machine with the shortest Operation Queue that is able to perform the particular operation. The order is then added to the Operation Queue of that machine.

Operations are allocated to machines during the *actual* production, not *before* the production starts. No operation schedule has to be generated. No assumptions have to be made about the length of operations or the availability of tools[4].

The very simple dispatching method that is used in DFMS enables a completely distributed implementation of the FMS control system. In order to take a dispatching decision, it is not necessary to have access to global data. Only information about the length of the Operation Queues on all machines is required. To obtain this information all Station Managers are connected through a network over which they can communicate.

Apart from an interface to the network, each Station Manager has an interface with the controller of a machine. This interface is used to give commands to the machine and to receive status messages.

3.1.1 Components of DFMS

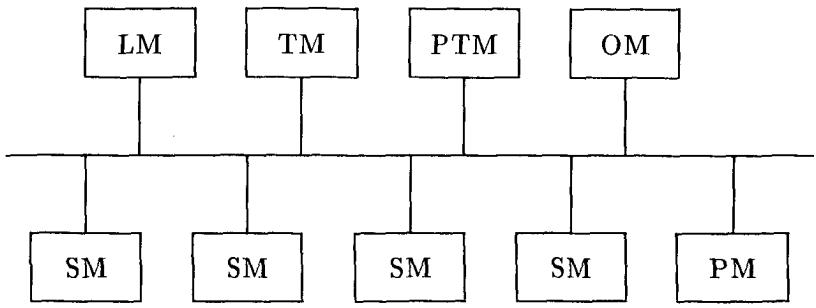
The Station Managers do not only communicate with each other, they also communicate with modules that perform general services. These modules are called *Function Modules*. Examples of services performed by Function Modules are pallet transport, tool transport and NC part program storage. The Station Managers require services from these Function Modules in order to create conditions necessary to perform operations.

A schematic overview of the DFMS control architecture in which Station Managers, Function Modules and their connection are shown, can be found

[3] In an actual implementation this list would be contained in a active data carrier attached to the pallet (see section 3.4.3).

[4] The *nominal* length of operations is always known. However the *actual* length may vary due to unexpected events during the operation.

in figure 3.1.



LM: Load Module
TM: Tool Module
PTM: Pallet Transport Module
OM: Operator Module
SM: Station Manager
PM: Program Module

Figure 3.1: Overview of the DFMS architecture.

In DFMS as many Station Managers are needed as there are machines. The Function Modules in the system are unique. In the following the functionality of each module is briefly defined:

- The Pallet Transport Module is responsible for the transport and storage of pallets in the system. Pallets are delivered to machines on request of the Station Managers;

- The Tool Module is responsible for the management of tools in the system. When tools are needed to perform operations the Tool Module is responsible for having the requested tools available at the requested time;

- The Program Module maintains all NC part programs that are required to perform the requested operations, and downloads these part programs on request to the Station Managers;

- The Load Module is used for introducing products into the system and for clamping and unclamping products on pallets;

- The Operator Module handles exception situations that might occur and offers a human operator the possibility to influence the system's operations.

In the next part of this section some important definitions necessary to explain the DFMS architecture will be given. In the subsequent sections the functions of the components will be discussed in more detail. In the original design documents the Yourdon specification methodology is applied to describe the architectural behavior [Ward 85a], [Ward 85b], [Ward 86]. In this chapter a description in *natural language* is presented. For an overview of the DFMS architecture see [Eve 86], [Kaas 87a], [Kaas 87d], [Bak 88a] and [Bak 88b].

3.1.2 Operation Classes

Definition of Operation Classes

In DFMS the operations that have to be performed on products are grouped in *Operation Classes*. The author defines an Operation Class as a set of operations that can be performed on a certain type of machine with a certain common toolset. To determine which Operation Classes exist and which operations belong to them, *cluster analysis* can be performed [Chan 86]. Cluster analysis is a generally applied method to form clusters of entities that have certain properties in common. In this case cluster analysis is used to find operations that use similar tools.

The cluster analysis procedure starts by describing the tool usage of all operations that can be performed on a particular machine type in a two dimensional matrix. The columns of the matrix represent tools, the rows represent operations. When an operation uses a tool a '*' is entered in the matrix, when the tool is not used a space is entered. An example of a (fictitious) unstructured matrix is shown in figure 3.2.

By now applying a cluster analysis algorithm such as MODROC [Chan 86] to this matrix, a structured matrix can be obtained. MODROC is a simple algorithm in which rows and columns are shifted in such a way that clusters are obtained. The clusters that are found are placed around the main diagonal. An example of a structured matrix that results from a clustering operation on the matrix of figure 3.2 is shown in figure 3.3. The cluster analysis procedure has to be repeated for each separate machine.

The cluster analysis example here presented is a near ideal case to show the principle. In many cases it will not be possible to form clearly separate

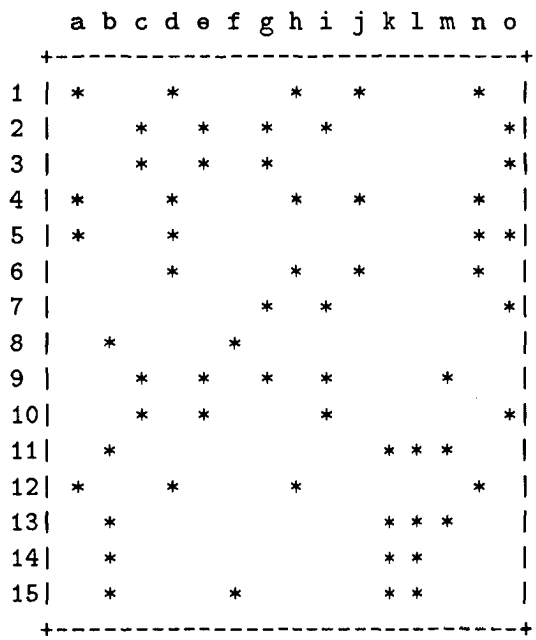


Figure 3.2: An example of an unstructured operations-tools matrix. Operations are labeled with numbers, tools are labeled with letters.

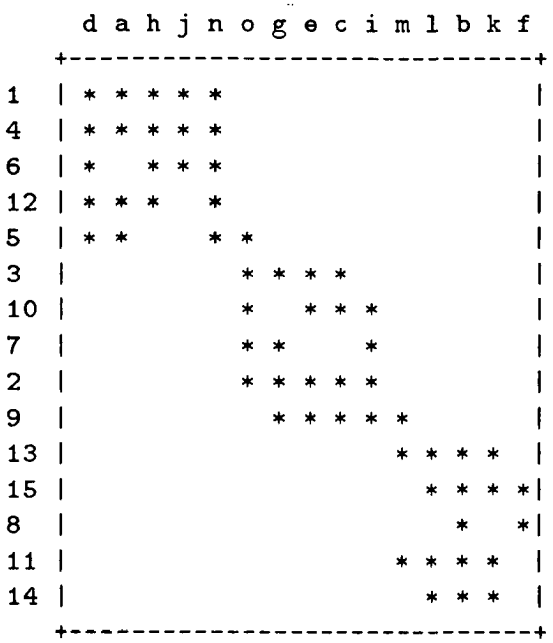


Figure 3.3: An example of a clustered operations-tools matrix. Operations are labeled with numbers, tools are labeled with letters. Three separate clusters can be recognized.

clusters as shown in the example. Sometimes special tools are used that are not shared by any operation. Sometimes tools are required for operations that belong to different clusters. Although these problems lead to less ideal clusters, they do not affect the principles of cluster analysis.

A distinction can be made between *tightly* and *loosely* defined Operation Classes. In a tightly defined Operation Class most tools are required for most operations. In a loosely defined Operation Class few tools are needed for most operations. Examples of loosely and tightly defined Operation Classes can be found in figure 3.4. In general, tightly defined Operation Classes are preferable above loosely defined Operation Classes, because a more efficient use is made of space in the tool store.

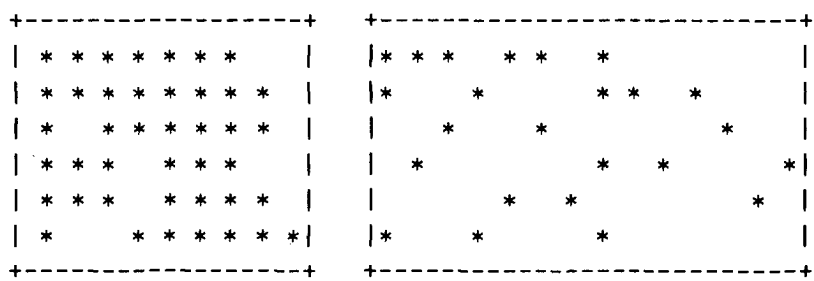


Figure 3.4: An example of a tightly defined Operation Class (left) and of a loosely defined Operation Class (right). For the tightly defined operation class only 9 tools are required. For the loosely defined Operation Class 14 tools are required.

Application of Operation Classes in DFMS

In DFMS machines *support* Operation Classes. When a Station Manager supports a particular Operation Class, the Station Manager is prepared to accept operations that are part of that Operation Class. In order to *efficiently* support an Operation Class the tool store of the machine has to contain a large number of tools required to perform the operations of the Operation Class. It is not necessary to demand that *all* tools that are needed have to be available in the local tool store. When a large number of the necessary tools is available, it can be guaranteed that to perform *any* operation defined in the Operation Class, at most a very small number of tools may have to be changed.

With the concept of Operation Classes there is no preference of one particular sequence of operations on a machine over another sequence of operations. The sequence in which the operations that belong to the same Operation Class are executed, is of no concern. Machine set up times can be completely avoided.

In figure 3.5 an example is given that shows which part of the tools have to be available and which tools might have to be delivered on request in order to have a machine efficiently support an Operation Class.

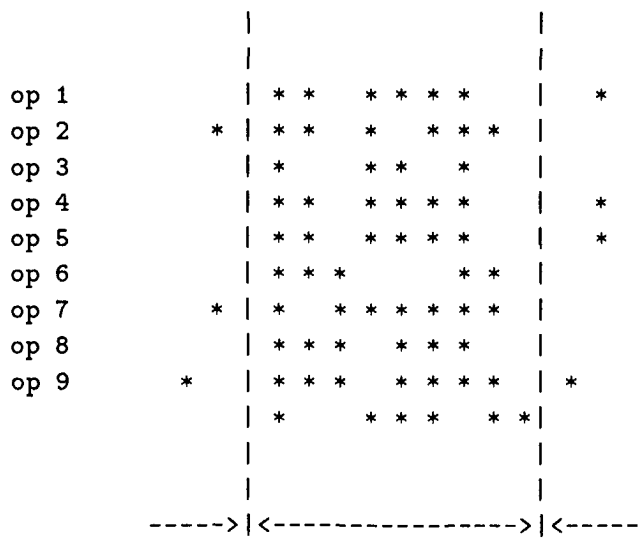


Figure 3.5: In order to allow for efficient production the tools between the two vertical lines have to be available in the local tool store. The tools outside the two lines are not used frequently and might be delivered on request.

When the size of the local tool store is large enough, a Station Manager may be able to support more than one Operation Class. When more Operation Classes are supported simultaneously, the tool store should be large enough to contain all commonly used, tools[5].

[5]When the same tools occur in two Operation Classes, the number of tools in the combination of two Operation Classes is smaller than the sum of the tools of the separate Operation Classes.

3.1.3 Product Scripts

Definition of Product Scripts

In DFMS, as in any FMS, a number of operations must be performed on each product. The process planning department has decided for each product which operations are required and in which sequence the operations have to be performed. This information is listed in the *Process Plan*. The first operation on the product is always a *clamp* operation in which the product is clamped on a pallet, the last operation is always an *unclamp* operation, in which the product is removed from the pallet.

For each product that is entered into the system a *Product Script* exists. The author defines the Product Script as a structure that contains the process plan for the product and some information describing the current state of the product. The current state consists of:

- the *Order Id* that holds the order identification of the product;
- the *Next Operation* variable that holds the line number of the operation that has to be performed next;
- the *Pallet Id* variable that contains the identification of the pallet on which the product is clamped;
- the *Priority* variable that gives the priority with which the product was entered into the system;
- the *History Log* that is a list of all events considered relevant.

An example of a product script is shown in figure 3.6. As can be seen from the example, it may occur that a product needs more than one clamp operation. This happens when the product needs to be machined in two different positions for which different *fixtures* are required. A combination of an unclamp and a clamp operation is used to change a fixture.

The Operation Classes and the operations that are mentioned in the product script can be of any type. Normal machining operations on machining centers and lathes occur frequently, but also product measurement and product cleaning operations do occur.

Application of Product Scripts in DFMS

The Product Script plays an important role in DFMS. The Product Script contains the information about products and about the operations that have

Process Plan

Nr	Operation Class	Operation	Fixture	Time
1	CLAMP	C1267	F100	20
2	OC9000	O9321	F100	50
3	OC7000	O7253	F100	30
4	UNCLAMP	U0000	F100	10
5	CLAMP	C1829	F134	27
6	OC2100	O2186	F134	11
7	UNCLAMP	U0000	F134	8

Status Info

Order	NR732726
Next Operation	3
Pallet Id	P54
Priority	1

History Log

Operation	Event
C1267	Performed by Station 23
C1267	Started at 11:03:23
C1267	Completed at 11:25:06
O9321	Performed by Station 11
O9321	Started at 12:22:45
O9321	Tool 35353 failure
O9321	Completed at 13:20:12

Figure 3.6: An example of a product script for a product that needs operations on two different fixtures. The first block contains the process plan, the second block contains some Status Information and the third block contains the History Log.

to be performed on them. As will be explained later, the Station Managers need the information in the Product Script to determine which machine will perform the next operation on a product.

The Product Script can be considered to be the *logical* representation of the *physical* product. The Product Script is transferred from Station Manager to Station Manager, just as the product is transferred from machine to machine.

3.1.4 Dispatching in DFMS

Unlike 'normal' FMS, a DFMS is not scheduled: no production plan is generated before the actual production starts. Each Station Manager maintains an Operation Queue in which operations are listed that will be executed by the machine. Only after a product has finished an operation on a machine will be decided on which machine the next operation has to be performed. When an operation on a machine is completed, the Station Manager updates the product script of the product[6]. The Next Operation counter is incremented and entries are added to the History Log. The Station Manager then reads the next operation that has to be performed on the product from the Product Script. The Operation Class of the operation is important dispatching information.

Each Station Manager knows which Operation Classes are supported by the Station Managers that are active in the system. All Station Managers that support the requested Operation Class are invited to make an offer to perform the operation. Each Station Manager replies with an offer that contains the earliest time that the product can be ready if the product is to be machined on its machine. The only consideration that influences the offer is the length of the current Operation Queue of the Station Manager that makes the offer. (The length of the Operation Queue can be expressed as the total time required to perform all operations in the queue or as the number of operations in the queue).

The operation is allocated to the Station Manager that made the best offer; a message that contains information about the operation to be performed is sent to the Station Manager to which the operation is allocated. The Station Manager then adds the operation to the tail of its Operation

[6]The Station Manager has access to the Product Script of the product that just finished its operation. The Product Script can be maintained in a programmable data carrier that is connected to the pallet. Each Station Manager is able to read and to modify Product Scripts.

Queue. Operation Queues are maintained strictly on a First-In-First-Out basis, no reordering of operations takes place under normal conditions. The operation on the new product will start as soon as all preceding operations have been finished.

3.1.5 Preparations for operations

The Station Manager that gets an operation allocated starts analyzing whether the tools required for the operation are expected to be available at the time the operation is expected to start. The Station Manager requests the Program Module to download the program data if they are not already available. The data contain the NC part program and a tool list describing the required tools. The NC part program will be executed by the CNC during the actual cutting and is not used for the early preparations. The tool list however enables the Station Manager to decide whether enough tool capacity is available. This is possible because the Station Manager has knowledge of the currently available tool capacity and about the tool requirements of the operations preceding the new operation. When a shortage is predicted, the Station Manager addresses a tool reservation request to the Tool Module. The Station Manager analyzes the tool requirements as soon as the operation has been added to the Operation Queue, because it might take the Tool Module some time before the tools can be made available from the central tool store.

Shortly before an operation is about to start the Station Manager determines if tool reservations have been made for the operation. If there are reservations made, the Tool Module is requested to deliver the tools. If no room is available in the machines tool store, unused tools will be exchanged to make room[7]. Under normal conditions the tools will be ready for delivery because they have been reserved some time earlier.

The Station Manager also requests the delivery of the pallet containing the product from the Pallet Transport Module. The time a pallet transport takes is assumed under normal conditions never to exceed a certain duration. When the pallet is requested before that time, delays because of late pallet arrival can be avoided.

When the pallet has arrived and when the preceding operation has been finished the new operation can be started. For the product on which just an operation has been completed, a new destination has to be found.

[7]Tools are delivered only just before they are required because otherwise tools might have to be removed that might still be needed.

3.2 Program Module

The Program Module is a Function Module that performs program services. All programs that are required to perform operations in the system are maintained by the Program Module. The programs consist of two parts: the actual NC part program that is executed by the CNC and a *Tool Requirements List*. The Tool Requirements List contains for each Tool Class the time the tool will be used. An example of a Tool Requirements List is shown in figure 3.7.

Tool Requirements List

Tool Class	Cutting Time
T12636300	20
T14244400	34
T15342400	17
T52666200	28
T59828000	31

Figure 3.7: An example of a Tool Requirements List.

Station Managers can request the Program Module to download programs when a new operation has been added for which the required program is not available. To request the downloading of a program the Station Manager sends a *Program Delivery Request* message in which the name of the requested program is specified. The Program Module replies by sending a *Program Delivery Reply* message that contains the NC part program as well as the Tools Requirements List. Examples of these messages can be found in figures 3.8 and 3.9.

Program Delivery Request

Operation	O9321

Figure 3.8: An example of a Program Delivery Request message.

The situation that a program is not available must be avoided. In order to prevent this from happening, the availability of the required programs is checked beforehand when a new order is accepted by the system. (In

Program Delivery Reply

Operation	O9321

Tool Class	Time
T12636300	20
T14244400	34
T15342400	17
T52666200	28
T59828000	31

NC Part Program

```
N001 G01 X353.45 Y754.885 Z847.999 S0945 M03
N002 G01 X545.5885
N003 G01 Y 500.900
.....
```

Figure 3.9: An example of a Program Delivery Reply message.

addition, programs cannot be deleted as long as they can be requested by a Station Manager). The module responsible for accepting orders (the Load Module) sends a *Program Availability Request* message to the Program Module in which the names of the required programs are specified. The Program Module checks for the availability of the programs and sends an positive or negative *Programs Availability Reply* message.

The NC part programs and Tool Requirements Lists that are maintained by the Program Module are generated outside the system by the process planning and NC part programming department. Only tested and released programs are made available to the system. If the need should arise to change programs once they have been introduced, this can be done locally on the CNC of the machine. The changed program can than be uploaded to the Program Module. The new version of the program will have to be authorized by the programmer before it is allowed to replace the original version.

3.3 Tool Module

The Tool Module is a function module that performs tool services [Roë 87b]. Tools can be stored in the local tool stores of the machines or they can be stored in a *central tool store*. The central tool store has its own controller, which manages the tools that are currently in the central tool store. The Tool Module is the only component that is aware of the location and status of *all* tools. A *Tool Handling System* is used to prepare tools and to transfer them between tool stores. The Tool Module controls the operations of this Tool Handling System.

3.3.1 Interface with the Station Managers

The Station Managers in the system determine which tools have to be ordered in order to be able to perform the allocated operations. Tool ordering is performed in two stages: *reserving* and *delivering*. When a Station Manager predicts a shortage of tools in one or more Tool Classes a *Tool Reservation Request* message is sent[8]. The request contains the name of the machine, the order for which the tools will be used, the time when the tools are needed, the missing tools in the particular Tool Classes and the

[8] A Tool Class (short for tool classification), represents a *type* of tool.

requested Tool Time for each Tool Class[9]. An example of the message is shown in figure 3.10.

Tool Reservation Request

Machine	12
Order	NR142525
Request Time	210

Tool Class	Time
T14244400	40
T52665300	28
T51767200	31

Figure 3.10: An example of a Tool Reservation Request message. Tools are reserved for order NR142525 that will be performed on machine 12.

Per requested Tool Class one or more tools are allocated to meet the requested Tool Time. The Tool Module tries to allocate existing, not already allocated tools to the machine. If not enough tools are available in the central tool store, the Tool Module generates orders for the Tool Room operators to provide new tools. These new tools have to be assembled and their exact dimensions have to be measured before they can be used[10]. If tools were already reserved for that particular order on the machine, the newly reserved tools are added to the reservation.

Once the tools are allocated to a machine, they cannot be used anymore by other machines. The tools are not reserved for a particular period of time, they are reserved until the Station Manager explicitly returns the tools or releases them. To prevent tools from being allocated too long, Station Managers only reserve tools for operations that are expected to start within a certain time.

The Tool Module sends a *Tool Reservation Reply* message to the Station Manager to inform which tools have been reserved and how much time each tool can be used. Whereas in the request, *Tool Classes* and *required* tool times are given, in the reply *Tool Identifications* and *actual* tool times are

[9]The time when the tools are needed is expressed as a *delta time*, as a number of time units from the current time.

[10]In section 3.3.3 the topic of tool data is further discussed.

given[11]. The time that the tools will be available is also given. (This might be later than the requested time, when tools have to be assembled and when the assembly capacity is not enough). An example of a Tool Reservation Reply is shown in figure 3.11.

Tool Reservation Reply

Machine	12
Order	NR142525
Available Time	220

Tool Id	Time
T14244427	20
T14244428	20
T52665323	40
T51767211	60

Figure 3.11: An example of a Tool Reservation Reply message. Note that two tools have been used to provide sufficient Tool Time for Tool Class T14244400. The tools are available 10 time units later than was requested.

The Station Manager may accept or reject the offer made by the Tool Module. When the tools become available much later than the time the operation is expected to start, the Station Manager can drop the operation and start processing the next operation in its Operation Queue. In that case the Tool Module receives a *Tool Reservation Cancel* message, which results in the cancellation of all tools that have been reserved on behalf of the machine that sent the message. The tools that were reserved become available again and can be allocated to every machine that requests tool capacity. The machine that cancelled the tool reservations will start to analyze the tool requirements of the operations that were left in the Operation Queue. This results in a new set of Tool Reservation Requests. It was necessary to cancel the reservations in the first place, because the tools that may have been reserved for the next operation will be now be needed much earlier than was agreed upon. This could cause problems with the delivery of tools as will be shown later.

[11]The format of a Tool Class and a Tool Identification are identical. A Tool Class is specified as a number of which the last two digits are 0. The Tool Identifications of tools of the same Tool Class differ only in the last two digits.

Shortly before the reserved tools are actually needed, the Station Manager will request the delivery of the tools by sending a *Tool Delivery Request* message. This message contains the Order Identification of the operation for which the tools were reserved and the time the Station Manager is prepared to wait for the delivery of the tools. See figure 3.12.

Tool Delivery Request

Order	NR142525
Wait Time	10

Figure 3.12: An example of a Tool Delivery Request message.

When the Tool Module is able to deliver the Tools within the specified period, the Tool Module sends an affirmative *Tool Delivery Reply* message. When all reserved tools are available, the tools will be transported. When some of the reserved tools are not yet assembled, the Tool Module waits until the assembly is ready and then transports the tools.

When the Tool Module is not able to deliver the tools, a negative *Tool Delivery Reply* message is sent. All tools that have been reserved for the machine will be cancelled.

The situation that the Tool Module has to send a negative Tool Delivery Reply has to be considered an exception. Assuming that no Station Manager ever requests delivery before the time the tools were reserved (and they never do), the Tool Module has not been able to realize the offer it made at reservation time.

In the case that a tool breaks during an operation and that no replacement tool is available at the local tool store, the Station Manager sends an *Emergency Tool Delivery Request* message. An example of this message is shown in figure 3.13

When an Emergency Tool Delivery Request is received, the Tool Module checks whether it is possible to deliver the requested tool within the specified time. If necessary the tool will be assembled. A positive or negative *Emergency Tool Delivery Reply* message is sent back to the machine.

3.3.2 Interface with the Tool Handling System

The Tool Module controls the *Tool Handling System* which is responsible for physically manipulating tools. The Tool Handling System consists of two

Emergency Tool Delivery Request

Machine	12
Tool Class	T12873300
Tool Time	20
Wait Time	10

Figure 3.13: An example of an Emergency Tool Delivery Request message.

components:

- **the Tool Preparation System**
  Before tools can be used in the system some preparations have to be performed on them. The tools have to be assembled and the exact tool dimensions have to be measured. These tasks are performed by operators that use special tool measuring equipment;

- **the Tool Transport System**
  Tools have to be transported from the central tool store to the local tools stores of the machines, when delivery is requested by the Station Managers. Tools that are no longer used by the machines are returned.

No detailed specifications have been made for the interaction between the Tool Module and the Tool Handling System. The interface between the Tool Module and the Tool Preparation System has to contain at least the following messages:

- Prepare Tool Class 'Tool Class-Id';
- Tool Class 'Tool Class-Id' ready.

The interface between Tool Module and Tool Transport System has to contain at least the following messages:

- Transport tools for OrderId 'OrderId' to machine 'MachineId';
- Tool 'Tool Id' returned from system.

3.3.3 Handling of Tool Data

When a NC part program is developed, the programmer assumes nominal tool dimensions. The nominal tool dimensions are fixed and are known

for each tool type (Tool Class). The actual tools that are used however, have dimensions that differ slightly from the nominal dimensions. Special equipment is used to measure the actual dimension of each tool before it is actually used in the system. The CNC of the machine on which the tool is used, makes corrections for the actual tool dimensions.

Apart from the tool dimension data, the remaining tool life time has to be maintained for each tool. When a tool is used on a machine the CNC subtracts the time the tool has been used from the tool life time. Tools that do not have any tool life time left can not be used any more and have to be removed from the system.

In older FMS the tool data are maintained centrally by the central control system and are downloaded to the machines where the tools are transported to. A new system has recently become available in which the tool data are stored in a programmable data carrier in the tool holder [Mur 87]. In this way tool and tool data are never separated. Advantages of this method as opposed to the older method are:

- **Reliability**
  Because tool and tool data are never separated the chance that the wrong data is used with the wrong tool has become negligibly small;

- **Simplicity**
  The FMS control system is not burdened with an additional task of maintaining and downloading tool data;

- **Robustness**
  When the FMS control system crashes it will not be necessary to determine the actual tool dimensions of each tool again. The data will not be lost, because it is available in the tool holder.

The disadvantage of having programmable data carriers in the tool holders is the higher costs of the toolholders and the necessity to have tool data readers/writers at each machine.

In the architectural description of the Tool Module it is assumed that the tool holders are equipped with programmable data carriers. No special functionality to maintain and download tool data has been defined.

If for some reason the use of programmable data carriers in the tool holders is not acceptable, the Tool Module has to perform the additional task of maintaining, downloading and uploading the tool data.

3.3.4 Starting Up the Tool Module

The Tool Module has to have knowledge of all tools that are currently in the system. When the Tool Module is started, all DFMS components that have a local tool store send information about their tools to the Tool Module. In section 3.8.3 the start up procedure of the Tool Module is discussed in detail.

3.4 Pallet Transport Module

The Pallet Transport Module is the Function Module that performs pallet transport services [Roë 87a]. Pallets can be stored in the pallet buffers of the machines or they can be stored in a *system pallet buffer*. Each Station Manager knows which pallets are contained in the pallet buffer of its machine. The system pallet buffer has its own controller that manages the pallets that are currently in the system pallet buffer. The Pallet Transport Module is the only component in the system that is aware of the location of *all* pallets.

3.4.1 Interface with the Station Managers

The Station Managers are responsible for the pallet management in the system. Shortly before a new operation is to start a Station Manager issues a command to the Pallet Transport Module to deliver the pallet on which the product is clamped. When the Station Manager decides that not enough space is left to receive a new pallet, the Pallet Transport Module is requested to remove one of the pallets that is currently in the pallet buffer of the machine. Figure 3.14 shows an example of a *Pallet Transport Request* message that is issued by a Station Manager to the Pallet Transport Module.

Pallet Transport Request

GetPallet	P43
StorePallet	P12

Figure 3.14: An example of a Pallet Transport Request message. Pallet P12 has to be removed and pallet P43 has to be delivered.

The Pallet Transport Module adds the transport request to a queue of pending requests. This queue is processed on a First Come First Served ba-

sis. Not all transport requests can be executed immediately. For one reason the number of transport vehicles is limited. If all transporters are occupied, the transport request is kept pending until a transporter is available. Another reason why a transport request may not be executed immediately is in the case of *dependent* transport requests. Assume that one of the transporters is executing a transport request for Station Manager A in which a GetPallet P12 and a StorePallet P43 are specified. If now a request is received from Station Manager B to deliver pallet P43, a problem arises: the position of pallet P43 cannot be determined. The Pallet Transport Module does not know where to find the pallet. To prevent this problem, transport requests are only executed when they are *independent* from each other. A transport request that would cause a dependency is kept waiting in the queue until the dependent transport request is completed. The situation is then reevaluated[12].

When a transporter is available and when independent transport requests are waiting in the queue, the transporter is allocated to the oldest independent transport request. When more than one transporter is available, the most suitable transporter is chosen. Criteria to choose a transporter are:

- The distance of the transporter to the position of the pallet that is to be retrieved. The shorter the distance the more suitable the transporter. This assumes that a model of the FMS layout is present in the Pallet Transport Module;

- The number of pallet positions on a transporter. A transporter that has two positions is more suited than a transporter with one position to execute a request in which both a pallet have to be delivered and stored.

The Pallet Transport Module generates a number of commands that will be executed by the transporter. The commands are passed one by one to the *Pallet Transport System* that is responsible for executing them.

[12] A more intelligent solution would be to *combine* the two transport requests. However, this complicates the design of the Pallet Transport Module considerably and the advantages in terms of system performance are negligible.

3.4.2 Interface with the Pallet Transport System

The Pallet Transport Module has a simple interface with the Pallet Transport System[13]. Three commands can be passed to the Pallet Transport System:

Move	'Vehicle Id'	'Position'
Get-Pallet	'Pallet Id'	
Store-Pallet	'Pallet Id'	

After a command has been accepted by the Pallet Transport System, the Pallet Transport Module waits for a status reply. For each transporter at most one command can be pending at any time. The following status messages can be received:

Success	'VehicleId'	
Vehicle-Error	'VehicleId'	
Position-Error	'VehicleId'	'Position'

When a **Success** message has been received, the next command that has been generated for the transporter is passed to the Pallet Transport System. If no more commands are left the transporter is available to process a new transport request.

If a **Vehicle-Error** message has been received, the Pallet Transport System has detected a failure of the vehicle that made it impossible to deliver the pallet to the machine of the Station Manager that requested the pallet. The requesting Station Manager is informed that it is not possible to deliver the pallet by means of a *Pallet Transport Error* message. The syntax of this message is equal to the syntax of a Pallet Transport Request message (figure 3.14).

If a **Position-Error** message has been received, the Pallet Transport System detected a failure of the local pallet buffer of a machine. No pallets can be transferred to that position until this problem is fixed. All pending or current transport requests that require a pallet that is blocked on the position will be deleted. Pallet Transport Error messages are sent to all Station Managers for which a transport request was aborted.

[13]The interface here presented is an example. In real implemented systems other (probably more elaborate) commands and status messages will occur.

3.4.3 Handling of Pallet Data

The Product Script plays an important role in the DFMS system. Station Managers need to have access to the Product Scripts in order to be able to decide which operations have to be performed on the product. Just as there is an advantage in not separating tool data from the tool, there is an advantage in not separating the Product Script from the product. The pallets are therefore assumed to be equipped with programmable data carriers that contain the Product Scripts. When the product is clamped on the pallet, the Product Script is written to the data carrier. Each Station Manager has to be equipped with a device that is able to read and write the Product Script.

Advantages of having the Product Script fixed to the pallet are higher reliability, robustness and simplicity (see section 3.3.3). A disadvantage is the higher costs of the pallets and the necessity to have read/write equipment on each machine.

If for some reason it is not possible to use pallets equipped with programmable data carriers, the Pallet Transport Module will have the additional responsibility of transferring the Product Scripts between Station Managers. In this case the Pallet Transport Module will send a *Product Script Update* message to the Station Manager just after a pallet has arrived at the machine. The message contains the complete Product Script of the product on the pallet. After the Station Manager has completed the operation on the product and after the Product Script has been updated, the Station Manager sends a *Product Script Update* message back to the Pallet Transport Module.

3.4.4 Starting Up the Pallet Transport Module

In order to be able to perform its functions, the Pallet Transport Module has to have knowledge of the system. The following data are required by the Pallet Transport Module:

- the number of pallets;
- the location of each pallet;
- layout information.

The layout information is only required when the Pallet Transport Module tries to optimize the usage of transport vehicles. The layout information

is fixed for a system and does not have to be updated when the system starts[11].

When the Pallet Transport Module is started all DFMS components that control a pallet buffer send information about the pallets in their buffer to the Pallet Transport Module. In section 3.8.4 the start up procedure of the Pallet Transport Module is discussed in detail.

3.5 Operator Module

The Operator Module has three separate functions:

- The Operator Module is used for error recovery;
- The Operator Module provides an operator interface to the DFMS system;
- The Operator Module is used for Management Information System purposes.

The three functions will be discussed separately. For a detailed description of the Operator Module functionality see [Roëll 87c].

3.5.1 Error Recovery

The Operator Module is a *special* Function Module. Unlike the other Function Modules, its services are never requested during normal system operations. Only when an operation cannot be performed by a Station Manager is the Operator Module needed. The Operator Module has an Operation Queue, just as a Station Manager, in which operations are listed that could not be performed. When a Station Manager detects a problem, control over the product is passed to the Operator Module by means of an *Operation Reject* message (figure 3.15).

The rejected operation is added to the Operation Queue and it is up to the (human) system operator to decide how to handle the error situation. A number of situations can cause a Station Manager to reject an operation. These situations and the reaction of the Operator Module to them are discussed below.

[11] The physical Pallet Transport System *always* requires layout information. A separate start up procedure for the Pallet Transport System will have to provide this information.

Operation Reject

Machine	12
Order	NR2873300
Operation Class	OC3400
Operation	O3410
Pallet Id	p66
Reject Reason	Could not reserve tools

Figure 3.15: An example of an Operation Reject message.

- If a tool is requested for an operation and the Tool Module is not able to have the tool ready at the requested time, a *Tool Reservation Exception* occurs. The Station Manager may decide not to wait for the tool to become available, but to proceed with the next operation. This exception normally only happens when the tool is not in store and has to be assembled and when there is temporarily too little capacity to assemble the tool before the requested time. In this case there is no structural problem and the operation can be reintroduced into the system later. It is also possible that the requested tool is not available and cannot be assembled either. When this occurs it is not possible to perform the requested operation and it is necessary to remove the product from the system;

- Even though at reservation time it seemed possible for the Tool Module to have the requested tools available, it can occur that the tools are not available when the Station Manager requests delivery: a *Tool Delivery Exception* occurs. This normally only happens when the Tool Room operators, due to various reasons, have assembled less tools then was expected. The exception is handled similarly to a Tool Reservation Exception;

- When the Pallet Module does not succeed in transporting a pallet to the machine where an operation is expected to start, a *Pallet Delivery Exception* occurs. This situation occurs for example when due to a pallet buffer malfunction a pallet cannot be removed from its original position. There is no immediate action that can be taken by the Operator Module. (The error that caused the problem has to be solved by maintenance personnel). The operation is kept in the Operation

Queue until the problem is solved, whereafter the product can be reintroduced into the system;

- A tool breakage during an operation may cause damage to the product. If the operation is aborted a *Product Damage Exception* occurs. Control over the product is passed to the Operator Module. The product will have to be removed;
- When a product is introduced into the system, the Load Module checks whether all required Operation Classes are supported. However, when a machine is removed from the system, the situation can occur that an Operation Class needed by a product already in the system, is not supported any longer. In that case it is not possible to reallocate the product to a new machine: a *Product Allocation Exception* occurs. The Operator Module keeps the operation in its Operation Queue until the Operation Class is supported again;
- When a Station Manager has to be removed from the system, the operations that were already accepted by the Station Manager can not be performed. This situation is called a *Machine Down Exception*. The Station Manager passes control over these operations to the Operator Module. The Operator Module tries to reallocate the operations as soon as they are added to its Operation Queue. Because a Station Manager has been removed, it may occur that the required Operation Class is not supported any longer: a *Product Allocation Exception* is the result.

3.5.2 Operator Interface

There is no real technical need to present DFMS with an operator interface. However, because human operators may have to work in the system it is important to help them understand what is going on. The Station Managers and each Function Module allow the operator to inspect the operations of the module. The operator interface allows a system operator to have a more global view of the system. The Operator Module does not maintain any global data. When an operator wishes to perform a query, the data have to be gathered throughout the system. For example, if the operator wishes to know which Station Manager is currently responsible for a particular product the Operator Module has to sent requests to all modules to upload their Operation Queues. After all Operation Queues have been received the requested data can be presented to the operator.

The queries that can be performed by an operator have not been defined in detail. They can be implemented according to the requirements of a particular implementation. Some possible operator functions are:

- Show the Operation Queue of a particular Station Manager;
- Show the positions of all pallets in the system;
- Show performance statistics for each machine;
- Show problems and exceptions detected by each Station Manager.

As stated before there is no technical requirement to have an elaborate operator interface. Care has to be taken that only data that are meaningful will be presented.

3.5.3 Management Information System

In chapter 2 the Management Information System functionality has been discussed. The modules in the DFMS architecture are responsible for storing all significant data about their behavior. The data that is generated has to be gathered in order to generate reports about the system behavior. It is proposed to let the Operator Module be responsible for this information gathering. The necessary functionality and messages have not yet been defined.

3.6 Load Module

The Load Module is a complicated Function Module. It performs two functions: introduction of new products into the system and the supervision over the clamp stations. For a detailed specification of the Load Module functionality see [Kaas 87b].

3.6.1 Introduction of new products

The Load Module receives orders from the higher level scheduling system. This higher level system presents DFMS at regular intervals with a set of orders that have to be processed in the next scheduling period. A scheduling period is typically a day or a week long. There is no preference for a particular sequence within the scheduling period for the products to become available, as long as they are ready at the end of the period.

The Load Module checks new orders to make certain they can be processed in the system. The Load Module inspects the list of currently supported Operation Classes to prevent introducing products that cannot be processed completely, because one or more required Operation Classes are not supported. In addition the Program Module is requested to confirm the availability of the required NC part programs by means of a *Program Availability Request* message (see page 87). When one of the conditions is not met, the product is rejected and the higher level scheduling system is informed.

The first operation on every product is a *clamp operation* in which the product is clamped on a pallet by means of a product specific *fixture*. The Load Module maintains a list of available pallets, available fixtures and of products waiting to be clamped. Newly accepted products are added to the list of products waiting for a clamp operation. Usually all pallets available will be used to clamp products. Due to limitations in the number of available fixtures, the choice of products to clamp is limited. New pallets and fixtures become available when products that have had all their operations are removed from their pallets at the clamping stations.

The Load Module employs a strategy to determine which product has to be clamped. As soon as a combination of product, pallet and fixture has been found, a clamp order is added to the Operation Queue of the Load Module. The strategies used by the Load Module can be very simple or quite sophisticated. The influence of some strategies will be discussed in the chapter on DFMS system performance (chapter 6).

3.6.2 Supervision of Clamping Stations

The Load Module supervises the clamping stations on which both clamping and unclamping operations are performed. The Load Module maintains an Operation Queue just as Station Managers do. This Operation Queue contains operations that can be performed exclusively on the clamping stations of the Load Module. Clamping stations are stations on which pallets can be stored and that have interfaces with the pallet transport system. The Load Module controls a number of these clamping stations. This is different from the situation of Station Manager where the operations are performed by only one machine.

The Load Module issues commands to the clamping stations to clamp or unclamp products. As soon as the clamping station reports the completion of a clamp or unclamp operation, the Load Module issues a new

command from its Operation Queue. When the Operation Queue is empty, the clamping station will be idle until a new operation is added. The Load Module requests the delivery of pallets that are required for operations that are expected to start shortly.

The unclamp operations are added on request of Station Managers. The last operation that has to be performed on any product is an unclamp operation. To a Station Manager this seems an operation just like any other. Through the normal product reallocation process the control over the product is passed to the Load Module, because the Load Module happens to be the only one that supports the unclamping Operation Class. The operation is added to the Operation Queue of the Load Module and is performed as soon as the preceding operations have been performed and a clamping station becomes available.

3.6.3 Starting Up the Load Module

The Load Module has to have knowledge of the available pallets and fixtures in the system. For the Load Module it is important to know:

- the number of pallets;
- which fixtures are clamped on the pallets;
- which fixtures are still available.

When the Load Module is started all DFMS components that have a local pallet buffer send information about the pallets and fixtures on those pallets to the Load Module. The information about the still available fixtures is provided by the operators of the clamping stations. In section 3.8.5 the start up procedure of the Load Module is discussed in detail.

3.7 Station Manager

Each machine in DFMS is equipped with a Station Manager. The Station Managers together are responsible for handling the orders to be processed in the current scheduling period. The functionality of the Station Manager is discussed in the coming sections. For a detailed specification see [Kaas 87c].

3.7.1 Processing operations in the Operation Queue

The Station Manager maintains an Operation Queue in which all operations are listed that will be performed by the machine. The operations are processed in the sequence that they were entered to the Operation Queue. The

Operation Queue contains all data to enable the Station Manager to make preparations for the operation. An example of an Operation Queue is shown in figure 3.16.

Operation Queue

Operation	Operation Class	Operation Time	Pallet Id	Fixture
O7002	OC7000	20	P12	F321
O7034	OC7000	45	P23	F321
O7102	OC7100	38	P02	F073
O6355	OC6300	26	P11	F938
O7074	OC7000	54	P09	F321
O7091	OC7000	22	P31	F553

Figure 3.16: An example of an Operation Queue.

When the current operation finishes, the Station Manager increments the *Next Operation* counter in the Product Script. The contents of the CNC tool memory is uploaded to the Station Manager to enable the Station Manager to base its tool requirements calculations on accurate data.

The Station Manager then tries to start the first operation in the Operation Queue. This operation can be started when the pallet is available in the pallet buffer of the machine. Even when not all tools required are available, the operation can be started all the same when the Tool Module acknowledged the delivery of the missing tools. It is not necessary to wait for the arrival of the tools.

When for the current operation a tool is needed that is not yet in the local tool store of the machine, the Station Manager checks whether the arrival of the tool is expected[15]. If the tool is expected nothing needs to be done. The operation is suspended until the tool arrives. When the tool was not expected, an *Emergency Tool Delivery* message is sent to the Tool Module. If it is not possible to deliver the tool within a reasonable time, a *Tool Delivery Exception* occurs. The operation is aborted and the control over the product is passed to the Operator Module.

Some time before the operation is expected to finish the Station Manager

[15] A tool is 'expected' when it is one of the tools that have been reserved for the current operation. The Station Manager has already requested the Tool Module to deliver the tools and the Tool Module has acknowledged the delivery.

starts making preparations for the next operation to be performed[16]. These preparations will be discussed in section 3.7.5.

3.7.2 Allocation of operations

The product that just finished its operation on a machine needs a next operation. (The last operation is always unclamping, which is performed under control of the Load Module). The Station Manager inspects the Product Script of the product to determine which operation is required; specifically the required Operation Class is of importance. Each Station Manager knows which Operation Classes are supported by all elements in the DFMS system. These data are maintained in the *World Model* (figure 3.17).

World Model

Operation Class	Manager1	Manager2	Manager3
OC12	1		
OC13	1	3	4
OC14	2		
OC15	1	4	
OC16	4	5	

Figure 3.17: An example of a World Model. The data of only three Station Managers are shown.

The Station Manager can easily determine which Station Managers do support the requested operation. To all Station Managers that support the operation a *Machine Capacity Request* message is sent. This message does not contain any data; the reception of the message just prompts the Station Managers to reply by sending a *Machine Capacity Offer* message in which the suitability of the Station Manager to perform the operation is expressed[17]. An example of a Machine Capacity Request message is shown in figure 3.18). The reply on the messages contains the earliest time that the operation could be started, if the operation would be allocated. The Station

[16]The amount of time before the end of an operation is a system parameter that has to be set for a particular system.

[17]Basically a simple broadcast to all Station Managers in the system would suffice. In that case the requested Operation Class would have to be specified in the message. However, broadcasting messages has some disadvantages (see section 3.8.1).

Manager that can start soonest is considered to be the best candidate to perform the operation. By applying this simple *dispatching rule* a uniform balancing of the workload is attempted.

Machine Capacity Offer

Start Time	120

Figure 3.18: An example of a Machine Capacity Offer message.

When all Machine Capacity Offers have been received, the Station Manager that issued the best offer gets the operation allocated. A *Machine Capacity Allocation* message is sent to that Station Manager (see figure 3.19). The Machine Capacity Allocation message contains all data necessary to enter the operation in the Operation Queue.

Machine Capacity Allocation

Operation	Operation Class	Pallet Id	Fixture
O7091	OC700	P31	F553

Figure 3.19: An example of a Machine Capacity Allocation message.

3.7.3 Downloading programs

As soon as a new operation has been added to the Operation Queue, the Station Manager checks whether the required program information is available in the CNC controller. The program information consists of the NC code that will be executed by the CNC and of the *Tool Requirement List* (see figure 3.7). When the program is not already available, the Program Module is requested to download the program by means of a *Program Delivery Request* message (figure 3.8). The Program Module sends back a *Program Delivery Reply* message (figure 3.9) containing the requested information.

3.7.4 Reserving tools

When the Tool Requirements List of the new operation is known, the Station Manager can predict whether enough tool capacity will be available at the

time the operation is expected to start. To be able to predict this, the CNC is requested to upload its tool memory to the Station Manager each time an operation has completed. The tool capacity required by each operation is subtracted from the available tool capacity. If there is already tool capacity reserved for an operation, this capacity is added to the predicted available capacity. Whenever the predicted available tool capacity for an operation is too small to perform the operation, the Station Manager requests the Tool Module to reserve the missing tools by sending a *Tool Reservation Request* message in which the required tool capacity is listed. The Tool Module replies by sending a *Tool Reservation Reply* message in which the exact tool capacity allocated to the Station Manager is specified. Examples of the two messages can be found in the section on the Tool Module (figures 3.10 and 3.11).

The predicted available tool capacity is analyzed again each time an operation completes and the actual tool capacity is uploaded from the CNC. It is possible that the actual tool capacity proves to be less than predicted, for example because of tool failures during the operation. In this case it is necessary to order additional tools for operations for which already tools were reserved. The Tool Module just adds the additional reservation to the reservation that was already pending for the operation.

The Station Manager reserves tools for operations as soon as an expected shortage of capacity is detected. This is to allow the Tool Module to prepare tools if they are not readily available. However, as soon as tools are reserved for an operation by the Station Manager, those tools cannot be allocated to other Station Managers. Therefore, in order to prevent tools from being allocated much longer then necessary, the Station Manager only analyzes operations that are expected to start within a certain time.

It is important to realize that the total number of tools that the Station Manager may have reserved for operations to be performed in the future can be more than can be stored in the local tool store at one time. However, there is no problem as long as all tools required for one particular operation can be stored in the local tool store at the same time. When new tools are actually delivered the Station Managers decides which tools will be exchanged.

3.7.5 Delivery of tools and pallets

Shortly before the current operation completes the Station Manager starts preparations for the next operation to be performed[18]. If there are tools re-

[18] When the Operation Queue is empty, no preparations are necessary.

served for the operation, the Tool Module is requested to deliver them: the Station Manager sends a *Tool Delivery Request* message (see figure 3.12). The Tool Module replies by sending a *Tool Delivery Reply* message, a positive reply if the tools can be delivered or a negative reply if this is not possible. If the tools cannot be delivered, the first operation in the Operation Queue will be skipped and preparations for the next operation will be started[19]. However, usually the Tool Module will be able to deliver the tools. The Station Manager decides at this time which tools will be exchanged to make room for the new tools. Tools are chosen that do not have any tool life time left or that will not be used in the near future.

After delivery of the tools has been confirmed, the Station Manager sends a *Pallet Transport Request* message to the Pallet Transport Module, in which the required pallet is specified. If the local pallet buffer does not have enough room to hold the new pallet, the Station Manager requests the Pallet Transport Module to remove one of the pallets in the local pallet buffer. The Station Manager chooses a pallet that contains a product that already has undergone its operation on the machine. An example of a Pallet Transport Request in which both a pallet to get and a pallet to store are specified is shown in figure 3.14.

The new operation will be started as soon as the pallet has arrived or when the previous operation has been completed, whichever comes later.

3.7.6 Interface with the CNC

Apart from communicating with other DFMS components, the Station Manager controls the operations of a machine by issuing commands to the CNC of the machine. The commands and messages that are exchanged between Station Manager and CNC are to some extent implementation dependent. Some commands that the Station Manager has to be able to give to the CNC are:

- start an operation;
- upload the tool memory of the CNC;
- upload the pallet memory of the CNC;
- download a program to the CNC.

The CNC has to report at least the following events:

[19]The control over the operation that could not be performed is handed over to the Operator Module.

- an operation just finished;
- a pallet arrived;
- new tools arrived.

The commands and messages listed here are just a selection of the necessary commands and messages. A detailed and complete overview of the interface between Station Manager and CNC is discussed in the chapter on Implementation of DFMS (chapter 4).

3.8 Adding and removing DFMS components

In the previous sections the functionality of a *static* DFMS has been discussed. However, three situations exist in which the configuration of DFMS changes:

- during a system initialization;
- during a system shutdown;
- when in a running system a machine is added or removed.

It is an important feature of the DFMS architecture that it is possible to add or remove machines when the system is active, without disrupting system operations. Special procedures are required to add or remove components from DFMS. These procedures will be discussed in this section. Before these procedures are discussed however, the role of the network that enables the sending of messages between DFMS components will be explained.

3.8.1 The DFMS Network

The DFMS components are connected through a *DFMS Network* that enables the components to communicate with each other[20]. The DFMS Network enables each DFMS component to send messages to all other components. Messages are sent from one specific destination to another specific destination. Broadcasting of messages is not allowed[21].

The DFMS components that have been discussed in the previous sections do not connect directly to the network: a *NetWork Interface* serves

[20] The term 'network' has to be interpreted in the architectural sense. The network is just another DFMS component of which the functionality has to be specified.

[21] Broadcasting is not supported in the MAP application layer specification according to RS511. In order to maintain compatibility with this standard, broadcasting of messages is not used in the DFMS architecture.

as a buffer between network and DFMS component. The Network Interface performs all network specific operations, the details remain hidden from the DFMS component. In figure 3.20 the relation between DFMS components and network interfaces is shown.

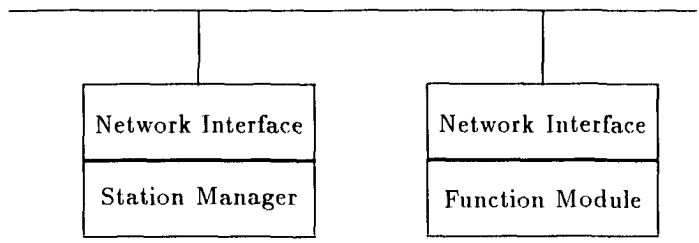


Figure 3.20: The relation between network interfaces and DFMS components.

The Network Interface of each component connected to the DFMS Network has a *Network Address*. In order to send a message to a destination, the Network Address of the Network Interface of that destination has to be known. Each Network Interface therefore needs to know the Network Addresses of all other Network Interfaces. A *Network Directory Server* is used to maintain the Network Address of all components in the system.

When a new component is added to the network, the component sends a *New Machine* message to the Network Directory Server. The message contains the Network Address of the new component and the name of the DFMS component. An example is shown in figure 3.21.

New Machine

DFMS Name	Network Address
Station Manager 1	DFMS221

Figure 3.21: An example of a New Machine message.

The Network Directory Server receives this messages and updates its Network Address information. Next this update is sent by means of a *Network Update* message to all components that are all already known[22]. An

[22]The Network Update message contains the *complete* network model. It would have

example of this message is shown in figure 3.22.

Network Update

DFMS Name	Network Address
Station Manager 1	DFMS221
Station Manager 2	DFMS341
Station Manager 3	DFMS131
Station Manager 4	DFMS451
Tool Module	DFMS006
Pallet Transport Module	DFMS006

Figure 3.22: An example of a Network Update message.

The *New Machine* and *Network Update* messages are processed by Network Interfaces, not by the DFMS components. In this respect they differ from the messages discussed in the previous sections. The messages are used to allow the dynamic configuration of the DFMS Network, without the need to hard-code Network Addresses in the DFMS components. The Network Interface of each DFMS component only needs to know the Network Address of the Network Directory Server in order to establish the initial connection.

The Network Interface of the DFMS components that receive the *Network Update* messages update their local Network Address information. When the DFMS component wants to communicate with a new destination, a logical link can now be established.

3.8.2 Starting Up a DFMS system

When a DFMS system is started up a number of unconnected Function Modules and Station Managers exists. Physically all components are connected to the network, but communication between the components is not yet possible.

When a new component is initialized, the Network Interface of the component communicates with the Network Directory Server to obtain an up-to-date copy of the Network Address information. After the connection to the network has been completed, the new DFMS component sends a *Ma-*

been possible to send only an *incremental* update. Sending shorter messages however does not significantly speed up the processing of the messages.

chine Up message to the Operator Module[23]. The message contains the name of the DFMS component and the Operation Classes that it supports. An example of the message is shown in figure 3.23.

Machine Up

Operation Class
OC9000
OC9500
OC7000
OC5200

Figure 3.23: An example of a Machine Up message.

The Operator Module receives the *Machine Up* message and updates its *World Model*. The World Model contains information about the available DFMS components and the Operation Classes they support. An example of a World Model is shown in figure 3.17. After each update of the World Model the Operator Module sends a copy to each DFMS component by means of an *Updated Machine Capabilities* message. This message contains a complete new World Model.

The DFMS components that receive an *Updated Machine Capabilities* message compare the new World Model with the previous one and determine which components have been added or removed. If a new component has been added with which communication is necessary, the Network Interface is commanded to establish a logical link with the new component[24]. In figure 3.24 is shown which DFMS components need to communicate with each other.

3.8.3 Connecting the Tool Module

Connecting the Tool Module requires some additional action, besides establishing the network links and updating the World Model. The Tool Module is responsible for the management of tools in the system. In order to be able to perform this function, it is important that the Tool Module knows which tools are currently in the system. When a Station Manager is added to the

[23]This functionality of the Operator Module has not yet been discussed in section 3.5.

[24]The Network Address of the new destination is known by the Network Interface, because previously the Network Address information has been updated.

	SM	TM	PTM	PM	LM	OM
SM	y	y	y	y	y	y
TM	y	n	n	n	n	y
PTM	y	n	n	n	y	y
PM	y	n	n	n	n	y
LM	y	n	y	n	n	y
OM	y	y	y	y	y	n

SM: Station Manager
TM: Tool Module
PTM: Pallet Transport Module
PM: Program Module
LM: Load Module
OM: Operator Module

Figure 3.24: An overview of necessary links between DFMS components.

DFMS, the Tool Module has to know about the tools that are stored in the local tool store of the machine.

After a new Station Manager has been successfully added to the system and the *Updated Machine Capabilities* message has been received, the Station Manager checks if the Tool Module is already connected to the system. If so, the Station Manager sends a *Tool Startup Information* message to the Tool Module. This message contains data about all tools currently in the local tool store of the machine[25]. An example is shown in figure 3.25.

The Tool Module updates its information of the tools in the system[26]. If some Station Managers have been added to the system before the Tool Module is connected, they will send *Tool Startup Information* messages as soon as they detect that the Tool Module has become active[27].

[25]The Station Manager knows which tools are locally available, because the tool data has been uploaded from the CNC to the Station Manager as part of the Station Manager startup.

[26]It is important for the Tool Module to know the Tool Identifications of all tools in the system, because it must be avoided that a new tool is assigned a Tool Identification that already exists in the system.

[27]This is the case when an *Updated Machine Capabilities* message has been received in which the Tool Module is mentioned for the first time.

Tool Startup Information

Tool Identification	Remaining Tool Life
T12636300	20
T14244400	34
T15342400	17
T52666200	28
T59828000	31

Figure 3.25: An example of a Tool Startup Information message.

3.8.4 Connecting the Pallet Transport Module

Connecting the Pallet Transport Module requires a similar procedure as connecting the Tool Module. The Pallet Transport Module has to be aware of all pallets in the system. When a new machine and Station Manager are added to the system, the new Station Manager informs the Pallet Transport Module of the Pallet Identifications of pallets that are stored in the local pallet buffer by means of a *Pallet Startup Information* message. If the Pallet Transport Module is not yet active, the message is sent as soon as the module is connected to the system. An example of the *Pallet Startup Information* message is shown in figure 3.26.

Pallet Startup Information

Pallet Identification
P12
P43
P54
P88

Figure 3.26: An example of a Pallet Startup Information message.

All components that have control over pallets must be able to send a *Pallet Startup Information* message. When pallets are stored in *local pallet buffers*, the Station Manager of the machines will send the message. When the system has a *central pallet buffer*, the controller of that buffer is responsible for sending the message.

3.8.5 Connecting the Load Module

In order to function properly the Load Module has to know about the number of pallets in the system. In addition the number and types of the fixtures are important information for the Load Module[28].

When a new Station Manager is added to the system, the Station Manager sends a *Load Startup Information* message to the Load Module. The message contains information about the pallets that are in the local pallet buffer. If fixtures are mounted on the pallets, the Fixture Identifications of those fixtures are included in the message. If products are clamped on the pallets the Load Module is informed. An example of the *Load Startup Information* message is shown in figure 3.27.

If the Load Module is not yet available at the time a new Station Manager is added to the system, the message will be sent as soon as the Load Module is connected to the DFMS Network.

Load Startup Information

Pallet Identification	Fixture Identification	Product
P12	F100	No
P43	—	No
P54	—	No
P88	F120	Yes

Figure 3.27: An example of a Load Startup Information message.

With the information in a *Load Startup Information* message the Load Module updates its list of available pallets and fixtures. When a product is clamped on one of the pallets, the Load Module requests the Pallet Module to deliver the pallet containing the product. At one of the clamping stations the Product Script will then be inspected and the product will be introduced into the system[29].

[28] These data are necessary to determine which products can be clamped on the available pallets.

[29] It is assumed that a machine operator will have to supply the Station Manager with the information to send a Load Startup Information message. It is a simple problem for the operator to determine the Pallet and Fixtures Identifications. The Product Script of the product however has to be read before the product can be introduced into the system. This task is assigned to the Load Module. It is possible to have the Station Manager perform this task as well.

3.8.6 Adding a Station Manager

A new machine is added to the system by simply connecting the Station Manager to the DFMS Network. The machine is powered up and the CNC of the machine is initialized in the usual way. The Station Manager is initialized and connected to the network. Part of the initialization of the Station Manager is the uploading of the CNC status information to the Station Manager. The CNC status includes:

- the tool data of the tools in the local tool store;
- information about the pallets in the local pallet buffer.

The uploaded tool data is used to send the Tool Startup Information message to the Tool Module, the uploaded pallet data is used to send the Load Startup Information message to the Load Module and the Pallet Startup Information message to the Pallet Transport Module.

To complete the addition of the Station Manager to the DFMS, the New Machine message is sent to the Operator Module that 'broadcasts' the updated World Model to all components in the DFMS system.

3.8.7 Removing a DFMS component

In principle all DFMS components can be removed from the system. However, removing a Function Module is a procedure that will normally not occur, because it will cause problems when one of the Station Managers require services of the removed Function Module. Removing a Function Module in a running system is therefore not supported.

A machine however may have to be removed, for example for maintenance purposes. When a machine is to be removed, the Station Manager sends a *Machine Down* message to the Operator Module. The message does not contain information, but will just cause the Operator Module to update the World Model. If there were Operation Classes that were supported exclusively by the Station Manager just removed, those Operation Classes are no longer supported[30]. Because the Station Manager has been removed no new operations can be allocated. The operations that were already accepted by the Station Manager are passed to the Operator Module that will try to reallocate them to the remaining Station Managers.

[30]This may lead to Product Allocation Exceptions later on when products have been introduced that require operations of that Operation Class.

If there are pallets in the local pallet buffer, the Station Manager will request the Pallet Transport Module to remove them in order to prevent them from becoming blocked when the machine is removed from the system. If an operation is active, the Station Manager waits for the completion of that operation. When the operation is ready and the last pallet has been removed, the Station Manager can be shut down completely.

3.9 Handling crashes

When DFMS components fail, it is important that the system be able to recover gracefully. In this section the consequences of failures of Station Managers and Function Modules will be discussed.

Only the consequences of crashes that can be rectified by restarting the crashed components are discussed. When a Function Module crashes and when it is not possible to start the Module again, the consequence will be a complete system standstill after some time. Problems of this type however, are expected to be highly exceptional.

3.9.1 Failure of the Pallet Transport Module

When the Pallet Transport Module fails, all pallet transports that were currently executed are aborted. The Station Managers that issued the request will not be informed, but the requested pallet will not be delivered. No further operations are possible on the machine until the Pallet Module has become active again.

Before the Pallet Transport Module can be restarted, the Pallet Transport System has to be brought into a defined state. Special procedures defined for the Pallet Transport System will be applied. These procedures fall outside the scope of the DFMS architecture.

When the Pallet Transport Module is restarted, the Station Managers send Pallet Startup Information messages, so that the Pallet Transport Module can rebuild its pallet location information. The Station Managers that had issued transport requests, without having received the requested pallets will *resend* the Pallet Transport Request message. The requested pallets will be delivered and the operations can start.

A failure of the Pallet Transport Module will not necessarily have consequences for operations on the machines. When the required pallets are already available in the pallet buffers of the machines, the failure of the

Pallet Transport Module may pass unnoticed by the Station Managers[31]. It is also possible that the Pallet Transport Module will have recovered before one of the Station Managers request services of the Pallet Transport Module.

3.9.2 Failure of the Tool Module

Failures of the Tool Module can be handled exactly like failures of the Pallet Transport Module. The problems will pass unnoticed to the system if no services are requested from the Tool Module during the crash. However when Station Managers request services of the Tool Module when the Tool Module has crashed, the Station Managers will wait until the Tool Module has recovered.

After the crash the Tool Module can be immediately restarted. The Station Managers send Tool Startup Information messages to let the Tool Module rebuild its tool location information. All reservations and deliveries that may have been pending are cancelled and the Station Managers reevaluate the tool usage of the operations in their Operation Queues. Reservation requests and delivery requests are reissued.

3.9.3 Failure of the Station Manager

A failure of the Station Manager only influences the operations on the machine itself. The rest of the system will not be affected by the failure. If the CNC also has crashed, the CNC will have to be initialized before the Station Manager can be restarted. When the CNC is not affected by the crash the operation can continue. When the operation is finished, the Station Manager can be restarted and can be reconnected to the DFMS Network.

A serious consequence of a crash of the Station Manager is that the Operation Queue may have got lost. If that happens a number of products in the system will not be listed in any Operation Queue. No operations will be performed on those products; they will never leave the system. This problem occurs not only when a Station Manager crashes, but more generally, when any component that maintains an Operation Queue crashes. In section 3.9.7 a special Product Recovery Procedure will be discussed that recovers 'lost' products.

[31] This situation applies when the machines are equipped with large local pallet buffers. To provide for this situation the Pallet Transport Module specification has to be adapted in order to enable the Pallet Transport Module to determine the destination of the pallets that are specified as 'Store' pallets in the Pallet Transport Request messages.

3.9.4 Failure of the Operator Module

The Operator Module is only needed during configuration changes of the DFMS and when Station Managers are not able to perform operations on products and want to pass control over the product to the Operator Module. A crash of the Operator Module will therefore generally have no consequences at all. The Operator Module can be restarted and the system continues functioning as if nothing has happened. However, when the services of the Operator Module are requested during a failure of the Operator Module, the requesting DFMS component will wait until the Operator Modules recovers.

Just as a Station Manager, the Operator Module maintains an Operation Queue, which during a crash may get lost. After a crash of the Operator Module it will therefore always be necessary to perform the Product Recovery Procedure.

3.9.5 Failure of the Load Module

The Load Module is used to introduce products into the system and functions as a Station Manager to perform product unclamping operations. When the Load Module crashes no new products can be introduced into the system. Station Managers that request the Load Module to accept products for unclamping operations will not be able to reallocate their products. The control over those products is passed to the Operator Module.

As soon as the Load Module is restarted and as soon as all Load Startup Information messages have been received, the Load Module resumes its normal operations.

The Load Module maintains an Operation Queue and is therefore susceptible for losing products during a crash: the Product Recovery Procedure always has to be executed after a crash.

3.9.6 Failure of the Program Module

Failures of the Program Module do not have consequences at all. The Program Module can simply be restarted and linked again to the network. Station Managers that requested the downloading of a program at the time of the crash will resend their request.

3.9.7 The Product Recovery Procedure

The Load Module maintains a list of products that have been introduced into the system. All products in this Product List should occur in one (and only one) of the Operation Queues in the system. When a product in the Product List does not occur in the system that product apparently was lost during a crash and needs to be recovered.

The Product Recovery Procedure is performed by the Load Module and consists of requesting the Station Managers and the Operator Module to upload their Operation Queues by sending a *Operation Queue Request*. The modules that receive the request reply by sending a *Operation Queue Reply* message that contains all products that they currently have under their control. An example of a Operation Queue Reply message is shown in figure 3.28.

Operation Queue Reply

Order Identification
O3410
O4352
O6625
O6326

Figure 3.28: An example of an Operation Queue Reply message. Note that only a summary of the Operation Queue is sent.

Once all Operation Queue Reply messages have been received, the Load Module can determine which products are not accounted for. When products seem to be lost, the Load Module will request the Pallet Transport Module to deliver the pallets on which the products are clamped. When the pallets arrive, the Product Scripts on the pallets can be inspected and the Load Module can reintroduce the products into the system.

The Product Recovery Procedure functions even when a number of Station Managers crash simultaneously: it is not necessary to know in which Operation Queue the recovered product originally was listed. However, when the *Product List* in the Load Module is lost, it is not possible to perform a reliable Product Recovery Procedure. The Load Module therefore has to be implemented using methods that make the corruption of its Product List extremely unlikely. In the exceptional situation that Operation Queues have

been lost and that the Product List is not reliable, the system operators have to manually account for each product in the system.

3.9.8 DFMS Network problems

When a DFMS component crashes, any DFMS component that wishes to communicate with the crashed component will not succeed. The Network Layer of the DFMS component that tried to send will detect the failure and will so inform the DFMS component, which will then handle the situation.

When a request message has been sent to a remote DFMS component to which a reply is expected, and when the remote component crashes before the reply has been sent, there is a danger that the local DFMS component waits forever for the reply. In order to prevent this, the DFMS component times out when it takes too long for a reply to arrive. When a time out occurs, the local component will handle the situation as if sending the original request had failed. No DFMS component ever gets into trouble because of failures to send or receive messages.

Chapter 4

Implementation of DFMS

The DFMS architecture as described in the previous chapter is structured to allow a distributed implementation. Two different realizations of a distributed implementation will be discussed.

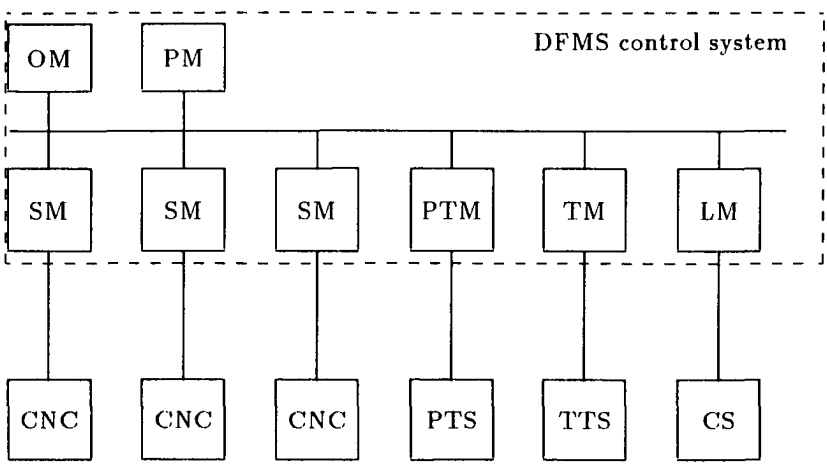
4.1 Distributed implementation

In a distributed implementation, the components of the DFMS architecture are implemented on separate computer systems. The computer systems are connected by means of a data communication network: for example by a *Local Area Network*. The messages defined in the architectural description are implemented as messages that are sent over the network. An overview of this implementation is shown in figure 4.1.

The distributed implementation is a *direct* implementation of the DFMS architecture. The distributed character of the architecture is completely preserved with this distributed implementation. Some advantages of the distributed implementation are:

- **Reliability**
  Because a number of (small) computer systems is used, it is easy to replace a computer system that is not functioning properly by a spare computer system;

- **Extensibility**
  It is easy to extend the DFMS control system by simply adding more DFMS components. The only component that is used for extension is the Station Manager.



OM:	Operator Module	LM:	Load Module
PM:	Program Module	CNC:	Computerized Numerical Control
SM:	Station Manager	PTS:	Pallet Transport System
PTM:	Pallet Transport Module	TTS:	Tool Transport System
TM:	Tool Module	CS:	Clamping Station

Figure 4.1: A distributed implementation. All DFMS components are implemented on separate computer systems.

Different implementations of the DFMS architecture can be obtained if some of the Function Modules are combined into a single computer system. For example the Load Module and the Pallet Transport Module can be combined to reduce the necessary number of computer systems. In that case however, the advantages listed above will become less explicit. A failure of a computer system on which a number of Function Modules are implemented, will affect all those Function Modules.

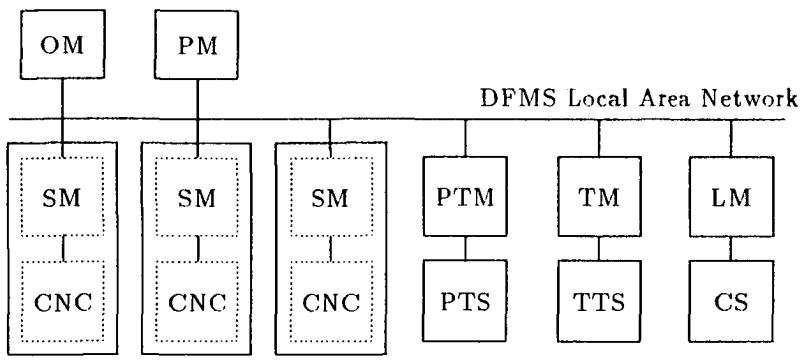
In appendix A two examples of centralized implementations are discussed. These do not have many advantages in contrast with the distributed implementation. They are shown for completeness.

4.2 Realization of the distributed implementation

Just as there are several ways to implement an architecture, so are there several ways to *realize* an implementation. Two realizations will be discussed: a realization in which Station Managers are implemented on separate computer systems and a realization in which the Station Managers are integrated into the controls of the machines. In both cases, the Function Modules will be implemented on separate small computer systems.

4.2.1 Programmable CNC realization

The distributed implementation defines a separate computer system for each of the DFMS system components. Each machine in the system is already equipped with a *Computerized Numerical Control* (CNC). It is now possible to integrate the Station Manager functionality with the CNC functionality. To implement both the CNC and the Station Manager in one computer system a *programmable CNC* is required. A programmable CNC is a computer system in which the standard CNC functionality is implemented and that allows a user to run additional user-defined system programs. The programmable CNC has a dual functionality: the traditional CNC functionality and the Station Manager functionality. Between the two functions a well defined interface exists. An overview of this realization is shown in figure 4.2. The specifications of a programmable CNC are discussed in section 4.2.3.

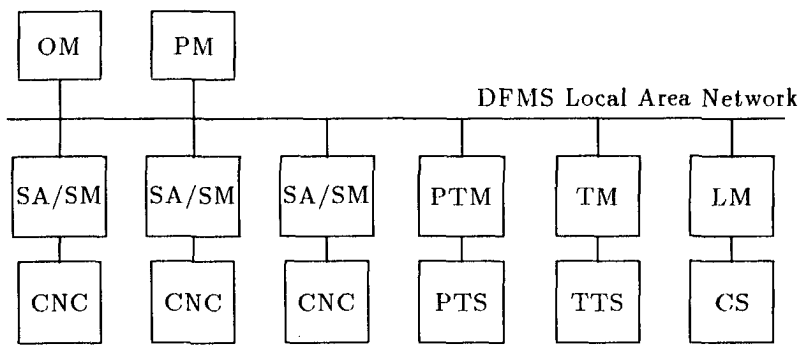


OM:	Operator Module	LM:	Load Module
PM:	Program Module	CNC:	Computerized Numerical Control
SM:	Station Manager	PTS:	Pallet Transport System
PTM:	Pallet Transport Module	TTS:	Tool Transport System
TM:	Tool Module	CS:	Clamping Station

Figure 4.2: A realization of a DFMS system with programmable CNCs. The drawn boxes represent computer systems. The dashed boxes represent functions.

4.2.2 Station Adapter realization

The realization described in the previous section presumes the availability of programmable CNCs. Not all machines to be integrated in a DFMS system will be equipped with a programmable CNC[1]. It is possible to implement the Station Manager functionality on a separate small computer system, connected to the CNC by a standard DNC link. The system that is used for this purpose is called a *Station Adapter*. The Station Adapter can be connected to the CNC by means of a simple serial data connection. The Station Adapters are connected to each other and to the Function Modules by means of a Local Area Network. An example of a Station Adapter realization is shown in figure 4.3.



OM:	Operator Module	LM:	Load Module
PM:	Program Module	CNC:	Computerized Numerical Control
SA/SM:	Station Adapter/Manager	PTS:	Pallet Transport System
PTM:	Pallet Transport Module	TTS:	Tool Transport System
TM:	Tool Module	CS:	Clamping Station

Figure 4.3: A realization of a DFMS system with Station Adapters. The drawn boxes represent computer systems.

[1] As of now, programmable CNCs are not widely available at all. It is to be expected though that major CNC manufacturers will announce CNCs with these capabilities in the near future. A prototype of a programmable CNC will be discussed in section 4.5.

4.2.3 Programmable CNC versus Station Adapter

The two realizations discussed in the previous section have advantages and disadvantages. Advantages of the programmable CNC as opposed to the Station Adapter approach are:

- It is to be expected that a programmable CNC will be the most cost effective realization. A Station Adapter has to work in a hostile environment and for that reason it has to be an *industrial quality computer system*. These systems tend to be expensive. The CNC is suited to work in hostile environments and can therefore be used to implement the Station Manager functionality without high additional costs;
- The data connection will be simpler and less vulnerable. When a Station Adapter is used, an extra interface between CNC and Station Adapter will be necessary. This extra interface, typically a V24 interface, adds to the complexity of the system, increases the costs and decreases performance.

The Station Adapter realization does not have any advantages compared to the programmable CNC realization, except that it allows machines with older, non-programmable CNCs to be integrated in a DFMS system.

4.3 Advantages of the distributed DFMS realization

The distributed implementation of the DFMS architecture has certain important advantages that will be discussed in this section.

The DFMS *distributed architecture* is extremely simple. The functionality of the system components is well defined and the components perform activities that look natural to human observers. The system components communicate by sending and receiving messages; a small set of messages has been defined. The information contents of the messages is straightforward and is easily understood. Important operations, such as starting the system, adding machines and removing machines, are very simple procedures in DFMS. The system is inherently robust and will be able to handle all but the most serious exception situations gracefully[2].

[2]Examples of exception situations that cannot be handled gracefully are failures of the data communication function or prolonged failures of some of the Function Modules.

The advantage of the *distributed implementation* is the possibility to extend the control system in small steps. If a new machine is to be added to the DFMS system, the control system is simply extended by adding another Station Manager. At each stage of the system the installed control capacity exactly matches the required capacity.

The main advantage of using *Programmable Computerized Numerical Controllers* to realize the distributed implementation is the low cost of the resulting control system. The additional hardware costs to extend the CNC functionality in order to include the Station Manager functionality is expected to be low.

The extreme simplicity and the low cost of DFMS, enable the application of FMS in areas where previously FMS was not practicable. The distributed implementation of DFMS enables a gradual introduction of FMS technology in small companies.

The FMS systems with traditional control systems that have been installed so far, are complicated, expensive and can only be used by well-trained users. These facts have limited the application of FMS severely. It is to be expected that the new DFMS concept does not have any of these restricting characteristics.

4.4 Specification of a Programmable CNC

In the previous sections the concept of a *Programmable CNC* has been introduced. In order to integrate a programmable CNC in DFMS, the CNC has to meet certain specifications. In this section these specifications will be discussed.

4.4.1 Interface between CNC and Station Manager

A strict separation between the functionality of the Station Manager and the CNC functionality is of great importance. The CNC process is a time critical process. The CNC controls the operation of the machine and disruption of the activities of the CNC is potentially dangerous. To prevent disruptions by the Station Manager a very tight interface between Station Manager and CNC has to be defined. The interface definition disallows the Station Manager to influence the CNC in a manner that endangers its correct operation. A schematic overview of the interface between CNC and

Station Manager is shown in figure 4.4. The Station Manager is able to

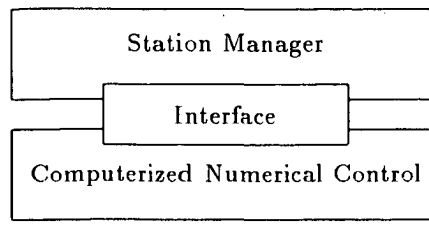


Figure 4.4: A schematic overview of the interface between CNC and Station manager. Communication is only possible through the interface.

inspect CNC data structures and can issue a limited set of commands. A summary of the commands that can be issued by the Station Manager:

- **Start Operation 'ProgNr' 'Pallet Id'**
  The command is used to start an operation. The parameters are the name of the program that has to be executed and the pallet on which the product is clamped. The CNC will transfer the specified pallet to the position where the operation is performed and will start the operation;

- **Continue Operation**
  This command is used to continue the operation, when an operation has been suspended (for example because a tool was not available);

- **Exchange Tool 'Tool Id'**
  When a new tool arrives, the Station Manager chooses for removal a tool that will not be needed for some time. The tool to be chosen is specified as a parameter in the command;

- **UpLoad Tool Memory**
  The Station Manager requests the uploading of the Tool Memory of the CNC, after each operation has completed. The tool data are required to analyze the tool requirements of the operations in the Operation Queue;

- **Upload Pallet Memory**
  The Upload Pallet Memory command is usually issued only once: during initialization of the Station Manager. The Station Manager has to

know which pallets are locally available in order to be able to manage the local pallet buffer;

- **DownLoad Program 'ProgNr'**
  The Station Manager will download the required program, when an operation has to be performed for which no program is currently available in the CNC. The name of the program to be downloaded is specified;

- **UpLoad Program 'ProgNr'**
  When a NC part program has been adapted on the machine, the changed program is uploaded in order to save the changes that have been made;

- **Delete Program 'ProgNr'**
  The Station Manager will delete a program that will not be needed in the near future, when not enough space is left in the program memory of the CNC to download a new program. The program that is to be deleted is specified;

- **Store Pallet**
  The pallet that has just arrived and from which the pallet data have been read is transferred from the read/write position to a standard pallet buffer position.

An overview of the status messages that the CNC can report to the Station Manager:

- **Operation Completed**
  This message is sent to the Station Manager, when an operation has been completed;

- **Operation Aborted**
  This messages is sent to the Station Manager, when an operation has been aborted. A reason to abort an operation is for example damage to the product after a tool failure occurred;

- **New Pallet**
  When the Pallet Transport System has delivered a new pallet, the CNC informs the Station Manager by means of this message. It is up to the Station Manager to read the pallet data (see section 4.4.3);

- **Pallet 'Pallet Id' Removed**
  When a pallet has been removed by the Pallet Transport System, the CNC informs the Station Manager by means of this message. The identification of the pallet that has been removed is specified as a parameter;

- **New Tool 'Tool Data'**
  This message is sent when the Tool Transport System just delivered a new tool. The data of the new tool are specified as parameters;

- **Tool 'Tool Id' Missing**
  When the CNC does not succeed in finding a tool of the required Tool Class, this message is sent to the Station Manager.

4.4.2 The Tool Data interface

As has been explained in the previous chapter, in DFMS the preferred method of handling tool data is based on the use of programmable data carriers in the tool holders of the tools. When a tool is inserted in the tool store of a machine, the tool data have to be read in order to enable the CNC to use the tool. Reading the tool data is the responsibility of the Station Manager.

The following procedure will be followed when a new tool arrives at the tool store:

1. The Tool Transport System places the tool in a fixed position from which it is possible to read the tool data. The physical transfer of the tool is controlled by both the Tool Transport System and the CNC;

2. The CNC reads the data from the tool holder. These data contain the Tool Class, the Tool Identification, the Tool Life and the tool dimensions;

3. The CNC sends a 'New Tool' message to the Station Manager. The message contains the new tool data;

4. The Station Manager sends an 'Exchange Tool' message to the CNC in which the tool to be removed is specified.

A schematic overview of the programmable CNC with tool data interface is shown in figure 4.5.

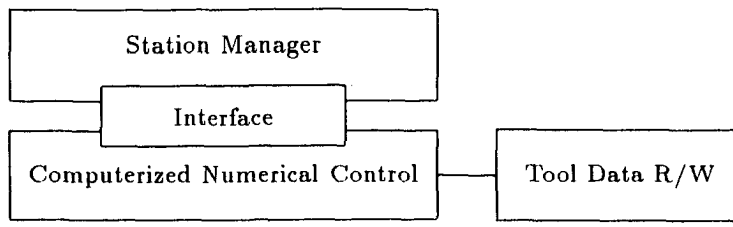


Figure 4.5: A CNC with interface to Tool Data Reader/Writer.

4.4.3 The Pallet Data interface

The Product Scripts in DFMS are transferred with the pallet in programmable data carriers connected to pallet. The Product Script data have to be read when a pallet arrives at a machine. The Station Manager is responsible for reading and writing the Product Script data.

When a pallet arrives at the machine, the following procedure will be followed:

1. The Pallet Transport System delivers a pallet to the local pallet buffer. The Pallet Transport System and the CNC coordinate the actual transfer of the pallet from the transporter to a buffer position. The buffer position is equipped with a pallet data reader;

2. The CNC sends a 'New Pallet' message to the Station Manager;

3. The Station Manager reads the Product Script data from the programmable data carrier;

4. The Station Manager sends a 'Store Pallet' message to the CNC, in order to transfer the pallet to a standard pallet buffer position.

A schematic overview of the programmable CNC with both a tool data interface and a pallet data interface is shown in figure 4.6.

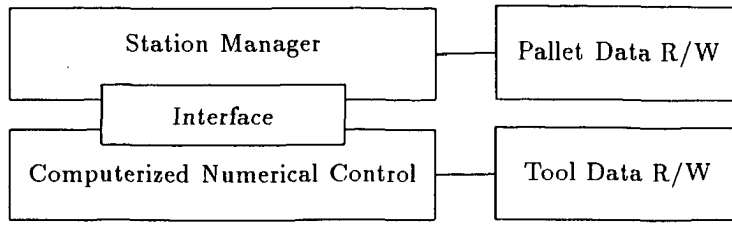


Figure 4.6: A CNC with interface to Pallet - and Tool Data Reader/Writer.

4.5 Prototype of a Programmable CNC

The concept of a programmable CNC has already been adapted by several CNC manufacturers and it is to be expected that more will follow[3]. A particularly interesting realization of a programmable CNC is one in which the user is provided with a MS-DOS environment. MS-DOS is a very widespread operating system for which a huge amount of software has been developed. A realization of a programmable CNC by Philips Numerical Control uses a separate XT-board that is connected to the CNC by means of an external interface[4]. The Philips CNC consists of a backplane to which a number of special purpose boards are connected. One of these special purpose boards implements a serial interface. The XT board communicates with the CNC by means of this serial interface. The XT board uses the keyboard and video display unit of the CNC. An overview of this solution is shown in figure 4.7. In principle, the XT board can be used to run every standard MS-DOS program[5]. However, because the keyboard and video display unit of the CNC are used, the CNC imposes its limitations to the programs that can be run on the XT board. The interface supported between Station Manager and CNC is an extension of the *Direct Numerical Control* interface as currently supported by Philips Numerical Control [Vis 88]. This interface definition supports roughly the interactions described in section 4.4.1; some

[3]The German control manufacturer IBH demonstrated a commercially available version of a MS-DOS CNC on the EMO in Milan in 1987. Philips Numerical Control demonstrated a prototype in 1988 on the METAF in Düsseldorf.

[4]An XT-board is a printed circuit board that is the main component of IBM-XT computer systems.

[5]In the current version no full screen handling is supported. This may be added in future versions of the programmable CNC.

changes regarding the handling of pallet - and tool transports will have to be made.

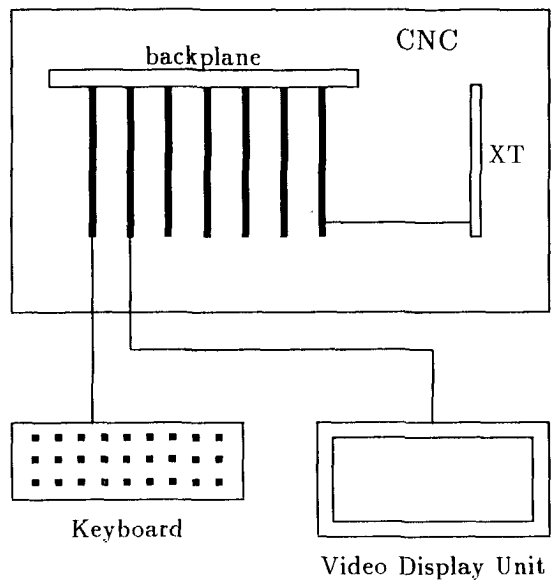


Figure 4.7: An overview of the programmable CNC. Two separate computer systems are used. The backplane holds special purpose boards. One board is used to establish the connection between the XT board and the backplane system. Two other boards are used to control keyboard and video display unit.

Chapter 5

The DFMS Prototype

5.1 Introduction

A DFMS prototype has been developed for the following purposes:

- to analyze the technical feasibility;
- to generate software that can be used in a first prototype of the programmable CNC;
- to have a system available for demonstration purposes.

A system consisting of a number of personal computers has been used to implement the Function Modules and Station Managers. The functionality of machines, clamping stations, transport systems and tool room have been simulated in software. In this chapter the details of the prototype will be discussed.

5.2 Hardware used for the prototype

The main components that have been used to build the prototype are IBM compatible AT computers and IBM compatible XT computers. In the rest of this chapter these systems will be referred to as respectively ATs and XTs. When no distinction has to be made between ATs and XTs the common name 'PC' will be used.

The XTs are equipped with Intel 8088 processors and contain 512 kbyte of internal memory. The 8088 processors run at a clock speed of 4.7 MHz. The ATs have Intel 80286 processors running at 8 or 10 MHz and contain

1024 kbyte internal memory of which 640 kbyte can be addressed directly by the operating system.

Both XTs and ATs are equipped with serial interfaces that connect to a central microVAX II. The ATs communicate with a speed of 9600 baud, the XTs communicate with a baud rate of 2400. The microVAX II is used as a message router for the messages sent between the PCs.

5.3 Internal structure of prototype software.

In this section some aspects of the software organization of the PC's will be discussed. The influence of the operating system characteristics on the software organization of the system will be explained. Special attention will be given to the software organization required to obtain multi-tasking capabilities in a single-task environment.

5.3.1 The operating system

Multi-tasking and single-tasking operating systems

Applications in which the operations of machines have to be controlled or in which data communication is necessary generally require *multi-tasking* operating systems. In multi-tasking operating systems a number of *tasks* can be performed quasi parallel. Tasks *compete* for processor time; *time slices* are allocated to each task by the operating system. Each task can be suspended at any time to allow another task to use the processor. The operating system applies a simple mechanism to determine which task has the highest priority.

Multi-tasking is required for these applications, because the computer application has to react on events from more than one source. It is important that the application reacts as soon as an event occurs. For more details on the principles of operating systems refer to [List 75] and [Com 86].

For reasons of *simplicity* some operating systems are *single-task* oriented. In these systems the concept of multiple tasks is not supported and consequently applications running under single-tasking operating systems can usually only adequately react on events from only one source or in special cases from two sources[1].

[1] A examples of programs that are able to handle inputs from two sources are *terminal emulators*. To obtain this functionality very low level interaction with operating system and hardware is required.

The MS-DOS operating system

The MS-DOS operating system is an example of a single-task operating system that has become the defacto standard for personal computer usage. An enormous amount of software has been developed for the MS-DOS environment. Very sophisticated software development tools like editors, compilers, object code libraries and debuggers are available for MS-DOS. Even though MS-DOS, as a single-tasking operating system, is not really suited for applications in which multi-tasking capabilities are required, the advantages of using MS-DOS are great.

5.3.2 Data communication

The PC's have to be able to communicate with each other in order to perform the DFMS functionality. A large number of network products are available for MS-DOS environments, but few support the required *task to task communication* functionality[2]. Task to task communication allows tasks on the same computer or tasks on different computers to communicate directly with each other. Examples of products that support task to task communication are *Locan* of Philips Hasselt and *DecNet DOS* of Digital Equipment Corporation (DEC). Both products have been tested extensively for suitability for use in the DFMS prototype; only DecNet DOS proved to be able to perform the data communication function reliably.

DecNet DOS multi-tasking

As has been mentioned in section 5.3.1 a multi-tasking capability is required to perform data communication. In the DecNet DOS application this problem is solved by an extension to the standard MS-DOS operating system. The DecNet DOS extension implements a rudimentary multi-tasking operating system that has only two tasks. One task is the data communication task or DecNet task, the other task implements a standard MS-DOS environment. The DecNet task is provided by DEC, the MS-DOS task seems a usual MS- DOS environment to the user[3]. The DecNet DOS system shares processor time between the DecNet task and the MS-DOS task. The multi-tasking is completely transparent to the user. An overview of the two tasks

[2]These network products support, for example, personal mail and file server facilities, but no task to task communication.

[3]The available memory to the MS-DOS task is smaller than usual, in the DFMS prototype approximately 80 kbyte of memory is occupied by the DecNet task.

in a DecNet DOS system is shown in figure 5.1.

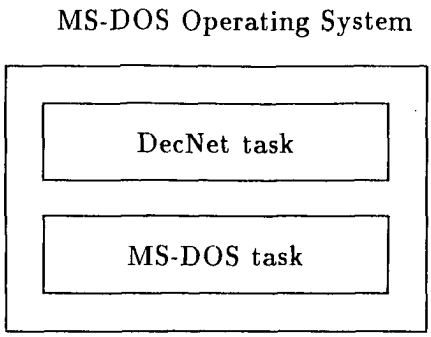


Figure 5.1: The two tasks in a DecNet DOS environment.

DecNet DOS interface

A program running in the MS-DOS task can issue calls that are processed by the DecNet task. The central concept in DecNet DOS is the *socket*. A socket is a connection to a *logical link*. Two tasks can only communicate with each other when a logical link exists between them. Each task can communicate with a number of remote tasks. To each task a separate logical link and hence a socket exists. DecNet DOS includes calls to establish links, to send data over a link and to read data from a link.

Network topology

The DFMS prototype uses a small number of PC's to implement the functionality of Station Managers and Function Modules. For data communication the DecNet DOS system is used. An overview of the physical means with which the network is implemented is shown in figure 5.2.

The microVAX II computer serves as a *message router*. When PC1 wants to send a message to PC3 the message is first transmitted to the micro VAX II where the DecNet software automatically routes the message to its destination: PC3. For the user this process is completely transparent. The *logical topology* of the network is shown in figure 5.3.

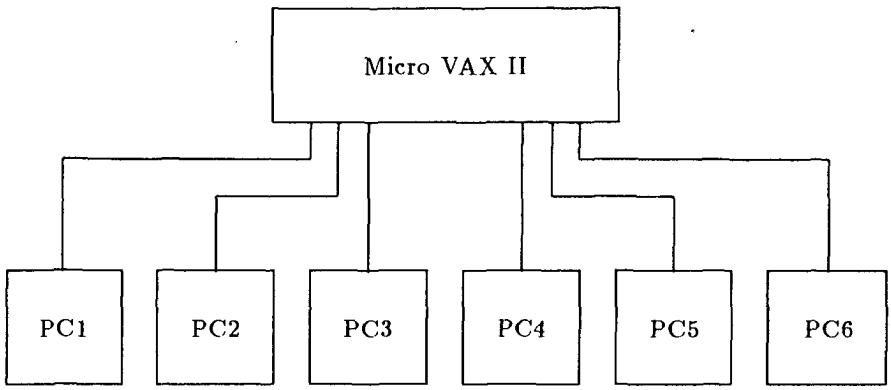


Figure 5.2: The physical network topology of the DFMS prototype.

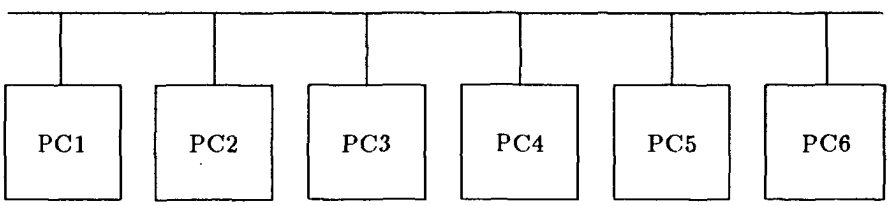


Figure 5.3: The logical network topology of the DFMS prototype.

5.3.3 The coroutine model

In the previous section the pseudo multi-tasking environment under DecNet DOS has been discussed. By applying this communication software we can guarantee that messages are *received* and *stored* immediately they are received. However, just storing the messages is not enough, it is important that the messages are *processed* soon after they are received.

All DFMS components do have an elaborate menu oriented user interface. The user chooses from menu options to determine which data structures will be shown. If no special precautions are taken, a user being slow in specifying the choices, could influence the speed with which received messages are processed; this is not acceptable. Within the MS-DOS task additional multi-tasking capabilities are required to prevent this problem from occurring. In order to obtain this capability, use is made of a *coroutine dispatcher*[1].

In the coroutine dispatcher a number of tasks can run quasi parallel. Unlike a full multi-tasking system where the operating system determines when the current task is suspended to let other tasks utilize the processor, in a coroutine system each task voluntarily passes control to the operating system by means of a *Suspend Self* call. The dispatcher applies a *round robin scheduling* mechanism to determine which task should become active. The new task remains active until it explicitly issues the Suspend Self call. For more details about coroutine programming see [Wak 81].

The coroutine dispatcher supports one event flag per task. The value of the event flag determines whether the task competes for processor time. When the event flag of a task contains a 0 the task is active and competes for processor time; when the event flag contains a non zero value, the task is dormant and does not compete for processor time. In figure 5.4 a graphic representation of tasks in a coroutine system is shown.

In the DFMS prototype the coroutine system is applied to implement the functionality of a DFMS component in two tasks.

- A *User Interface* task is responsible for the interaction with the operator and handles all keyboard and screen activities;

- A *Message Processing* task implements the DFMS functionality of the component. The Message Processing task receives messages that

[1] The dispatcher has been originally developed by Han Koster of the Philips Center For Manufacturing Technology.

are sent by remote DFMS components and processes them. If necessary, the Message Processing task updates internal data structures and sends messages to other DFMS components.

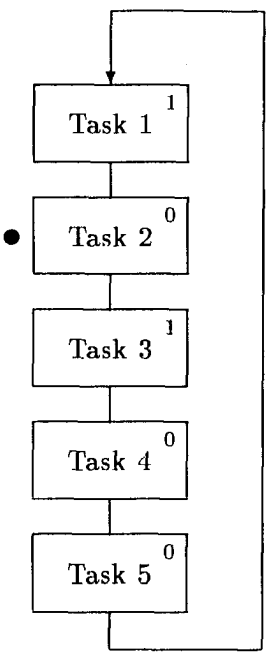


Figure 5.4: The tasks in a coroutine system. Only tasks 2, 4 and 5 are currently competing for processor time. Task 2 is the current task. When Task 2 issues a Suspend Self call the coroutine dispatcher selects the next computable task: Task 4. After Task 4 and Task 5 have issued a Suspend Self call, Task 2 will become the current task again. Task 1 and Task 3 will never become the current task unless one of the active tasks sets the event flag of Task 1 or Task 3 to 0.

The important data structures, necessary to implement the DFMS functionality, are defined within the context of the Message Processing task. The User Interface task is allowed to access these data structures for both reading and writing[5].

[5]Of course, the operator can only access *selected* data structures. Special menu options and functions have to be provided, in order to present the operator with the contents of

In a full multi-tasking time sliced operating system it is necessary to guard data structures from being accessed by two tasks simultaneously[6]. If data structures are not guarded it is possible that two task are updating a data structure simultaneously and that the update performed by one of the tasks is undone by the update of the other task. For more information on the problems of task synchronization see [Ben 82].

In a coroutine environment these problems do not occur, because the switching between tasks is performed on well defined places in the program code. A task switch only occurs when a task issues a 'Suspend Self' call. At that moment no data structures are accessed and no conflicts can result.

5.3.4 Overview of the software organization

In the previous sections aspects of the operating system, data communication and coroutine processing have been discussed. In figure 5.5 the software organization as currently applied in the PC's in the DFMS prototype is shown. The multi-tasking capabilities have been implemented in two levels: firstly by means of the DecNet software, secondly by the coroutine dispatcher. This layered implementation of multi-tasking has proven to work satisfactorily.

5.4 Overview of the DFMS prototype

In the DFMS prototype that has been developed, the following DFMS components are implemented:

- one Load Module;
- one Tool Module;
- one Pallet Transport Module;
- one Operator Module;
- one Program Module;
- three Station Managers.

An overview of the DFMS prototype is shown in figure 5.6. In the following sections each of the components of the DFMS prototype will be discussed in detail.

---

a data structure.

[6] In a time sliced system the operating system allocates a fixed amount of time to each task. After the time slice has elapsed, the task is suspended and the next task continues from where it was suspended earlier.

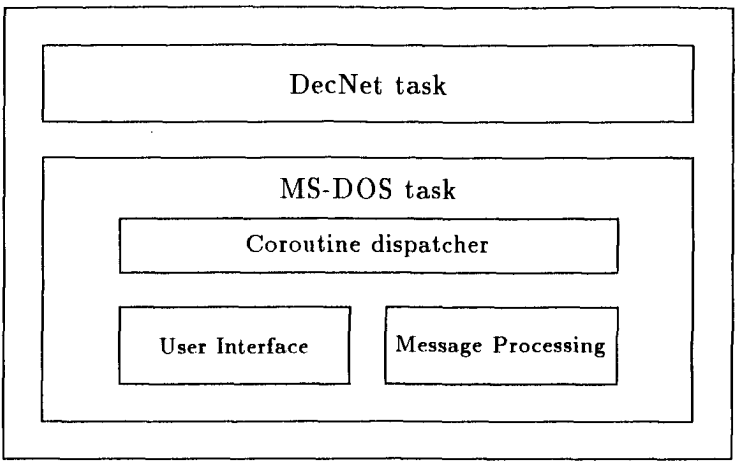


Figure 5.5: The software organization of the PC's in the DFMS prototype.

5.5 Operator Module/Program Module

The Operator Module and the Program Module are implemented on one PC. There was no particular reason to combine these two Function Modules except for the lack of computer systems. The implemented functionality of Operator Module and Program Module will be discussed separately.

5.5.1 Program Module

The DFMS prototype contains a rudimentary implementation of the Program Module. A simple implementation has been chosen because the Program Module was not considered to be a critical component with regards to the technical feasibility of DFMS.

The programs maintained by the Program Module do not contain the NC part program codes, but contain only the Tool Requirements List. The programs can be downloaded on request; uploading of changed programs is not provided. The Program Module maintains a number of programs that are read in from files when the module is initialized. No new programs can be added once the Program Module has been started. The operator can inspect the contents of the programs that are maintained by the program

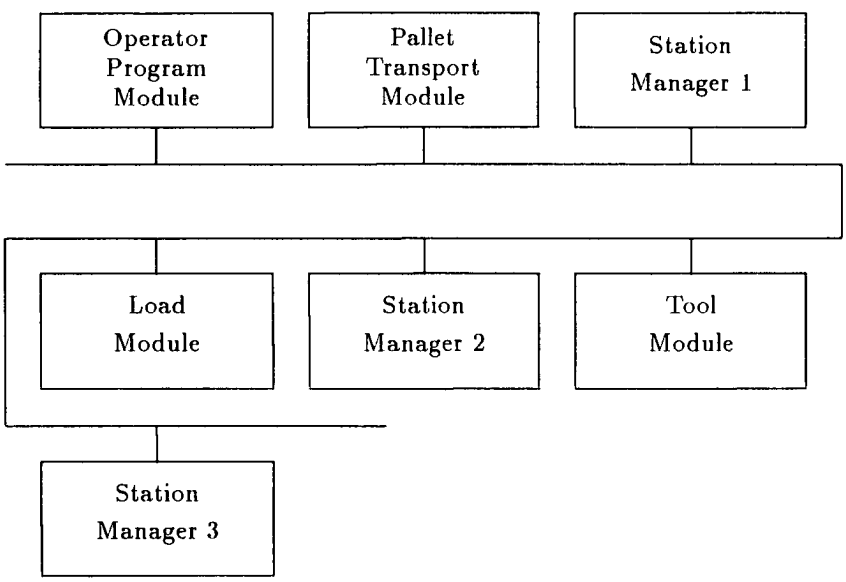


Figure 5.6: The layout of the DFMS prototype.

Module.

5.5.2 Operator Module

The functionality of the Operator Module can be divided in three major parts (see section 3.5.):

- System Interface;
- Error Recovery;
- Management Information System.

The Management Information System functionality has not been implemented at all. The System User Interface and the Error Recovery functionality have been implemented according to the architectural specifications.

User interface functionality

The user interface to the Operator Module provides the operator with the following options to influence the behavior of the system:

- remove a Station Manager from the system;
- add a supported Operation Class to a Station Manager;
- remove a supported Operation Class from a Station Manager;
- introduce a product in the Operation Queue into the system;
- remove a product in the Operation Queue from the system.

In addition, the User Interface supplies the operator with the possibility to view important data structures. Examples of data structures that can be inspected are:

- the Operation Queue;
- the World Model that lists the Operation Classes supported by each DFMS component.

Error recovery functionality

The Error Recovery functionality has been fully implemented. Station Managers that cannot perform or complete an operation on a product, hand over control to the Operator Module. The Operator Module places the operation in its Operation Queue or reintroduces the product immediately if the product came from a Station Manager that was shut down. The Operator can specify whether the product has to be removed from the system or whether the product will be reintroduced into the system (now or some time later).

When one of the DFMS components fails and when there is a danger that an Operation Queue has been lost, it is possible to initiate a Product Recovery Procedure from the Operator Module.

5.6 Tool Module

The Tool Module in the DFMS prototype has been implemented on a PC together with simulation modules for the tool room activities and the tool transports. The relation between the Tool Module and the simulation modules is shown in figure 5.7. The functionality of the Tool Module and the simulator modules will be discussed separately.

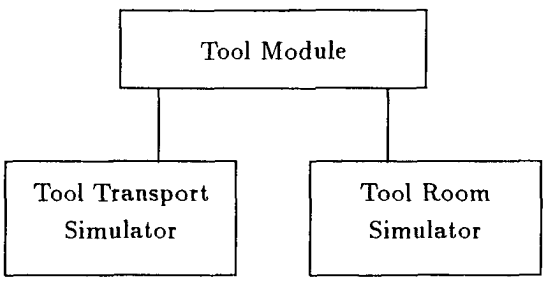


Figure 5.7: The relation between Tool Module and simulation modules.

5.6.1 Implemented Tool Module functionality

The Tool Module has been implemented in accordance with the architectural specification. Station Managers can reserve tools as soon as they predict a shortage of tools. The Tool Module allocates tools to Station Managers and

issues commands to prepare new tools. When Station Managers request delivery of tools for a particular order, the Tool Module orders the Tool Transport System to deliver the tools at their destination. Tools that are returned from the system can be reused if the remaining tool life is significant; otherwise the tool is discarded.

5.6.2 Simulating Tool Room operations

The Tool Module issues orders to operators to prepare tools. The orders are appended to a queue and operators process the orders one by one. The activities of operators that prepare tools are simulated by simply scheduling delays. After a delay has elapsed the Tool Module is informed that a tool assembly order has been processed. The number of operators determines how many tool orders can be processed simultaneously.

5.6.3 Simulating tool transports

The transport of tools is simulated by sending messages between Tool Module and the machines where the tools are delivered or removed. When the Tool Module orders the Tool Transport System to deliver a set of tools to a Station Manager, a *Tools In* message is sent to notify the arrival of the tools. An example of a Tools In message is shown in figure 5.8[7]. The Station Manager that receives new tools determines which tools will be exchanged. A *Tools Out* message is sent to the Tool Module to simulate the transfer of these tools to the central tool store. The syntax of the Tools Out message is similar to the syntax of the Tools In message.

Tools In

Tool Id	Tool Time
T12560004	23
T34500090	40
T13373301	100
T14571392	50

Figure 5.8: An example of a Tools In simulation message.

[7]The Tools In message used in the DFMS prototype does not contain tool dimension data.

5.7 Pallet Transport Module

The Pallet Transport Module in the DFMS prototype has been implemented on a PC together with simulation modules for the Pallet Transport System and for the system pallet buffer. The relation between the Pallet Transport Module and the simulation modules is shown in figure 5.9. The functionality of the Pallet Transport Module and the simulator modules will be discussed separately.

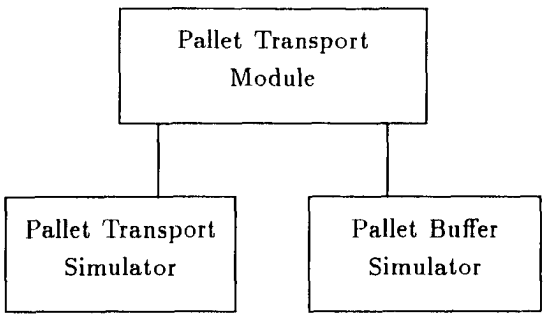


Figure 5.9: The relation between the Pallet Transport Module and the simulation modules.

5.7.1 Implemented Pallet Transport Module functionality

The Pallet Transport Module has been implemented in accordance with the architectural specification. Station Managers request transports of pallets. The Pallet Transport Modules allocates transport vehicles for the requests and issues commands to the Pallet Transport System in order to perform the transports.

5.7.2 System Pallet Buffer

The system pallet buffer is used to store pallets that have to be removed from local pallet buffers on request of the Station Managers. The Pallet Transport Module system can send transport vehicles to the system pallet buffer to store or remove pallets. Interactions with the system pallet buffer are handled internally. No Pallet In or Pallet Out messages have to be sent when pallets are stored in or are removed from the system pallet buffer.

5.7.3 Pallet Transport System

Simulating pallet transports

The Pallet Transport System simulator receives the following commands from the Pallet Transport Module:

Move	'Vehicle Id'	'Position'
Get-Pallet	'Pallet Id'	
Store-Pallet	'Pallet Id'	

The execution of the commands is simulated by scheduling delays, after which *Success* status messages are sent back to the Pallet Transport Module. The transport of pallets is simulated by sending messages from the Pallet Transport Module to the DFMS components where pallets arrive or are removed. A *Pallet In* message is sent to a DFMS component awaiting the arrival of a new pallet, a *Pallet Out* message is sent to the component where a pallet just has been removed. The syntax of the Pallet In - and Pallet Out message is similar. An example of a Pallet In message is shown in figure 5.10.

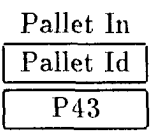


Figure 5.10: Example of a Pallet In message.

In actual systems the pallets contain *Product Scripts* in their programmable data carriers. In the DFMS prototype the Product Scripts have to be sent explicitly. The Pallet Transport Module maintains the Product Scripts of all pallets in the system. After a Pallet In message has been sent to a DFMS component[8], a *Pallet Data* message containing the Product Script is sent to the DFMS component. When the DFMS component has completed an operation on a product and has updated the Product Script, it sends a *Product Script Update* message to the Pallet Transport Module. The syntax of the Pallet Data and the Product Script Update messages is

[8]DFMS components that handle pallets are the Station Managers and the Load Module.

similar; both contain a complete copy of a product script. An example of a Pallet Data message is shown in figure 5.11.

Operation Sequence

Nr	Operation Class	Operation	Fixture	Time
1	CLAMP	C1267	F100	20
2	OC9000	O9321	F100	50
3	OC7000	O7253	F100	30
4	UNCLAMP	U0000	F100	10
5	CLAMP	C1829	F134	27
6	OC2100	O2186	F134	11
7	UNCLAMP	U0000	F134	8

Status Info

Order	NR732726
Next Operation	3
Pallet Id	p54
Priority	1

History Log

Operation	Event
C1267	Performed by Station 23
C1267	Started at 11:03:23
C1267	Completed at 11:25:06
O9321	Performed by Station 11
O9321	Started at 12:22:45
O9321	Tool 35353 failure
O9321	Completed at 13:20:12

Figure 5.11: An example of Pallet Data message.

Simulating error situations

Several error situations can occur when pallet transports are executed:

- transport vehicles can break down;
- pallet buffers can become inaccessible.

When these situations occur the Pallet Transport Module is informed and the DFMS component that requested the pallet transport will be sent a Pallet Transport Error message. It is the responsibility of the Pallet Transport System to fix the problem and to inform the Pallet Transport Module when the failed components have become available again. The operator can simulate failure and recovery of the Pallet Transport System by choosing appropriate options from the user menu.

5.8 Load Module

The Load Module in the DFMS prototype has been implemented on a PC together with simulation modules for the operator activities and for the local pallet buffer. The relation between the Load Module and the simulation modules is shown in figure 5.12. The functionality of the Load Module and the simulator modules will be discussed separately.

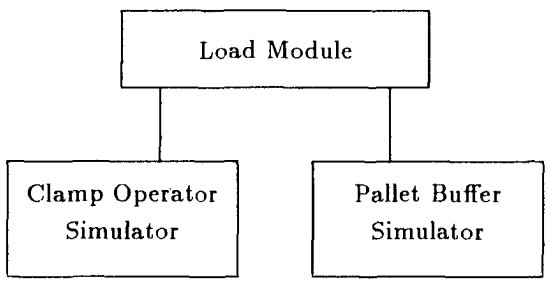


Figure 5.12: The relation between Load Module and simulation modules.

5.8.1 Implemented Load Module functionality

The Load Module is responsible for introducing new products into the system. A simple load algorithm selecting products in the sequence in which they occur in the order list is implemented. Limited availability of pallets and fixtures is taken into account. The Load Module accepts requests from Station Managers to unclamp products. All clamp and unclamp operations are entered into the Operation Queue. Orders from this Operation Queue are issued to the clamping stations.

5.8.2 Simulating the local pallet buffer

The clamp and unclamp operations are performed on clamping stations. The clamping stations are standard pallet buffer positions from which easy access to fixtures and other tools is possible. A Pallet Buffer Simulator is used to simulate the functionality of a pallet buffer. The Load Module can request the Pallet Transport Module to deliver or to remove pallets. The actual delivery or removal of pallets is handled by the Pallet Buffer Simulator. When pallets arrive or are removed, the Load Module is informed.

5.8.3 Simulating operator clamp/unclamp activities

The Load Module issues commands to perform clamp and unclamp operations. The operations are simulated by scheduling delays. After a delay has elapsed the Load Module is informed. The number of operations that can be performed simultaneously depends on the available number of clamp positions.

5.9 Station Manager

The Station Manager in the DFMS prototype has been implemented on a PC together with a module that simulates the behavior of the CNC. The relation between the Station Manager and the simulation module is shown in figure 5.13. The functionality of the Station Manager and the simulator module will be discussed separately.

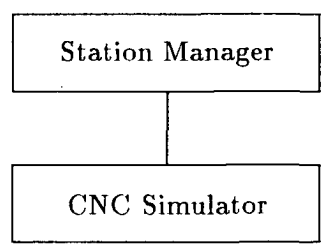


Figure 5.13: The relation between Station Manager and CNC simulation module.

5.9.1 Implemented Station Manager functionality

The Station Manager is the most important component of the DFMS architecture. The implementation of the Station Manager is in accordance with the architectural specification and has been designed to allow the developed software to be used in a first prototype of a programmable CNC.

Each Station Manager maintains an Operation Queue that contains operations that will be performed by the machine. Operations in the Operation Queue are processed on a First Come First Served basis. New operations are added to the Operation Queue through a process of negotiation. The Station Managers analyze the tool requirements and request the Tool Module to reserve and to deliver tools when needed. The Pallet Transport Module is requested to deliver pallets shortly before the operation is about to start. The Station Manager issues orders to the CNC to perform the operations in the Operation Queue.

5.9.2 The CNC Simulator

The CNC receives its instructions from the Station Manager. The CNC simulator is able to interpret these instructions and to simulate the reactions to them. The CNC also interacts with the Tool and Pallet Transport Systems.

Simulating the operation

The CNC simulator executes a NC part program by selecting the necessary tools from the local tool store. For each tool that is selected a delay is scheduled that simulates the time the tool is used. If no tool of the required Tool Class is available, the CNC simulator informs the Station Manager and the operation is suspended. When the last tool has been used for the current part program the CNC signals the completion of the operation.

Simulating tool arrivals and removals

When new tools arrive at the machine, the CNC simulator receives a Tool In message from the Tool Module. The CNC simulator informs the Station Manager by means of a New Tool message. The Station Manager replies by sending a Exchange Tool message, in which the tool that has to be replaced is specified. The CNC simulator then simulates the removal of the tool by sending a Tool Out message to the Tool Module.

Simulating pallet arrivals and removals

The CNC simulator receives a Pallet In message from the Pallet Transport Module when a pallet is delivered at the pallet buffer of the machine. The CNC Simulator then sends a New Pallet message to the Station Manager. The transfer of the Product Script is simulated by a Pallet Data message that is sent by the Pallet Transport Module to the Station Manager. The CNC simulator sends a Script Update message to the Pallet Transport Module as soon as the contents of the Product Script are updated.

When a pallet is removed from the local pallet buffer, the Pallet Transport Module sends a Pallet Out message to the CNC Simulator. To inform the Station Manager of the event, the CNC sends a Pallet Removal message to the Station Manager.

Simulating operation failures

The CNC simulator allows the simulation of tool failures. The operator who runs the prototype can cause a tool to fail by selecting an option from the menu. The CNC simulator will try to replace the tool and if no replacement is available will inform the Station Manager by sending a 'Tool Missing' message. The operation is suspended and can be resumed or aborted by the Station Manager.

The operator can also simulate the situation that a tool fails and that the product is damaged. In this case the operation will be aborted and the Station Manager will be informed.

5.10 Simulating crashes of DFMS components

The DFMS architecture has been designed to be very robust. Crashes on one DFMS component are not allowed to cause problems on other DFMS components. In the chapter on DFMS architecture the crashing of DFMS components is discussed in detail.

In order to test robustness of the DFMS architecture, crashes of Function Modules and Station Managers can be simulated. A special crash procedure has been designed to circumvent the problems caused by the differences between the DFMS prototype and a real DFMS system with real machines, pallets and tools.

Before the crash procedure will be discussed, the reason why a special crash procedure is needed will be explained. In order to understand this

explanation it is necessary first to discuss some network details.

5.10.1 Some details about the DecNet network

In DFMS the network is used to transfer messages that are defined in the architecture between DFMS components. In addition, in the DFMS prototype special messages are transferred to simulate the transport of pallets and tools.

The DecNet implementation of the DFMS Network guarantees that messages are *received* correctly. The DecNet network can *not* guarantee that received messages are also *processed* correctly. To understand the distinction between receiving and processing, it is necessary to inspect the inner operations of the DecNet Layer.

When a message is sent between two DFMS components, the Network Layers of the components communicate in order to guarantee the error free transfer of the message. The Network Layers conclude that the transfer of the message was successful when the message is received intact by the destination Network Layer. The DFMS component that sent the message is informed by its Network Layer that the transfer was successful.

However, the message that was sent is only entered into the *Network Queue* of the destination Network Layer. The remote DFMS application still has to process the message. If at this point the remote DFMS application crashes, all messages that are in the Network Queue and that are not yet processed by the DFMS component are lost. So even though the sender of the message assumes that the message has been processed, this may actually not be the case! The distinction between receiving and processing of messages only becomes apparent when a DFMS component crashes and if at that time the Network Queue of its Network Layer is not empty.

5.10.2 Performing the crash procedure

For standard DFMS messages the distinction between receiving and processing messages is not an important one. When a DFMS component crashes, the other components remain functioning independently. When the crashed component is later restarted all data structures are rebuilt from scratch. At that point it is of no importance whether any messages might have been pending in the Network Queue at the time the component crashed.

For simulation messages the situation is entirely different. If for example the Pallet Transport Module sends a Pallet Out message to a Station Manager and if the Station Manager crashes in an uncontrolled fashion, it

is possible that the Pallet Transport Module assumes that the sending of the message was successful even though the Station Manager did not process the message. If later on the Station Manager is restarted, some pallets will occur twice in the system[9]. In order to prevent these problems, DFMS components are 'crashed' by executing a special crash procedure:

1. The Network Layer is instructed not to accept any new messages. From that point on it is not possible for any remote Network Layer to successfully transfer a message to the component;
2. All messages being in the Network Queue at this moment are processed;
3. When no more messages are left, a configuration file is saved on disk to allow a restart of the DFMS component with the correct data;
4. The crash of the DFMS component is simulated.

The effect of this crash procedure is that the crash is not always performed immediately, but only after a short delay. As far as the standard DFMS Network traffic is concerned, the crash occurs at a completely random moment. This crash procedure makes possible an accurate assessment of the robustness of the DFMS architecture.

5.11 Testing the DFMS prototype

The DFMS prototype has been tested and analyzed to:

- detect behavior that is not in accordance with the architectural specifications, i.e caused by implementation bugs;
- detect behavior that, although in accordance with the specifications, is not correct, i.e caused by erroneous specifications.

The result of extensive testing is that at this point of time the architectural specifications do not contain any apparent errors. Implementation errors have been removed as far as they have been discovered. However, due to the amount of code that has been generated for the prototype, approximately

[9]Station Managers have a configuration file in which the pallets and tools are listed that are available at the machine. The configuration is read when the Station Manager is initialized and is written when the Station Manager shuts down or 'crashes'.

50.000 lines of C code, it is to be expected that the software still contains numerous bugs. It is estimated that the removal of these bugs will take a considerable amount of time. Because the main reason to develop the DFMS prototype is to analyze the technical feasibility and because this has been proven, it is not considered worthwhile to invest further time in debugging the prototype. In the following sections the tests that have been performed are discussed.

5.11.1 Testing the system under normal conditions

The DFMS prototype has been tested extensively to analyze its behavior under standard conditions. The test procedure that has been followed, was to initialize and then to introduce a set of orders into the system and to observe the operations of all DFMS components. By changing the orders a large number of different situations have been tested.

5.11.2 Testing the system under special conditions

The reaction of the DFMS system to non standard situations have been tested. Examples of non standard situations that have been tested are:

- Station Managers are added to an active DFMS system;
- Station Managers are removed from an active DFMS system;
- the Tool Module is not able to reserve tools within the specified time;
- the Tool Module is not able to deliver tools within the specified time.

5.11.3 Testing the capacity of the control system

To test the capacity of the DFMS control system, the average duration of operations in the system has been varied from many minutes down to 10 seconds. The DFMS control system has proven to be able to perform all necessary functions, even when products with short operation durations are introduced into the system. Unacceptable delays caused by overload of the network or by overload of the PC's have never occurred. The fact that no NC part programs are downloaded from the Program Module to the Station Managers evidently leads to a more optimistic view of the available network capacity than is realistic. However, because modern CNCs are able to contain quite a large number of NC part programs, it is expected that a large amount of the required programs can be stored locally in the CNC.

After the system has been running for some time, downloading of programs will then only have to take place for special programs[10].

5.11.4 Testing the robustness of the system

To test the robustness of the DFMS architecture a large number of tests have been performed in which failures of system components were simulated. The following situations have been extensively tested:

- a tool fails during an operation;
- a product is damaged by a tool failure;
- a transport vehicle breaks down;
- a pallet buffer becomes unreachable;
- a Station Manager fails completely;
- the Pallet Transport Module fails completely;
- the Tool Module fails completely.

The reaction of the DFMS prototype on these error conditions is in accordance with the expectations based on the architectural description.

5.11.5 Test Results

The DFMS prototype has successfully passed all tests it has been submitted to. The tests add to the evidence that the DFMS architecture is capable of performing the functions that are expected from a control system for Flexible Manufacturing Systems[11].

Furthermore, the tests show that an implementation of the DFMS architecture using:

- IBM compatible personal computers with 640 kbyte internal memory;
- the MS-DOS operating system;
- and DecNet DOS communication software;

[10] Even if programs have to be downloaded frequently the capacity of the network is expected to be large enough not to cause problems.

[11] The primary evidence of the capabilities of the DFMS concept is the architectural specification. By merely reasoning about these specifications, it is possible to form a judgement over the capabilities of DFMS.

is possible and that no special problems are encountered[12].

The conclusion that the MS-DOS operating system, in spite of its limited capabilities, can be used to realize the DFMS architecture, is of great importance. It means that no special equipment or technical skills are required to develop DFMS components. The standard MS-DOS software development tools can be used.

[12]The Station Manager implementation (in combination with the CNC Simulator) has been known to crash when only 512 kbyte of internal memory is available.

Chapter 6

DFMS dispatching and loading

The DFMS architecture uses simple algorithms to determine the sequence of operations on the machines. The possible disadvantage of using these simple algorithms is that in some applications the performance of the system will not be satisfactory. In this chapter the consequences of using simple algorithms are discussed.

6.1 System Performance

Several criteria can be used to express the performance of a Flexible Manufacturing System. In [Lenz 86] the performance of FMS is discussed in terms of system capacity and work in progress. In this chapter a slightly different set of criteria is used. When the system under consideration regularly receives a set of orders that has to be processed before the following set of orders is accepted, a combination of the following two criteria is used[1]:

- **Makespan**
  The makespan is defined as the time it takes to completely process a set of orders. A control system influences the makespan by making decisions about the sequence of operations on machines. Good decisions lead to a low makespan, poor decisions lead to a high one;

[1] When the system receives its orders in a continuous stream instead of in a sequence of order sets, the makespan criterion can better be replaced by the *maximum lateness* or *mean tardiness* criterion.

- **Resource Requirements**
  The amount of resources required to produce a set of orders determines the efficiency of the system. Examples of resources that are expensive are pallets, fixtures and tools. It is important that a control system utilizes these resources as efficiently as possible. Efficient use of resources leads to low resource requirements.

In figure 6.1 the relation between makespan and resource requirements is schematically shown.

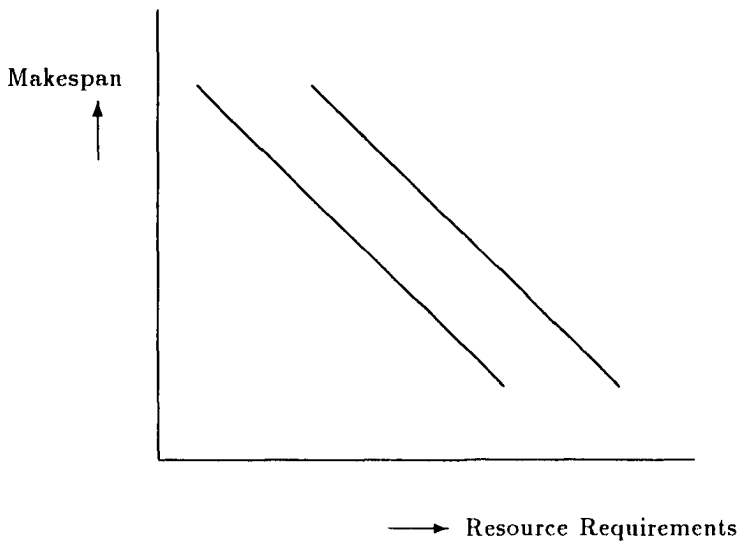


Figure 6.1: Qualitative overview of the relation between makespan and resource requirements. The line to the left represents an efficient FMS. The line to the right represents a less efficient FMS that requires more resources to obtain an identical makespan.

The frequently used *machine utilization* criterion is not very valuable. It would be reasonable to use the machine utilization criterion if the FMS could be regarded as a completely isolated system and if no due dates were attached to the products. This however is generally not true: products manufactured in the FMS have to be ready before a certain due date, because they are needed for subsequent process steps. If the products are not available in time, the higher level scheduling is disturbed which leads to high

additional costs. When analyzing the performance of an FMS, the following has to be taken into consideration:

- The characteristics of the production problem for which the FMS is defined influence the performance of the system greatly. If a very diverse set of products, requiring a large number of different tools and fixtures has to be processed, it will be difficult to realize a high system performance;

- The higher level scheduling system (HLS) determines which products have to be processed by the DFMS in the coming scheduling period. Input for the HLS is a set of orders that has to be processed before a specified, not necessarily common, due date. For each scheduling period a set of orders is chosen. The HLS tries to choose the set of orders so that the workload for the system will be balanced. In order to be able to estimate the workload, the HLS uses a model of the DFMS;

- The DFMS Load Module determines the sequence in which products are introduced into the system. Input for the Load Module is a set of orders for each scheduling period. The Load Module tries to find the optimal sequence in which to introduce products into the system. For the Load Module it is important to minimize the time spent at building fixtures;

- The DFMS dispatching method determines the sequence of operations of products that are already introduced in the system.

The influence of the last three factors on the DFMS system performance will be discussed in the following sections.

6.2 The higher level scheduling system

The higher level scheduling system is considered to be outside the scope of an FMS control system. A number of important assumptions concerning the interface of a DFMS system with the higher level scheduling system are discussed below:

- **Balancing of the system**
  The set of orders issued by the higher level scheduling system should

contain a workload that can be processed by the DFMS system and that will preferably lead to an acceptable utilization of the machines in the system. A poor system performance caused by a poorly balanced workload is not the responsibility of DFMS;

- **Operation Class allocation**
  The higher level scheduling system is responsible for determining which machines have to support the required Operation Classes. If the workload of the order sets varies considerably, the higher level scheduling system has to specify for each new scheduling period which Operation Classes each machine has to support. The higher level scheduling system may have to use a DFMS simulation system to determine the optimal Operation Class distribution;

- **Operation Class limitation**
  The number of different Operation Classes that can be supported simultaneously on a machine is limited due to the limited number of tools that can be stored in the tool store of the machine. If there are many Operation Classes defined in the DFMS system, this can lead to the restriction that during a particular scheduling period not all Operation Classes can be supported. Only products requiring operations belonging to supported Operation Classes can be produced during a particular scheduling period. During *different scheduling periods* the system would then be able to produce *different subsets* of the total product variety. It is up to the higher level scheduling system to determine which product subset will be supported during a particular scheduling period.

  The problem of the Operation Class limitation can be minimized by defining *small* Operation Classes. The disadvantage of large, loosely defined Operation Classes is that many tools have to be available, but that only relatively few tools are actually used.

6.3 Loading in DFMS

It is the responsibility of the Load Module to determine the sequence in which products are introduced into the system. Several algorithms to determine this sequence can be used. A restriction that applies to all algorithms is that the number of products that can be introduced into the system is

limited by the available number of pallets and the available number of fixtures.

6.3.1 Simple load algorithms in DFMS

The Load Module uses simple algorithms to determine the sequence in which products have to be introduced into the system. The following considerations apply:

- The sequence in which products are introduced into the system *influences* the efficiency of the system;
- Fixtures are expensive. It is an advantage if only few fixtures are necessary. The fixtures that are used must be used intensively;
- Assembling fixtures is an activity that requires time. In order to prevent the clamp stations from being overloaded, it is important to minimize the rebuilding of fixtures.

Several load algorithms can be applied:

- **Rigid Linear Top Bottom**
  This procedure introduces products in the sequence in which they occur in the Order List. When no free fixture is available for the first product in the list to be clamped, no other product is clamped. The system waits until a fixture becomes available. This procedure depends heavily on the sequence of the orders in the Order List and will generally only work satisfactorily, if the list has been prepared by a higher level loading system that is aware of the capabilities of the manufacturing system;
- **Loose Linear Top Bottom**
  This procedure is a variant on the *Rigid Linear Top Bottom* method. If no fixture is available for the first product in the list, the next product is tried. The system tries to avoid that pallets are not used;
- **Minimized Fixture Rebuild**
  This procedure tries to avoid the rebuilding of fixtures by only removing a fixture from a pallet when no product in the product list requires the fixture. When a pallet is returned from the system, the Order List is scanned to find the first product that can be clamped on that particular fixture. If no product is found, the fixture is removed

and replaced by the fixture that is required for the first product in the Order List;

- **Optimal Fixture Distribution**
  The procedure calculates the optimal number of fixtures for each fixture type and tries to maintain this optimal fixture distribution. When a new fixture has to be mounted on a pallet, the fixture that minimizes the difference between the current and optimal fixture distribution is chosen. The optimal fixture distribution is determined by calculating the total number of required minutes per fixture type. The distribution is then scaled to the number of pallets. The procedure will only be meaningful if the number of fixture types required for the current Order List is considerably smaller than the number of pallets available in the system;

- **Balanced Machine Load**
  The procedure takes the current load of the machines into account, when a new product is to be introduced into the system. The system prefers to introduce a product that needs an operation that can be performed on the machine that is currently the least occupied. This method stresses the influence of the Load Module on the balancing of the system. Disadvantage of this method is that the Load Module requires the Operation Queue information of all Station Managers. In the current specification of the DFMS architecture, this information is not available;

- **Parameterized Balanced Machine Load**
  This procedure is based on a simple method described by Zeestraten and Moonen [Moo 88]. The loads of all machines are expressed as a set of parameters that can easily be calculated with data that are available to the Load Module[2]. The goal of the procedure is to minimize the difference between the loads of separate machines. When a new product is to be introduced, the procedure calculates for all products how they affect the load parameters for the machines. The product choice that results in the best balancing of the load parameters is chosen. Disadvantage of the method is that all operations that have to be performed on a product contribute to the load parameters, even when

[2]The Load Module 'knows' which products have been introduced into the system and can therefore estimate the system load.

the first few operations already have been performed and therefore no longer contribute to the actual system load.

In an implemented system it is necessary to use one particular load algorithm. Tests have shown that no single algorithm leads to the best results in all situations[3]. One way of dealing with this uncertainty is to use a DFMS simulation system to determine for each particular order set, which algorithm gives the best results. The chosen algorithm can than be applied when the actual production starts.

An important restriction that the load algorithm has to take into consideration is the limited amount of pallets and fixtures that are available. When a large number of pallets and fixtures are available the problem that has to be solved is generally simpler: less restrictions have to be dealt with. However, a large number of pallets represents a considerable investment and may not always be acceptable.

6.3.2 Task Selection Loading

The algorithms discussed in the previous section have the advantage that they are very simple, but it is not guaranteed that they always will find a load sequence that leads to acceptable results.

Zeestraten [Zee 89] has developed a loading algorithm, called the *Task Selection Function* (TSF) that can very well be used in DFMS and that applies linear programming techniques to determine an optimal loading sequence.

The concept of an *Order List* is not really supported in the TSF method. The work that has to be performed during a particular scheduling period is specified as a set of *tasks.* A task consists of a clamp and unclamp operation, with in between one or more 'standard' operations. The TSF algorithm allocates tasks to available pallets. The result of the TSF algorithm is a set of tasks that have to be performed for every pallet in the system.

When this method is applied, the role of the DFMS Load Module is reduced to performing the clamp and unclamp operations that have been defined for each pallet. The relation between the Load Module in DFMS and the external task selection function is shown in figure 6.2.

[3]These tests have been performed with the DFMS simulation system that will be described in chapter 7.

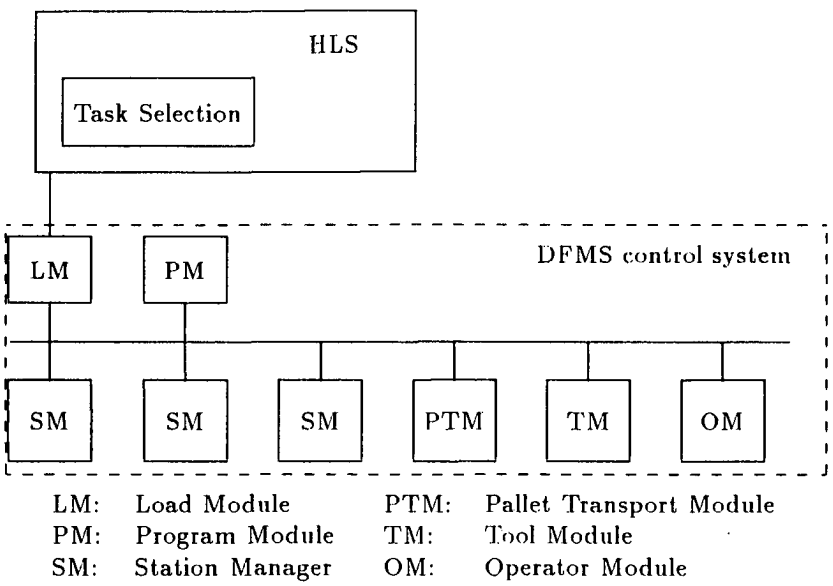


Figure 6.2: The relation between DFMS and Task Selection Module.

6.4 Dispatching in DFMS

6.4.1 Principle of priority dispatching rules

Many traditional FMS are scheduled using simple scheduling methods. Frequently the scheduling is based on the application of *priority dispatching rules*. A schedule is made by *simulating* the operations in the system. Whenever a machine finishes an operation, a priority dispatching rule is used to determine which pallet has to be processed next on that machine. An example of a priority dispatching rule is the *Shortest Processing Time* (SPT) rule: 'from the pallets waiting for an operation that can be performed by the machine, choose the pallet for which the next operation length in minutes is the shortest'. Another example of a dispatching rule is the *First Come First Served* (FCFS) rule: 'from the pallets waiting for an operation that can be performed by the machine, choose the pallet that has been waiting longest'. By applying these rules repeatedly a sequence of operations can be determined for all machines simultaneously. The sequence of operations that is obtained this way forms the schedule that the dispatcher later on will try to execute as precisely as possible.

Traditionally, scheduling systems based on priority dispatching rules were developed for job shop problems without routing flexibility: for each operation to be performed only one machine can be chosen. The scheduling problem that has to be solved in DFMS (as in any FMS) can be characterized as a job shop scheduling problem *with* routing flexibility. The usual priority dispatching rules can still be applied in this case with only slight modifications. The general procedure of applying priority dispatching rules for the scheduling of a job shop with routing flexibility is described below:

- The system maintains a set of operations waiting to be processed;
- The set is divided in subsets of operations that can be processed on each machine[1];
- When a machine finishes an operation on a product and if the product requires a new operation, the operation is added to the set of operations waiting to be processed;
- When a machine finishes an operation, a new operation is chosen from the subset for that particular machine. Which product is chosen de-

[1] If an operation can be performed on more than one machine, the product will occur in more than one subset.

pends on the applied dispatching rule. The set (and subsets) of operations are updated.

Each subset can be interpreted as a priority ordered queue. If a new operation is added to the subset, it is inserted in the queue according to its priority. When an operation is chosen from the subset, the operation at the top of the queue (having the highest priority) is selected.

6.4.2 The DFMS dispatching rule

In the standard scheduling systems based on priority dispatching rules all operations waiting to be processed are members of *one* set. In DFMS a *distributed* set of operations waiting to be performed is maintained. In this case for every machine in the system a set of operations that will be performed by that machine exists. When an operation finishes on a machine and if further operations are required on the product, the system chooses a machine that will be responsible for performing the operation and adds the operation to the operation set of the machine. The machine that has finished an operation chooses a new operation from its own operation subset. In DFMS the subset of operations for a particular machine is called an *Operation Queue*.

The DFMS system applies a simple rule to determine which machine will perform the next operation on a product. A formulation of this rule is: 'When a pallet finishes an operation on a machine, allocate the next operation on the pallet to the machine that *is able to perform the operation* and that has the smallest subset of operations at that time'. Part of the rule definition has been emphasized: 'being able to perform the operation' is a critical issue in DFMS. A machine is only able to perform an operation when it supports the Operation Class to which the operation belongs. If more than one machine support a particular Operation Class, the system assumes that they are equally capable of performing the operation. The only criterion used by the system to choose between machines that support the Operation Class is the length of the Operation Queues of the machines. The operation is allocated to the machine with the shortest Operation Queue in order to achieve a balanced system load.

The DFMS control system applies the First Come First Served dispatching rule to determine which operation is chosen from the Operation Queue when a machine has become idle. The consequence of applying this rule is that the relative position of one operation with respect to other operations

never changes. This characteristic makes it possible to analyze the tool requirements of the operations and to reserve tools before they are actually needed. If operations could be inserted somewhere in the queue, the tool reservations made previously would be invalidated.

A very important characteristic of the DFMS dispatching process is that the sequence of operations on the machines is determined *without requiring a schedule that has been previously made*. The DFMS dispatching process is not executed *before* the actual production starts, but *during* the actual production.

6.4.3 Performance of priority dispatching rules

Different priority dispatching rules lead to different machine utilization and therefore to differences in the total time taken to process a given set of orders. Although much research has been dedicated to this problem, it remains difficult to compare priority dispatching rules. If the priority dispatching rules are analyzed with respect to the various criteria, their relative performance differ with the criterion applied. The SPT rule for example, gives good results if the average lead time criterion is used; in contrast SPT performs rather poorly if maximum lateness is considered. There is not one priority dispatching rule that functions best in all situations.

In [Zee 88] a comparison has been made between a number of priority dispatching rules applied to the job-shop scheduling problem with routing flexibility. In this paper the rules were judged on the *makespan* criterion. One of the rules tested was the (extended) First Come First Served rule. The makespan found for this rule for a number of randomly generated test cases was on the average 20% higher than the makespan obtained by the most successful rule. The makespan found for the very commonly used SPT rule was 27% higher than the makespan obtained by the most successful rule.

Some of the rules tested in this study however, performed considerably better. For example the *Most Operations Remaining* (MOPR) rule had a makespan only 4% higher than the optimal makespan. The criterion used in the MOPR rule is: 'from the pallets waiting for an operation that can be performed by the machine, choose the pallet that has the largest number of remaining operations'. To apply the MOPR rule successfully, all operations that will be performed by each pallet during the particular scheduling period have to be known beforehand[5]. In this respect the MOPR rule differs from

[5]It is possible to apply the rule while only taking the operations of the current product

the simpler rules like SPT and FCFS.

6.4.4 DFMS dispatching versus traditional scheduling

When comparing the scheduling methods applied in traditional FMS and the dispatching method used in DFMS, similarities and differences can be recognized.

Many traditional FMS control systems are scheduled by applying simple priority dispatching rules like SPT or FCFS. The operations waiting to be processed are members of *one* set from which can be chosen when a machine becomes available. The dispatching rule is applied on a *global* set of operations.

In DFMS a product is allocated to a machine immediately it requires an operation. The criterion for allocating an operation to a machine is the wish to balance the operation load evenly over the machines. When a machine becomes idle, a dispatching rule is applied to choose an operation from the *local* operation set. In DFMS, the FCFS rule is applied.

Another very important difference between traditional FMS and DFMS, is that the traditional FMS control systems generate *real* schedules, to which detailed anticipation is possible. Before the production starts it is known which tools are required on a particular machine and at what time. In DFMS it is only possible to anticipate on a *much less detailed* scale: it is known *which* products will be processed during a particular scheduling period, but not exactly *when*. In DFMS this does not lead to problems because of application of the Operation Class concept.

6.4.5 Operation Classes in DFMS

Local, extended and virtual tool sets

In DFMS the concept of *Operation Class* is introduced. A machine supports a particular Operation Class if the tools necessary to perform the operations are available. Machines in DFMS have access to a *virtual tool set*. A virtual tool set consists of the *local* and the *extended* tool set. The local tool set is available at the machine, the extended tool set is stored in the central tool store but is allocated to a particular Station Manager. Usually, the extended tool set will be much smaller than the local tool set. To support an Operation Class the Station Manager has to have a virtual tool set that contains all

into account. This however, will lead to less optimal results.

tools required to perform the operations defined for the Operation Class. If the extended tool set is small, only few tool transports will be necessary. In figure 6.3 the relation between local, extended and virtual tool sets is shown.

Station Managers have knowledge of both the local and the extended tool set. When tools are reserved for a Station Manager the tools are added to the extended tool set of that Station Manager. When tools are delivered to the machines they are transferred from the extended tool set to the local tool set. The tools that are exchanged are either removed from the system or are added to the common tool set. When tools have been allocated to the virtual tool set of a Station Manager, they can no longer be used by other Station Managers. It is not known when the tools will become available again, if ever.

The common tool set contains tools that have not yet been allocated to a particular virtual tool set; they can be allocated to any Station Manager that may request them. When tools are reserved that are not available in the common tool set, the tool room operators will assemble and prepare a new tool. Because tools are reserved some time before they are actually needed, their preparation need not delay the operation. In order to prevent tools from being allocated longer than necessary, they are only reserved by Station Managers if they are needed for operations that will take place in the near future.

Operation Classes and tool management

The dispatching method used in DFMS has some consequences for the tool management. The fact that in DFMS, unlike in traditionally scheduled FMS, it is not known exactly which tools are needed on each machine at a particular time, requires a special approach. If it can be guaranteed that the virtual tool set of each machine can be defined to contain the tools required to support the Operation Classes no problems are expected. Under normal conditions this will always be possible. An additional condition to prevent the tool transport system from being overloaded is that the extended tool set should be much smaller than the local tool set.

It is not expected that the tool management in DFMS will be less efficient than in traditional FMS. As has been explained earlier, in DFMS tools cannot be shared between machines, i.e. virtual tool sets cannot overlap. This can be interpreted to be a disadvantage compared to the tool management in traditional FMS control systems. However, very few of the traditional systems are known to be able to share tools between machines.

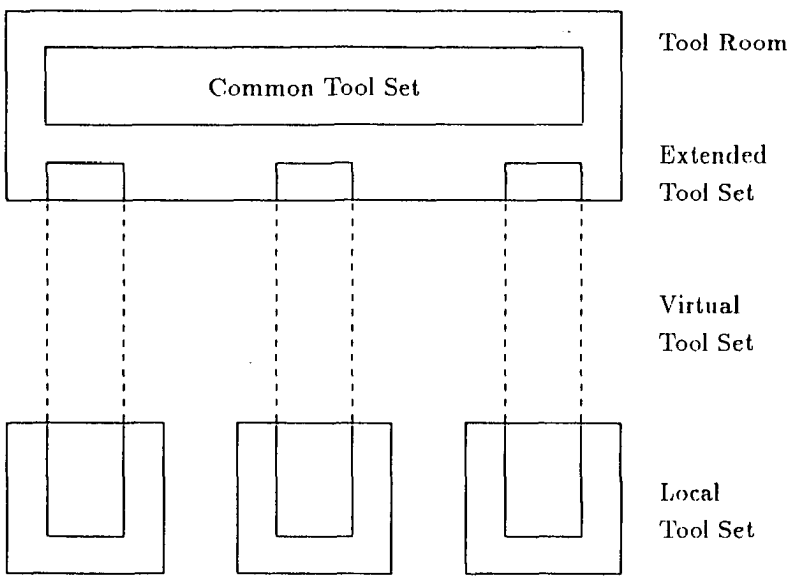


Figure 6.3: Virtual tool sets in DFMS. The virtual tool set consists of the local and extended tool sets.

Furthermore, the complications involved in sharing tools are probably difficult to overcome. If for example, due to unexpected circumstances a tool that is needed on another machine is not yet available, operations on that other machine can not proceed. Sharing tools introduces exactly the type of dependencies between machines that causes the problems in traditionally scheduled systems.

From the fact that tools cannot be shared between machines, it follows that if Operation Classes require tool sets for which expensive, special purpose tools are required, those Operation Classes should be supported by just as few Station Managers to have enough capacity to handle the required volume of products of those Operation Classes.

6.5 Increasing DFMS performance

For most applications the performance of DFMS will not be considerably less than of traditional FMS control systems. If however, the wish exists to increase the performance of DFMS, it is possible to adapt the DFMS architecture. Unfortunately, some features of the DFMS architecture will be lost. In this section some adaptations and their consequences are discussed.

6.5.1 Applying alternative dispatching rules

In DFMS it is possible to use alternative *local* dispatching rules; i.e. rules that are used to select an operation from the operations already allocated to a machine. Instead of using the simple FCFS rule, more sophisticated rules could be used that are better able to optimize to criteria considered important.

Disadvantage of applying *any* rule other than FCFS in DFMS is that *tool reservations are no longer possible.* The Station Managers use an iterative procedure to determine the tool requirements of the operations in the Operation Queue. If a dispatching rule is used that is able to place a new operation anywhere in the Operation Queue, the sequence of operations is not fixed and tool reservations will have to be continuously changed. To prevent this problem form occurring, the tool reservation mechanism will have to be disabled when other dispatching rules than FCFS are used. In this situation there is no extended tool set defined: the virtual tool set is identical to the local tool set.

If tool reservations are disabled the Station Managers analyze the tool requirements of each operation shortly before the operation is expected to

start. If not enough tool capacity is available the Station Manager will request *delivery* of the tools. All tools that are requested for delivery should be available in the common tool set; the Tool Module does not have the opportunity to prepare new tools on such short notice. Tools that are requested (and that are available in the common tool set) are transferred directly from the common tool set to the local tool set.

The condition that all tools that can be requested are available in the common tool set is not unrealistic. The tool usage in every scheduling period can be estimated. It is not difficult to derive from this estimate how much tool capacity is missing. The tool room operators anticipate this and prepare the required tools before production starts. The new tools are added to the common tool set and are available for whichever Station Manager that might request them.

If one has a system in which the tool reservation facility is not required, one can apply most commonly used priority dispatching rules. To analyze the application of dispatching rules in DFMS the rules can be divided in two groups. For the rules of the first group no adaptations are required:

- First Come First Served (FCFS);
- Shortest Processing Time First (SPT);
- Longest Processing Time First (LPT).

The rules of the second group can best be applied when it is known which operations are to be performed by each pallet during the scheduling period under consideration. These rules can for example be used in combination with the TSF loading algorithm (see section 6.3.2):

- Fewest Operations Remaining (FOPR);
- Most Operations Remaining (MOPR);
- Shortest Remaining Processing Time (SRPT);
- Longest Remaining Processing Time (LRPT).

Rules based on due dates are not applicable in DFMS, because the due date is assumed to be identical for all products during a particular scheduling period. For an overview of the characteristics of the rules mentioned above refer to [Bla 82]. Some care should be taken in interpreting the performance of these rules, because:

- in DFMS the job shop problem *with* routing flexibility is solved;
- in DFMS the application of the dispatching rules is performed *after* the operations have been allocated to the machines.

6.5.2 Applying centralized dispatching

The methods discussed in the previous section are all based on first finding suitable machines to perform operations and then determining the sequence in which the machines perform those operations. This method can be characterized as *distributed dispatching*. A more conventional method is the *centralized dispatching*. In section 6.4.1 the concept of centralized dispatching is explained. Operations are maintained in a central set of operations to be performed. As soon as a machine becomes available, an operation is chosen from the subset of operations that can be performed by that machine. The dispatching system has an overview of all operations to be performed and can therefore make more optimal decisions than a distributed dispatching system can.

Central dispatching can be accommodated within the DFMS architecture as an extension. When a machine completes an operation, the normal procedure in DFMS is that the control over the product is passed to another machine and that the Station Manager selects a product from the products in the Operation Queue. When centralized dispatching is applied, a *Dispatching Module* is responsible for determining which operation will be performed by each machine. When a new operation is expected to start shortly, the Station Manager informs the Dispatching Module that a new operation can be accepted. The Dispatching Module selects a new operation. The Station Manager is still responsible for requesting delivery of tools and pallet. When an operation completes, the Station Manager informs the Dispatching Module. The Dispatching Module adds the operation to the central set of operations waiting to be performed. The DFMS can be made to switch between the central and distributed dispatching modes.

The disadvantage of centralized dispatching is that it is not possible to reserve tools some time before they are actually needed. This disadvantage has already been discussed in the previous section. However, the application of centralized dispatching has some advantages. The dispatching rules described above can all be used. Because they are now applied on a global, instead of on a local scale their characteristics will be more pronounced. Furthermore, it is possible to apply newly developed, highly sophisticated dispatching algorithms. An example of such an algorithm is the *Look Ahead Dispatching* algorithm developed by Zeestraten [Zee 88]. The algorithm can be compared with a chess program. When a dispatching decision has to be made, the system *simulates* the consequences of all possible dispatching decisions. An *evaluation function* is used to determine which initial decision

led to the 'best' end situation. The algorithm is reported to minimize the makespan of a set of orders even better than the MOPR algorithm does.

Chapter 7

The DFMS simulator

In principle, the DFMS architecture can be applied to any production problem normally handled by Flexible Manufacturing Systems. However, the DFMS architecture has some characteristics that might limit its use to a subset of FMS production problems. A DFMS simulator has been developed in order to:

- analyze for which set of FMS production problems DFMS is most suited;
- have a tool to design a DFMS system to handle a particular production problem.

In the first section of this chapter some of the issues that are of importance for the design of FMS for a particular production problem are discussed. The possibilities and advantages of applying simulation systems to the design of FMS are explained.

In the subsequent sections the specifications of the DFMS simulation system will be discussed. In the next chapter the problem of application of DFMS will be discussed in greater detail.

7.1 Designing FMS

In order to understand the specification of the DFMS simulator, it is necessary to analyze which factors influence the design of FMS. Three major influences can be distinguished:

- the production problem definition;

- the objectives of the system;
- the cost of FMS components.

These three factors of influence will now be discussed.

7.1.1 The production problem definition

Of major importance is the question which products have to be manufactured and in what quantities. The process planning department has to specify for each product:

- which operations are required;
- on which machines the operations can be performed;
- how long each operation takes;
- which tools are required.

The factory scheduling and planning departments have to specify:

- the average number of products for each product type per year;
- the average batchsize for each product type;
- the length of the scheduling period.

Furthermore it is important to know how many product types generally occur in a typical order set for a particular scheduling period. On the basis of this information it is possible to *estimate* how many machines will be required and which tools and fixtures have to be available.

The products that are to be processed in the system cannot be chosen at random. As has been explained before, it is essential that the products have similar production characteristics. It will be difficult to design an FMS that is able to operate efficiently, when a large number of the products require completely different tools and fixtures.

7.1.2 The objectives of the system

The FMS has to be able to manufacture the products for which it is designed. However, a number of important choices have to be taken into account. Three examples of choices:

- Is it acceptable that sometimes not all products from an order set can be processed before the scheduling period expires or is this always unacceptable?

- Is it acceptable that the number of *different* product types that can be processed during a particular scheduling period is limited or is it necessary that any product for which the system is designed can be manufactured at any time?

- Is it necessary to have excess capacity to process emergency orders?

The answers on these questions influence the design of the system considerably.

7.1.3 The cost of FMS components

The ultimate goal of an FMS installation is to contribute to the profits of the company. Given the constraints discussed in the previous two sections, this can best be achieved by, at a given performance, minimizing the total investment. In order to be able to minimize the total investment, it is essential to know the prices of all system components. For example, insisting on improving the machine utilization can be bad policy, if to obtain a small improvement, a large number of possibly very expensive pallets have to be used.

7.2 Applying simulation to FMS design

The design of FMS is a complex procedure. It is not possible to determine the optimal number of machines, pallets, fixtures and tools by mere calculation: many variables have to be taken into account and many dependencies exist between these variables. Examples of important variables in the design of FMS are:

- the types of machines;
- the number of machines of each type;
- the number of pallets;
- the type and number of fixtures;
- the required product transport capacity;
- the size of the local tool stores;
- the size of the central tool store.

The design of FMS consists of determining values for all relevant variables. Simulation is a very useful tool to aid the design of FMS [Brow 82], [Car 82], [Croo 85] and [Iwa 84].

Before the design of the FMS can proceed, the product problem definition data have to be available, the system objectives have to be defined and the approximate cost of components have to be known. The designer then determines start values for the system variables. In an iterative process the values of the system variables are determined:

1. A typical order set is fed to the simulation system. The processing of the order set is simulated;
2. The performance of the system is analyzed;
3. If the performance is satisfactorily, the initial design is ready, otherwise the designer chooses new values for some system parameters.

Once an initial design is ready, the system is tested with typical order sets from the real system. If the overall performance is still acceptable the design of the FMS is ready, otherwise some system variables are adapted and the procedure is performed again.

7.3 Functionality of the DFMS simulation system

The DFMS simulation system is a *model* of a real system. In order to keep the complexity of the model limited, details have been ignored and abstractions have been made. In this section the functionality of the DFMS simulator will be discussed. For all major DFMS components will be discussed which part of the behavior is considered to be relevant and which functionality is implemented in the DFMS simulation system.

7.3.1 Pallet Transport System

In the design of DFMS the capacity of the Pallet Transport System is an important issue. The Station Managers request delivery of pallets shortly before the pallets are expected to be needed. The capacity of the Pallet Transport System has to be large enough, in order to prevent delays at the machines.

The DFMS simulation system enables a rough estimation of the required transport capacity. The number of transport vehicles and the average transport time can be chosen by the operator. In the current version of the simulator, no lay-out information is used; this limits the precision of the estimation[1].

The DFMS simulator presents the operator with data about the performance of the pallet transport system. Based on these data the operator can decide whether the pallet situation is acceptable or whether the system definition has to be changed.

7.3.2 Tool Module

A critical aspect of the design of a DFMS application is the tool management. The concept of Operation Classes have been introduced to facilitate tool management. Operations are combined into Operation Classes based on their similar tool usage. Station Managers perform only operations that are member of the Operation Classes that are supported by the Station Managers. This organization was developed in order to minimize the number of tool changes.

When Station Managers reserve tools for a particular order, the simulation system tries to allocate existing tools to the order. If no tools are available, new tools will be assembled. The total time required to assemble all tools for an order is calculated. Based on the capacity of the tool preset stations and on the current number of tool preset orders, an estimate is made of the time the tools will be available. The Station Managers that reserved the tools *always* assume that the tools will be available and do not check if the time the tools are available is acceptable.

When the Station Manager requests delivery of earlier reserved tools, the simulation system checks whether the tools are already available. If they are not, a tool delivery problem occurs. The event is noted and the machine waits until the tools become available[2].

The operator can determine the initial contents of the central tool store and the local tool stores. For each tool the assembly time is known. The number of tool preset stations and the average tool transport time are variable.

The system presents the operator with data concerning the distribution

[1] Version 1.0, November 1988.

[2] This situation is handled differently in a real DFMS system. There, the Station Manager has the option to skip an operation if preparations take too long.

of tool reservation requests. The tool management situation will generally be considered acceptable when a limited amount of tools have to be exchanged to perform all required operations. Even in situations that a large amount of tools have to be exchanged, the situation may be judged acceptable if no further minimalization of tool exchanges is possible and if the required tool transport capacity is available.

The operator can decide from the data presented by the tool simulation system, whether the tool situation is acceptable or whether the system definition has to be changed.

In the current version of the DFMS simulator it is not possible to precisely model the tool transport system. Only a rough estimate of the required capacity can be made.

7.3.3 Station Manager

In the design of a DFMS application a number of decisions concerning machines have to be made:

- How many machines of particular machine types are available;
- What are the sizes of their tool stores;
- Which Operation Classes do they support;
- How many pallet buffers do they have.

The operator is presented with data concerning machine usage, required tool changes and decides from these data whether the definition of the system will have to be adapted.

The full Station Manager functionality is simulated in the DFMS simulation system.

The DFMS simulator presents the operator with data concerning machine utilization, product lead time and system balancing. These data help the operator determine system parameters.

7.3.4 Load Module

For the simulation of a DFMS application, the Load Module is a critical component. The Load Module determines in which sequence products are introduced into the system. Inputs for the Load Module are the orders that have to be processed, the list of available fixtures and the list of available pallets. The Load Module controls a number of clamp/unclamp stations where products are clamped on or removed from pallets. The simulation

system allows the operator to choose the load algorithms that have been specified in section 6.3. The number of pallets and fixtures that are available in the system are inputs for the simulation system.

The simulator presents the operator with data concerning the performance of the Load Module. The effectiveness of the chosen load algorithm can be analyzed and the required capacity of the clamping stations can be determined.

7.3.5 Program Module

The Program Module is not implemented in the DFMS simulator. In DFMS the only function of the Program Module is to download NC part programs and Tool Requirements Lists on request of Station Managers. In the simulation system, operations are simulated by scheduling delays: the actual NC part programs are not required.

The Tool Requirements Lists are globally available to all Station Managers, there is no need to explicitly make them available.

7.3.6 Operator Module

The Operator Module does not have an important function in a normally operating DFMS. Its functions are mainly requested when problems occur in the system. The DFMS simulator has been designed for quantitative, static analysis of DFMS operation; error handling is not analyzed by the DFMS simulator[3]. No Operator Module functionality is implemented in the DFMS simulator.

7.4 Input of the DFMS simulator

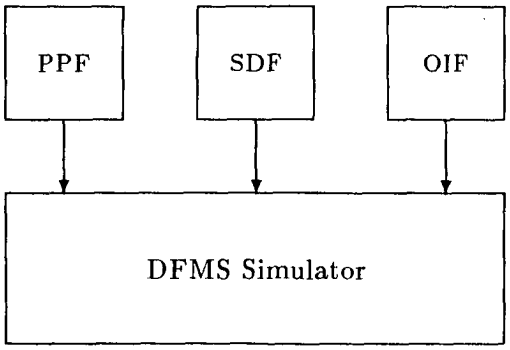
The DFMS simulator has three types of input:

- information about the production problem;
- characteristics of the system to be simulated;
- information about the orders to be processed.

The production problem data are fixed for a particular problem. The goal of the simulation is to find the optimal system characteristics. The production problem data, the system characteristics and the order information are

[3]Error handling is an important topic in the DFMS prototype.

specified in three separate files. The contents of these files are discussed in this section. In figure 7.1 the relation between the DFMS simulation model and the three input files is shown.



PPF: Production Problem File
SDF: System Definition File
OIF: Order Information File

Figure 7.1: The three input files of the DFMS simulator.

7.4.1 The Production Problem File

In the production problem file the following characteristics of the production problem are specified:

- the Tool Classes that are known to the system;
- the fixtures that are known to the system;
- the operations that can be performed;
- the Tool Classes that are required for those operations;
- the Operation Classes supported in the system;
- the products that are known in the system;
- the operations that have to be performed for each operation.

In figure 7.2 a summary of the records in the production problem file is shown.

TOOLS	ToolClass	ToolSize	ToolLife	AssemblyTime
FIXTURE	FixtureClass	AssemblyTime		
PROGRAMS	ProgramName	ToolClass	ToolTime	
OPCLASSES	OpClass	ProgramName		
PRODUCTS	ProductName	ProgramName	FixtureClass	OperationTime

Figure 7.2: The format of entries in a Production Problem File.

7.4.2 The System Definition File

The System Definition File contains all information that is necessary to describe the system that is to be simulated. In the current version of the simulation system the following characteristics are specified:

- the number and type of machine tools;
- the number of pallet positions for each machine tool;
- the size of the tool store for each machine tool;
- the contents of each machine tool store;
- the size of the central tool store;
- the contents of the central tool store;
- the number of central buffer positions;
- the number and type of available fixtures;
- the number of clamp/unclamp operators;
- the number of pallet positions for clamp/unclamp operations;
- the number of pallets;
- the number of transport vehicles;
- the Operation Class distribution;
- the average pallet transport time;

- the preparation time[4].

In figure 7.3 a summary of the records of the System Definition File is shown.

MACHINE	StationId	BufferSize	ToolStoreSize	
CENTRALTOOLSTORE	Positions	PresetPositions		
TOOLS	StationId	ToolClass	ToolTime	NrTools
CENTRALTOOLS	ToolClass	ToolTime	NrTools	
FIXTURES	FixtureName	Count		
LOAD	StationId	NrOperators	NrBuffers	
CENTRALBUFFERS	NrBuffers			
PALLETS	NrPallets			
VEHICLES	NrVehicles	NrVehicles		
MCAPCLASS	OpClass	StationId		
TRANSPORTTIME	Time			
PREPARATIONTIME	Time			

Figure 7.3: The format of entries in a System Definition File.

7.4.3 The Order Information File

The Order Information File contains information about the orders to be processed in the current scheduling period. The list specifies which products have to be made in a particular quantity. In figure 7.4 a summary is shown.

[4]The amount of time that is available between request of pallet and reserved tools and the expected start of the operation.

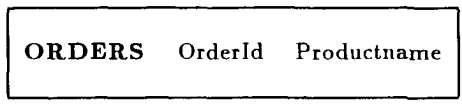


Figure 7.4: The format of entries in a Order Information File.

7.5 Output of the DFMS simulator

The DFMS simulation system can be used to design DFMS installations. The system helps to determine:

- the required number of machines;
- the distribution of Operation Classes over the machines;
- the required sizes of the local tool stores;
- the required size of the system tool store;
- the required tool transport capacity;
- the required number of pallets;
- the required number of fixtures;
- the required pallet transport capacity.

The operator defines a system and runs a simulation to analyze the behavior of the system. To analyze the behavior of the system, the operator has to have access to performance data. In the current system the operator can analyze the following data:

- **Performance data for all machines**
  Examples of data that are presented for each machine:
  - the number of operations performed;
  - the percentage of total time that the machine was busy;
  - the number of tool reservations that were made;
  - the maximum and average length of the Operation Queue;
- **A machine activity Gantt chart**
  The Gantt chart shows for each machine the state the machine was in during a specific period. Four states are distinguished:
  - processing a product;

- being idle because no operations have to be performed;
- being idle because the product is not yet available;
- being idle because tools are not available;

In figure 7.5 an example of sample output is shown.

- **An Operation Class distribution**
  The Operation Class distribution is a bar chart, showing how much time each machine spent processing products of a particular Operation Class. From the information in the bar chart it is possible to decide whether the machines support the correct Operation Classes. In figure 7.6 an example of an Operation Class distribution is shown;

- **An overview of tool reservations**
  The tool reservation overview is a chart that shows the distribution of tool reservation requests. A distinction is made between tools that were already available in the common tool set and tools that had to be assembled. In figure 7.7 an example of a tool reservation distribution is shown;

- **A list of all tools that were reserved**
  This list shows which tools were reserved for each particular operation. The information allows the operator to decide which tools are needed and which tools have to be available in the central tool store. An example of a tool reservation overview is shown in figure 7.8;

- **A list of all tools that could not be delivered**
  All tools that were reserved, but that could not be delivered when were actually needed are shown;

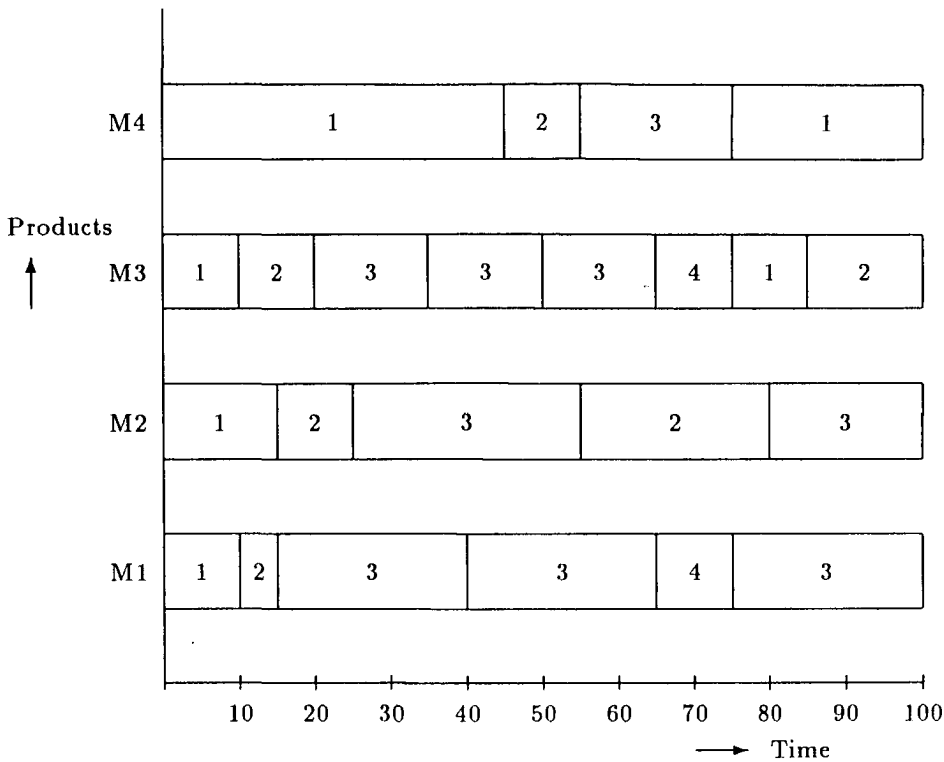
- **An overview of the transport requests**
  The overview of the transport requests is a chart that shows the number of 'active' pallet transport requests. An active transport request is a request that is currently being processed or that waits in a queue until it is processed. In figure 7.9 a sample transport request chart is shown;

- **A distribution of the transport times**
  This distribution is a bar chart that shows which percentage of the transport requests were processed within a specified time. This chart is of critical importance to decide whether enough transport capacity

is installed. In figure 7.10 an example of a transport time distribution is shown;

- **An overview of the use of fixtures**
  The fixture overview shows for all fixtures the maximum number of fixtures that were simultaneously in use. An example of a fixture usage list is shown in figure 7.11;

- **An overview of the activities of the Load Module**
  The overview of the Load Module activities is a chart that shows at which time products were introduced into the system by the Load Module. An example of an activity chart of the Load Module is shown in figure 7.12.



Status 1: Waiting for a operation
Status 2: Waiting for a pallet
Status 3: Processing a operation
Status 4: Waiting for tools

Figure 7.5: A Gantt chart of the machine activities. For every machine (labelled M1 through M4) is shown at which period of time the machine was in one of the four defined states.

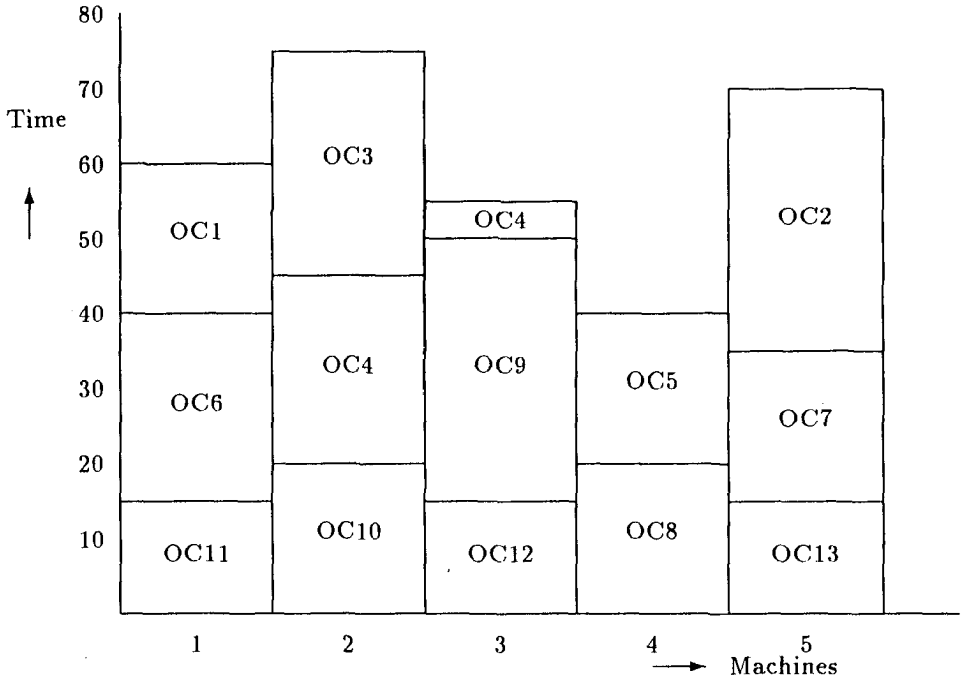


Figure 7.6: An Operation Class distribution bar chart.

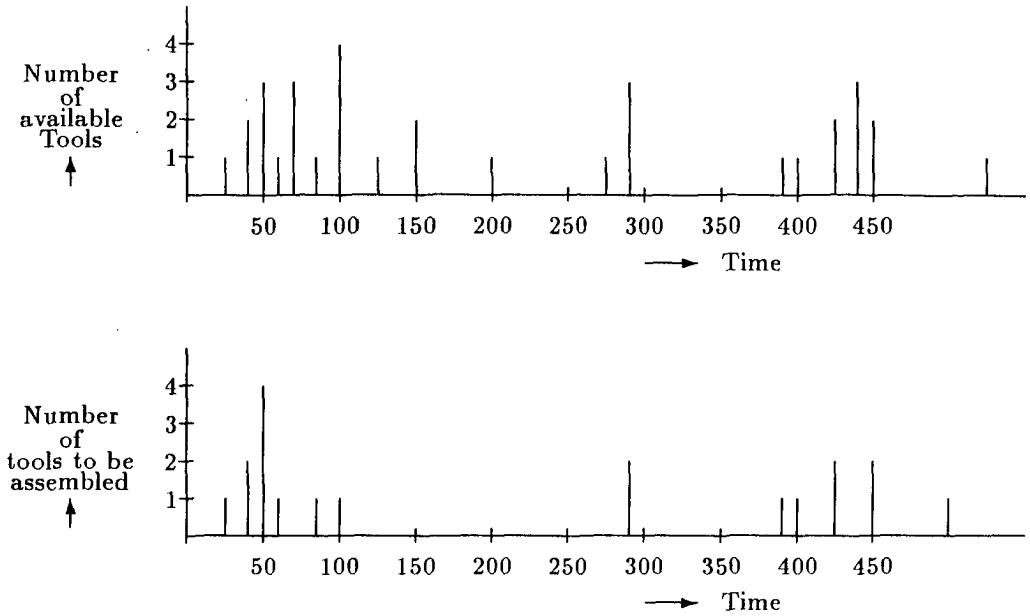


Figure 7.7: An overview of tool reservations. The upper chart shows the tool reservations for which the tools were already available. The lower chart shows the tool reservations for which tools had to be assembled.

Tool Reservation Overview

Operation	ToolClass	NrTools	Total
O9321	T14245200	1	4
	T54241100	1	
	T66352200	1	
	T67737700	1	
O6762	T55245400	1	2
	T63621600	2	
O8727	T43424100	1	1

Figure 7.8: An overview of the tool reservations per operation.

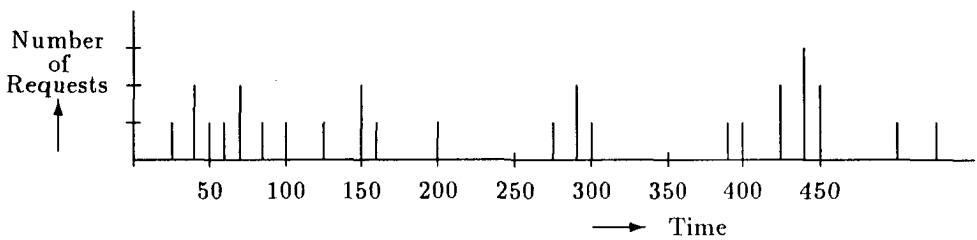


Figure 7.9: A distribution of transport requests.

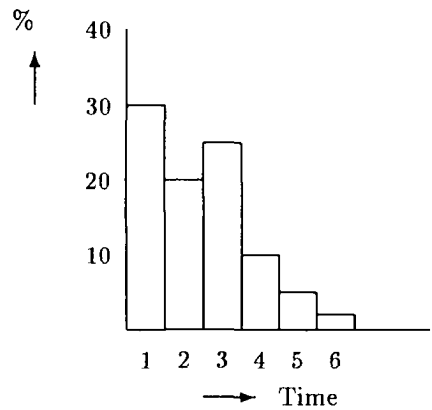


Figure 7.10: A distribution of pallet transport times.

Fixture Usage Overview

Fixture Name	NrAvailable	Maximum Used	Average Used
F100	5	2	0.34
F101	3	3	0.9
F102	10	1	0.003
F104	2	0	0

Figure 7.11: A summary of the fixture usage.

7.6 The implementation of the DFMS simulation system

7.6.1 Introduction

The DFMS simulation system as described in the previous paragraphs, is implemented on an IBM compatible personal computer. A specialized simulation package that supports process oriented simulation models, MUST, has been applied to facilitate the implementation [Must 88]. In appendix B

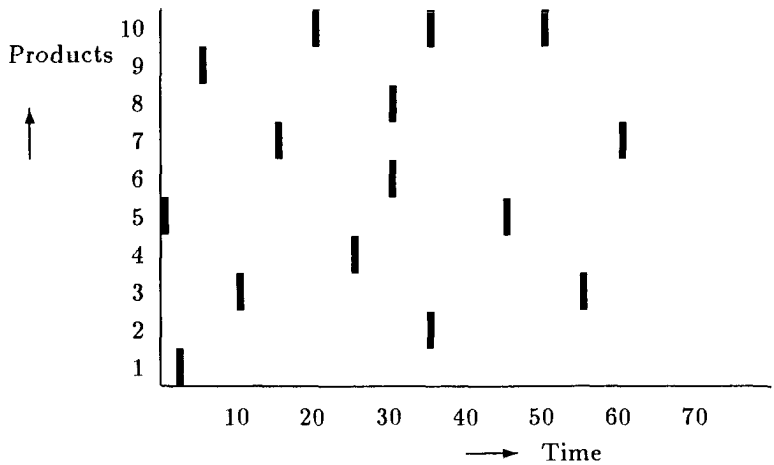


Figure 7.12: The activities of the Load Module. The graph shows at which time products are introduced into the system.

a short introduction on FMS simulation methods is presented.

The MUST software package gives the user the possibility to create *processes* in a standard *Turbo-Pascal* environment[5]. In addition, MUST presents the user with a set of functions that are frequently used in simulation systems.

7.6.2 The structure of the simulation system

In the DFMS simulation system a number of processes are defined. Although MUST is a process oriented simulation system, the processes defined in the DFMS simulation model do not deal with the behavior of physical components. The DFMS simulation system has currently the character of an *event description* system. When the simulation system was designed, the definitions of processes as described in this section, were understood to be the most convenient. However, it has been shown that a simulation model with a much more process oriented character could have been designed in MUST. The functionality of the 'processes' in the DFMS simulation model

[5]Turbo-Pascal is a product of the software house Borland.

is discussed in this section. In figure 7.13 an overview of the processes in the DFMS simulator are shown.

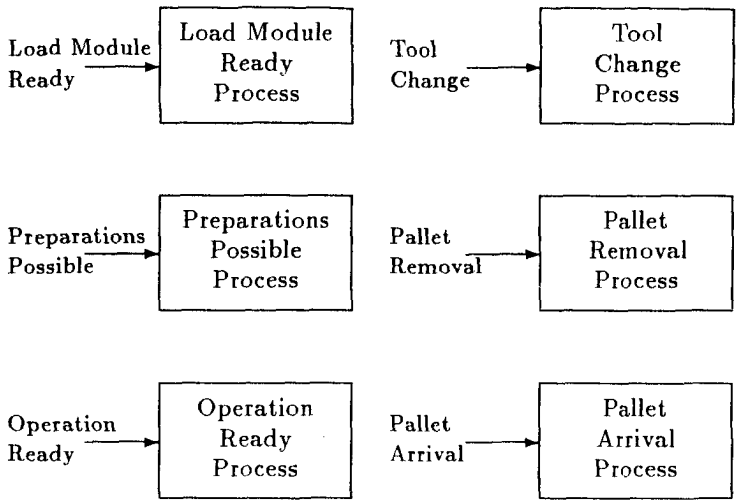


Figure 7.13: The processes of the DFMS simulator.

The Operation Ready process

The Operation Ready process describes the activities that are performed when an operation on a machine is completed. The system determines which machine will perform the next operation on the product. The simulation system knows which Operation Classes are supported by the Station Managers and has access to all Operation Queues. The operation is placed in the Operation Queue of the Station Manager supporting the required operation having, at that moment, the smallest work content. The tool requirements for the new operation are calculated. If the Operation Queue in which the operation was inserted was empty, a Preparations Possible event is scheduled to start the short term preparations (requesting delivery of the reserved tools and the pallet.)

The simulation system checks if a new operation can be started on the machine where the operation just finished. If this is possible, a Preparations

Possible and an Operation Ready event are scheduled for the machine. If no new operation can be started, the current time and the reason why no new operation could be started are listed.

Preparations Possible process

The Preparations Possible process is activated when the short term preparations have to be performed for a machine. The short term preparations consist of requesting the delivery of reserved tools and of the required pallet. The preparations have to be started shortly before the current operation is expected to finish.

If the pallet is not already available at the machine, the simulation system checks whether a pallet buffer position is free to receive the new pallet. If no pallet position is available, the simulation system determines which pallet has to be removed. Both for the pallet that is to be removed and the pallet that is to be delivered a Pallet Arrival and a Pallet Removal event are scheduled. If tools have been reserved for the operation a Tool Change event is scheduled.

Load Module Ready process

The Load Module Ready process is activated when an operation at a clamp station finishes. Two types of operations can be performed at the clamp stations: clamp - and unclamp operations.

If a clamp operation was finished the simulation system determines on which machine the next operation on the product will be performed. The operation is placed in the Operation Queue of the Station Manager supporting the required operation and having at that moment the least work.

If an unclamp operation was finished, the simulation system checks whether the product needs additional operations. If no more operations are needed, the product lead time is calculated and the product is removed from the system. If an additional operation is required the product will first have to be clamped. The product is added to the list of products that are waiting to be introduced into the system, the pallet is added to the list of available pallets. The simulation system then decides whether a new product can be introduced.

If a new product can be introduced into the system a clamp operation is added to the Operation Queue of the Load Module. As soon as a clamp station is available the operation will be carried out: a new Load Module Ready event is scheduled.

Pallet Arrival process

The Pallet Arrival process is activated when a pallet arrives at a machine, at a clamp station or at the central pallet buffer. If the pallet arrives at the central pallet buffer, the position and status of the pallet are updated[6].

If the pallet arrives at a machine or at a clamp station, the simulation system checks if the arrival of the pallet enables an operation to start. This is true when the machine is idle, when the tools for the first operation are already available and when the pallet that just arrived, contains the required product. If the operation can be started at a machine, a Preparations Possible - and an Operation Ready event are scheduled for the machine. If an operation can be started at a clamp station a Load Operation Ready event is scheduled.

The simulation system checks whether the transport vehicle that performed the pallet transport is now available for a new pallet transport request. If this is so and if a pallet request is pending, a new set of Pallet Arrival - and Pallet Removal events are scheduled, both for the pallet to be removed and for the pallet to deliver.

Pallet Removal process

The Pallet Removal process is activated when a pallet is removed from a pallet buffer at a machine, from a clamp station or from the central pallet buffer. The position and status of the pallet are updated.

Tool Change process

The Tool Change process is activated when tools are to be exchanged between the tool store of a machine and the central tool store. When not all reserved tools are ready, the process waits until they are. When all tools are ready, the simulator decides whether tools have to be removed from the machine and which tools can best be removed. The tool data structures are updated to reflect the changed positions of the tools.

The simulation system decides whether the arrival of the new tools enables an operation to start. This is so when the machine is idle, when the product for the first operation is already available and when the tools that just arrived were required for the operation. If the operation can be started

[6]Pallets have a status that specify whether they are currently involved in a transport or not.

a Preparations Possible - and an Operation Ready event are scheduled for the machine.

7.7 Application of the DFMS simulation system

The DFMS simulation system has been tested in order to determine whether the system is able to successfully represent the essential behavior of a DFMS application. Randomly generated production problems have been used to this purpose. The simulator proved to have been implemented according specifications.

As has been stated in the beginning of this chapter, the simulator has been developed for two purposes:

- to analyze for which set of FMS production problems DFMS is most suited;
- to have a tool to design a DFMS system to handle a particular production problem.

It proved to be very difficult to derive generally applicable rules for determining for which production problems DFMS is most suited. There does not seem to be a clearly defined area of production problems where DFMS is not suited at all, nor does there seem to be an area where the performance of a DFMS system is expected to outrank the other control system solutions. In general, DFMS will be applicable for the large majority of the FMS production problems.

With respect to the second objective, it can be said that, based on the current experience, the available simulator is a valuable tool in the design of a DFMS system for a particular production problem.

Chapter 8

Application of DFMS

The DFMS architecture and implementation were originally designed to be used as a control system for FMS. The DFMS architecture however, appears to be suited to cover a range of applications, varying from very simple to very sophisticated, fully automated systems. From the DFMS specifications three different systems can be derived:

- DFMS-J for application in job shop environments;
- DFMS-D for application as a DNC system;
- DFMS-F for application as an FMS control system.

The application and functionality of these systems will be discussed in this chapter. An estimation of the economical feasibility from the supplier's point of view will be presented.

8.1 The DFMS-J system

DFMS-J is designed to control job shop operations. In this section first the principles of a job shop will be discussed. Subsequently the functionality of DFMS-J will be explained and the economical feasibility will be analyzed.

8.1.1 Job shops

A job shop is a manufacturing system that consists of a number of general purpose machines that are used to manufacture a very wide range of products. The products are generally made in small series and are produced on

order. A job shop usually contains manually controlled machines or workstations and machines that are equipped with a CNC. At any time a large number of jobs can be processed in the system. For all but the smallest systems, it poses problems to keep track of all the jobs and to guarantee that all required operations are performed and that no raw material or products are lost.

8.1.2 DFMS-J functionality

DFMS-J can be used to control a standard job shop. A pure DFMS-J system can handle only manually controlled machines or workstations; to integrate CNC controlled machines DFMS-D components are required.

The DFMS-J system consists of a network of small computer systems. Each machine in the system is equipped with a computer; the computer is not connected to the machine, but implements a reduced version of the Station Manager functionality. The Function Modules that have been defined in the DFMS architecture are not implemented on separate computer systems but are implemented on one system: the *Control Module*. An overview of the DFMS-J system is shown in figure 8.1.

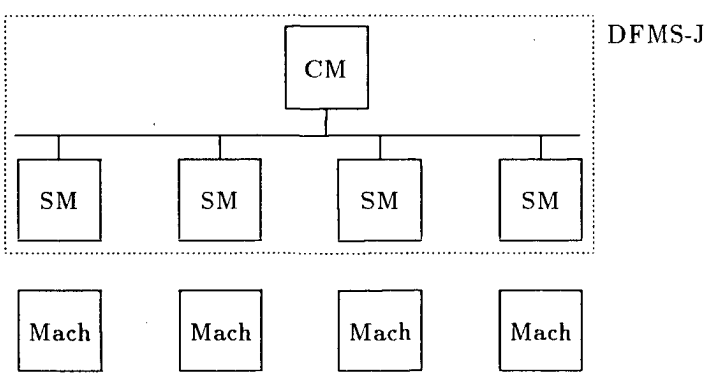


Figure 8.1: An overview of the DFMS-J system. The Control Module (CM) implements reduced versions of the Pallet Transport Module, the Tool Module, the Load Module, the Program Module and the Operator Module. The Station Managers (SM) do not have a direct connection with the machines in the system.

Each Station Manager supports one or more Operation Classes, deter-

mining the operations that can be performed. The Station Managers maintain Operation Queues, accept operations on request of other Station Managers, perform accepted operations and reallocate products to other Station Managers when the products have been processed.

When a new operation is allocated to a Station Manager, the Program Module is requested to download the operation instructions. The operation instructions consist of written instructions to the operator and a tool requirements list. The instructions describe how the product needs to be clamped on the machine and which operation needs to be performed. A workpiece drawing can be included or can be transported together with the product.

The Station Manager reserves tools that are required for the operation. To limit the complexity of the system, tools are usually only reserved for one operation. When the tools arrive at the machine, the operator informs the Station Manager. When the tools have been used, they are transferred back to the central tool room. There an operator decides whether the tools will be used again or will be removed.

The Station Manager requests the Control Module to deliver products when they are needed[1]. The Control Module receives all pallet transport requests and has knowledge of the positions of all products in the system. The Control Module issues orders to human operators to transport products to the Station Managers that requested their delivery. When products arrive, the machine operator informs the Station Manager.

The Station Manager gathers data concerning the performance of the workstation. These data are processed to extract relevant information. The Control Module gathers the processed performance data from the Station Managers and presents *management information reports*. The reports help management to optimize system operations.

8.1.3 Economic considerations

In this section an estimate is made of the prices for which a supplier could offer a DFMS-J installation for 10 connections. Five calculations are made, each based on an estimate of the number of systems to be sold: *10*, *50*, *100*, 500 and 1000 systems.

[1] In a DFMS-J system products will generally not be transported on the expensive metal pallets that are used in fully automated FMS. More typically, wooden pallets will be used.

Fixed Costs

The fixed costs consist of the initial investment for developing the DFMS-J system. The initial investment is considered to consist of:

- software development costs;
- documentation costs;
- marketing costs.

The software required for a DFMS-J system consists of software for the Station Managers and software for the Control Module. The Station Manager software can be derived directly from the prototype Station Manager. Small functional changes are required, but hardly any significant new specifications will have to be developed. The specifications of the Control Module can be derived from the specifications of the Pallet Transport Module, the Tool Module, the Load Module, the Program Module and the Operator Module.

A rough estimate of the time required to transform the prototype software to a standard DFMS-J implementation is approximately two man year[2]. Because highly qualified personnel is required, the costs per person per year are estimated to be 150,000 guilders. The total software costs therefore become 300,000 guilders.

Additional costs have to be made to market the DFMS-J product and to provide customers with adequate documentation. A rough estimate of the required time for both activities is one man year. For these functions less qualified personnel are required: the costs per person per year are estimated to be 100,000 guilders. This leads to additional costs of 200,000 guilders.

The total fixed costs for the DFMS-J system are, based on the estimations of costs of software development, documentation and marketing, approximately 500,000 guilders.

Variable Costs

The variable costs of each DFMS-J installation consists of the following components:

- hardware costs for the Station Manager PCs;
- hardware costs for the Control Module PC;
- costs for the network (hardware and software);

[2] In this estimate the time required for functional changes (one man year) and the time required for bringing the software to industrial quality level (one man year) are included.

- installation costs.

It is assumed that the hardware equipment can be obtained by the supplier with an OEM discount of 30%. This discount has already been subtracted from the hardware prices listed.

The required hardware per Station Manager is one small PC[3]. The costs of these PCs depend heavily on whether an industrial version is considered necessary[4]. A standard IBM-AT compatible PC can be obtained for approximately 3,000 guilders. An industrial version of an IBM-AT compatible systems will be much more expensive: the estimated costs are 10,000 guilders[5].

Because the DecNet-DOS communication system has proven to be able to handle DFMS communication (chapter 5), the data network that is used in DFMS systems is based on the hardware that is supported by DecNet-DOS: Ethernet[6]. The required interface and software can be obtained for approximately 1,600 guilders per connection[7]. The hardware costs per connection therefore range from between approximately 4,600 to 11,600 guilders.

For the Control Module a PC with a large hard disk is chosen. It is not necessary to obtain an industrial version PC because the Control Module can be placed in a save environment, away from the machines. The estimated costs of the Control Module PC are 4000 guilders[8].

The installation costs include the costs of the network cable, the adaptations required to connect the PCs and the manual labor. They are estimated to be 1,000 guilders per Station Manager.

The real hardware costs for the supplier are higher than just the purchase prices of the components. An additional margin for handling and storage of 25% is added to obtain the real costs. The prices for the standard-and industrial PCs therefore become 14,500 and 5,750 guilders. The price of a

[3] In this chapter the name PC will be used to refer to IBM PC compatible computer systems with MS-DOS operating system.

[4] It is not at all certain that industrial quality computer systems are required. It will probably be enough to protect the PC from environmental disturbances by positioning it in a cabinet.

[5] This estimation is based on the prices of Kontron IR 286 industrial computer equipment (October 1988).

[6] Ethernet is a collaborative development of Digital Equipment, Intel and Xerox Corporations.

[7] This cost estimation is based on the Digital Equipment Corporation DEPCA ethernet board and DecNet-DOS license (October 1988).

[8] This costs estimation concerns an IBM-AT compatible with a hard disk of 50 MB (October 1988).

Control Module PC becomes 5,000 guilders.

Calculated sales prices of DFMS-J

A supplier will only consider marketing DFMS-J, if a reasonable profit appears to be possible. It is not necessary for the supplier to make a profit on the purchased components. However it is important that a profit is made on the investment by the supplier, which has been estimated to be 500,000 guilders. It is assumed that a profit of 100% is acceptable, so the fixed costs component in the prices of DFMS-J installations has to be based on 1,000,000 guilders.

In table 8.1 an overview of the estimated sales prices of a DFMS-J system consisting of 10 machines, using standard PCs is shown. In table 8.2 the same data are shown for a system where industrial PCs are used. In the estimated sales prices of the system part of the development costs, the costs for 10 connections and the installation costs are included.

Number of systems to be sold	10	50	100	500	1000
System development overhead	100,000	20,000	10,000	2,000	1,000
Costs of Control Module PC	5,000	5,000	5,000	5,000	5,000
Costs of 10 standard PCs	57,500	57,500	57,500	57,500	57,500
Installation	10,000	10,000	10,000	10,000	10,000
Estimated sales price	172,500	92,500	82,500	74,500	73,500

Table 8.1: Estimated sales prices of a DFMS-J system of 10 machines using standard PCs. The system development overhead is amortized over the varying number of systems sold. Prices are in Dutch guilders, October 1988.

8.1.4 Evaluation of DFMS-J

One of the reasons to use DFMS-J is the high system efficiency that can be obtained. The Station Managers are responsible for having the required resources available when they are needed, operation instructions are downloaded and tools are reserved when necessary. DFMS-J balances the work load over the machines in the system to use all available capacity.

The obtainable higher efficiency can lead to either cost reductions or to increases in productivity. A simple calculation shows which savings are possible. The average costs per machine - and operator hour are estimated

Number of systems to be sold	10	50	100	500	1000
System development overhead	100,000	20,000	10,000	2,000	1,000
Costs of Control Module PC	5,000	5,000	5,000	5,000	5,000
Costs of 10 standard PCs	145,000	145,000	145,000	145,000	145,000
Installation	10,000	10,000	10,000	10,000	10,000
Estimated sales price	260,000	180,000	170,000	162,000	161,000

Table 8.2: Estimated sales prices of a DFMS-J system of 10 machines using industrial PCs. The system development overhead is amortized over the varying number of systems sold. Prices are in Dutch guilders, October 1988.

to be 150 guilders. If the introduction of DFMS-J causes each machine to be productive a quarter of an hour per shift longer, considerable savings are achieved. In a system of 10 machines, working 200 days a year, the total savings per year per shift can be calculated to be 75,000 guilders[9].

Even more important than an efficiency improvement is the higher organization level of a DFMS-J job shop. The use of DFMS-J makes the time when products become available more predictable. This is of crucial importance because the products that are manufactured in the job shop are frequently *parts* required for other, more complex products. If the products processed in the job-shop are late, the scheduling of all later manufacturing steps is disturbed which can lead to high costs.

An additional, very important, advantage of the DFMS-J system is the possibility to gather management information data. These data allow the management to better control operations on the workfloor.

A conservative estimate of the savings achievable by these additional advantages is 25,000 guilders, which brings the total, estimated savings achievable by applying DFMS-J, on 100,000 guilders per year[10]. If the total investment has to be earned back in three years the highest acceptable purchase price from the viewpoint of the consumer is 300,000 guilders.

Even when industrial PCs are used and when only 10 systems can be sold by the DFMS-J supplier, the supplier can realize a profit of 100% over his investment. When larger number of systems can be sold or when standard PCs are used, considerably higher profits are possible. Based on the

[9]The savings discussed here are *virtual* savings. Only if the extra time is used to increase the yearly production one can speak of *real* savings.

[10]Assuming that the system is used only one shift per day; more shifts per day lead to higher savings per year.

calculation presented in this paragraph DFMS-J can be considered to be a feasible system.

8.2 The DFMS-D system

DFMS-D can be used for the traditional *Direct Numerical Control* (DNC) application. In this section first the principles of DNC will be explained. Subsequently the functionality of DFMS-D will be discussed and the economical feasibility will be analyzed.

8.2.1 DNC systems

In many NC applications the machines are used in a stand-alone environment. Each machine is a separate organizational entity. The part programs are usually developed by programmers from a programming department and *punched tape* is used to transfer part programs to the CNCs. The tool data of the tools used on the machine are entered manually by the operator. This simple mode of operation is not very efficient. Some of the problems of traditional NC operation are listed here:

- the punched tape is mechanically vulnerable;
- a complicated organization is needed to maintain the tape archive;
- changing the part programs is not possible without making a new tape[11];
- punching the tapes takes considerable time;
- entering the tool data manually is error prone.

Direct Numerical Control (DNC) is applied to overcome these problems [Gim 85], [Gim 86], [Gim 87]. The principle of DNC is that each machine is connected to a central system. This central system functions as a database where the part programs that are used in the system are stored. The operator at each machine is provided with a simple *DNC terminal* that is used to request the downloading of programs. In many DNC systems a *tool preset station* can be connected. The tool data measured at the preset station are stored in the central system and are transferred to the machines on request of the operator of the machine where the tools are used. The DNC

[11] On some CNCs the program can be edited. To prevent having to make the changes time and time again, a new tape will have to be punched.

system completely eliminates the need to use punched tapes, thereby preventing tape handling and most of the problems listed above. In figure 8.2 an overview of a DNC system is shown.

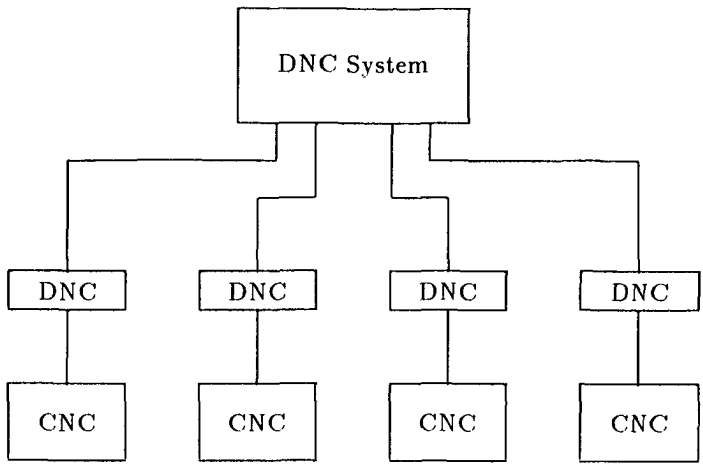


Figure 8.2: An overview of a DNC system. The CNCs are connected by means of simple DNC terminals to a central computer system.

The application of DNC brings the additional advantage of the possibility to gather machine performance data. Usually only information like: 'the machine is running' or 'the machine is not running' can be monitored.

8.2.2 DFMS-D functionality

DFMS-D can be used to control a system with roughly the DNC functionality. In a DFMS-D system the machines are CNC controlled; to each CNC a Station Manager is connected, either by means of a programmable CNC or a Station Adapter. An overview of DFMS-D is shown in figure 8.3. The mere fact that CNC controlled machines are used instead of manually controlled machines has many advantages; these advantages will not be discussed here (for more information on NC organization refer to [Vol 85] and [Reij 88].) In this section the functionality of DFMS-D will be discussed.

Each Station Manager supports one or more Operation Classes that determine which operations can be performed. The Station Managers maintain Operation Queues, accept operations on request of other Station Managers,

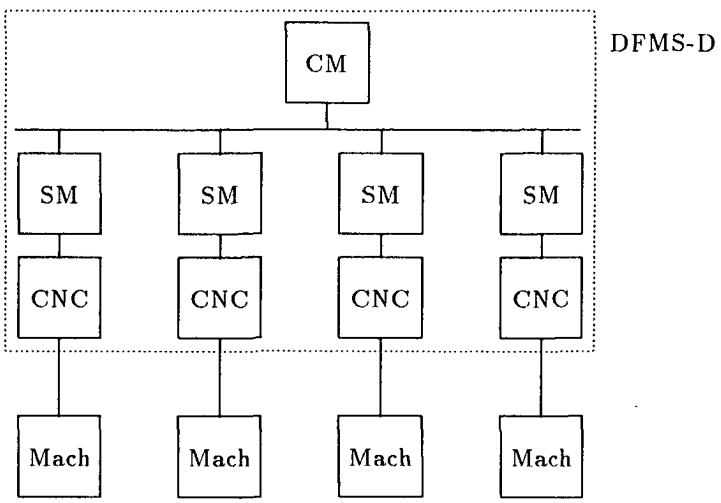


Figure 8.3: An overview of DFMS-D. The Control Module (CM) implements reduced versions of the Pallet Transport Module, the Tool Module, the Load Module, the Program Module and the Operator Module. The Station Managers have a direct connection with the CNCs that control the machines.

perform accepted operations and reallocate products to other Station Managers when the products have received their operations.

When a new operation has been allocated to a Station Manager, the Control Module is requested to download programs. A program consists of the NC part program, the tool requirements list and written instructions. The instructions describe how the product needs to be clamped on the machine. A workpiece drawing can be included or can be transported together with the product. The downloading of the NC part program from Station Manager to the CNC is performed by the Station Manager.

NC part programs can be adapted on the CNC when optimization is necessary. The adapted part program can be uploaded to the Control Module. A programmer has to authorize the changes before the changed program can be down loaded again.

The Station Manager reserves tools that are required for the operation. The Control Module receives all requests, the tool room operator is responsible for performing them. Shortly before they are needed the Station Manager requests delivery of the tools. The Station Manager determines which tools can be removed to make place for the new tools. The tool data of the new tools are downloaded by the Control Module[12]. The Station Manager is informed by the CNC when tools are delivered or are removed.

The exchanged tools are transferred back to the central tool room. The tool data of these tools are uploaded to the Control Module. An operator decides whether the tools will be used again or will be removed.

The Station Manager requests the Control Module to deliver products when they are needed[13]. The Control Module receives all requests and has knowledge about the positions of all products in the system. The Control Module issues orders to human operators to transport products to the Station Managers that requested their delivery. When products arrive, the machine operator informs the Station Manager.

The close connection between Station Manager and CNC enables an extensive Management Information System functionality. All events concerning CNC or Station Manager considered relevant, can be logged by the Station Managers. The logs can be uploaded to the Control Module where MIS reports can be generated.

[12]In simple DNC-like systems the extra investment required to transfer tool data with the tool itself may be too high.

[13]In a DFMS-D system products will generally not be transported on the expensive metal pallets that are used in fully automated FMS. More typically, wooden pallets will be used.

8.2.3 Economic considerations

In this section an estimate is made of the prices for which a supplier could offer a DFMS-D installation for 10 connections. Five calculations are made, each based on an estimate of the number of systems to be sold: 10, 50, 100, 500 and 1000 systems[14]. Again, an OEM discount of 30% has already been subtracted from the prices listed.

Fixed costs

The fixed costs consist of the initial investment for developing the DFMS-D system. Identical to DFMS-J, the initial investment is considered to consist of:

- software development costs;
- documentation costs;
- marketing costs.

The software required for DFMS-D system consists of software for the Station Managers and software for the Control Module. The Station Manager software can be derived directly from the prototype Station Manager. Due to the additional functionality the Station Manager is slightly more complicated than the DFMS-J Station Manager.

The DFMS-D Control Module closely resembles the DFMS-J Control Module. There is no significant difference between downloading operation instructions for manually operated machines and downloading NC part programs.

It is estimated that approximately three additional man years are required to develop the DFMS-D software when the DFMS prototype is used as starting point[15]. The costs of the software development are approximately 450,000 guilders.

Additional costs have to be made to market the DFMS-D product and to provide customers with adequate documentation. A rough estimate of the required time for both activities is one man year. This leads to additional costs of 200,000 guilders[16].

[14]The estimation of 1000 systems to be sold is based on data from RWT, one of the largest DNC manufacturers. RWT claims to have installed more than 3000 DNC connections in more than 1000 systems. In the Netherlands in 1988, still only a few companies apply DNC.

[15]When the DFMS-J implementation is already available only a half man year is estimated to be necessary.

[16]The costs of DFMS-D and DFMS-J marketing and development will partially overlap.

The total fixed costs for the DFMS-D system are, based on the cost estimations of software development, documentation and marketing, approximately 650,000 guilders.

Variable Costs

The variable costs of each DFMS-D installation consists of the following components:

- hardware costs for the Station Manager implementations;
- hardware costs for the Control Module PC;
- costs for the network (hardware and software);
- installation costs.

Two different DFMS-D realizations are possible: a realization with Station Adapters and one with programmable CNCs. The required hardware per Station Adapter is comparable to the DFMS-J hardware. Therefore, the costs are approximately 4,600 guilders for standard PCs and 11,600 guilders for the industrial computer systems.

The costs of a programmable CNC depend on the realization chosen. In chapter 4 the prototype Philips programmable CNC realization is discussed. For reasons of simplicity the costs of a programmable CNC realization are considered to be comparable with the costs of the standard PC implementation[17].

Because the Control Module in DFMS-D is much heavier used than in DFMS-J, an IBM-AT compatible based on a Intel 80386 microprocessor will be used. The costs of a medium speed 80386 based PC is estimated to be approximately 10,000 guilders.

The installation costs for DFMS-D are estimated to be equal to the costs for DFMS-J: 1,000 guilders per connection.

The real hardware costs for the supplier are higher than just the purchase prices of the components. An additional margin for handling and storage of 25% is added to obtain the real costs. The prices for the standard and industrial PCs therefore become 14,500 and 5,750 guilders. The price of a Control Module PC becomes 12,500 guilders.

[17] In reality it is to be expected that a programmable CNC can be realized for considerably less than 4,600 guilders.

Calculated selling prices of DFMS-D

The fixed costs have been estimated to be 650,000 guilders. When the 100% profit is included, the fixed costs component in the prices of DFMS-D installations can be calculated to be 1,300,000 guilders.

In table 8.3 an overview of the sales prices of a DFMS-D system consisting of 10 machines, using standard PCs or programmable CNCs is shown. In table 8.4 the same data are shown for a system where industrial PCs are used. In the sales prices part of the development costs, the costs for 10 connections and the installation costs are included.

The sales prices are calculated under the assumption that modern CNCs are available for which a connection with a Station Manager can be easily established[18]. If older CNCs, with limited DNC capabilities, have to be integrated, the required software adaptation will lead to considerable additional costs.

Number of systems to be sold	10	50	100	500	1000
System development overhead	130,000	26,000	13,000	2,600	1,300
Costs of Control Module	12,500	12,500	12,500	12,500	12,500
Costs of 10 standard PCs	57,500	57,500	57,500	57,500	57,500
Installation	10,000	10,000	10,000	10,000	10,000
Estimated sales price	210,000	106,000	93,000	82,600	81,300

Table 8.3: Estimated sales prices of a DFMS-D system of 10 machines using standard PCs. The system development overhead is amortized over the varying number of systems sold. Prices are in Dutch guilders, October 1988.

8.2.4 DFMS-D versus traditional DNC systems

Functional comparison

Currently a large number of DNC systems are operational. The traditional DNC systems closely cooperate with a NC part programming system. For this cooperation however, there is no real need. The DNC system just has to be able to transfer normal ASCII files; there is no need to have knowledge of the internal structure of the files. In the DFMS architecture part programming and DNC are independent functions.

[18] A suitable CNC is for example the programmable CNC that is currently being developed by Philips Numerical Control (October 88).

Number of systems to be sold	10	50	100	500	1000
System development overhead	130,000	26,000	13,000	2,600	1,300
Costs of Control Module PC	12,500	12,500	12,500	12,500	12,500
Costs of 10 standard PCs	145,000	145,000	145,000	145,000	145,000
Installation	10,000	10,000	10,000	10,000	10,000
Estimated sales price	297,500	193,500	180,500	170,100	168,800

Table 8.4: Estimated sales prices of a DFMS-D system of 10 machines using industrial PCs. The system development overhead is amortized over the varying number of total systems sold. Prices are in Dutch guilders, October 1988.

DFMS-D offers, apart from the standard DNC functionality, some additional advantages:

- **Dispatching of operations**
  The system chooses which machines will perform operations on products. Once a product is entered in the system, it will receive all required operations;

- **Automatic request of programs**
  The Station Managers request the Program Module to download program data when an operation needs to be performed for which the data is not already available. Unlike standard DNC systems, no operator is required;

- **Pallet management**
  The Station Managers request the Pallet Transport Module to deliver pallets shortly before they are needed;

- **Extended tool management**
  The Station Managers determine the tool requirements and request the Tool Module to deliver missing tool capacity. Tool life data can be easily maintained;

- **Extensive Management Information System functionality**
  The Station Managers are able to report all events considered relevant;

- **Extensibility**
  Perhaps the most important feature of DFMS-D is the extensibility. It

RWT	140,000
UNC	120,000
Gildemeister	120,000
Siemens	250,000
DLog	80,000

Table 8.5: The commercial prices of a number of DNC systems, by German manufacturers, for 10 machines. Prices are approximate and in Dutch guilders, October 1988.

is possible to gradually extend a DFMS-D system to a fully automated FMS. It is a small step from a DFMS-D system to a DFMS-F system.

Economic comparison

In a recent survey of DNC systems several DNC manufacturers made quotations for a system consisting of 10 machines. The DNC system included Management Information System functionality. In table 8.5 the results of this study are shown.

When comparing the prices of traditional DNC system in table 8.5 with the prices of the DFMS-D system as listed in tables 8.3 and 8.4, the following considerations apply:

- The prices of the DFMS-D system are *estimated.* The quotations of the DNC manufacturers in table 8.5 are real;
- The functionality of the DFMS-D system is much higher than the functionality of the traditional DNC systems, therefore the prices are not completely comparable;
- The most reasonable guess concerning the number of systems that can be sold is 100. The DFMS-D prices in the middle column of the tables therefore apply.

8.2.5 Evaluation of the DFMS-D system

The reasons to apply DFMS-D are roughly similar to the reasons to apply DFMS-J. The savings that can be achieved due to the higher productivity are in this case somewhat higher, because the machine hour costs of numerically controlled machines are estimated to be higher than the machine hour

costs of manually operated machines (200 guilders instead of 150 guilders). If the same calculation is applied as has been performed for the DFMS-J system, the yearly savings are calculated to be 100,000 guilders per shift.

The higher organizational level and efficiency improvements that have already been discussed for the DFMS-J system are for the DFMS-D even more important advantages. A very conservative estimate of the savings by these improvements amount to 50,0000 guilders per year.

The total savings achievable by applying DFMS-D, are estimated to be 150,000 guilders per year. If the total investment has to be earned back in three years the highest acceptable purchase price from the viewpoint of the consumer is 450,000 guilders.

The sales price, when industrial PCs are used and when only 10 systems can be sold by the DFMS-D supplier has been calculated to be approximately 300,000 guilders. Comparing this to the assumed acceptable purchase price of 450,000 guilders, the supplier can expect to realize a profit of much more than 100% over his investment. When larger number of systems can be sold or when standard PCs are used, even higher profits are possible.

The data in table 8.5 suggest that a sales price of approximately 200,000 guilders is maximally achievable[19]. Even this lower acceptable price does not pose any problems. When standard PCs are used, the sales price is 210,000 guilders when only 10 systems are sold. When industrial PCs are used at least 50 systems have to be sold in order to be able to limit the price to 200,000 guilders.

Based on the calculation presented in this paragraph DFMS-D can be considered to be a feasible system.

8.3 The DFMS-F System

DFMS-F is designed to control fully automated Flexible Manufacturing Systems. In this section first some characteristics of FMS will be discussed. Subsequently, the functionality of DFMS-F will be explained and the economical feasibility will be analyzed.

[19]This is considerably lower than the estimate based on cost reductions by efficiency improvements. Suppliers may have to reckon with a careful and conservative consumer market.

8.3.1 Flexible Manufacturing Systems

In a typical FMS the machines are able to perform a diverse range of operations on products. The products are members of a product family for which the FMS is designed. Products are transferred automatically from machine to machine. An elaborate tool management system is used to guarantee that the required tools are available when needed. FMS is a much more complicated system to apply than a DNC system. DNC is basically a simple system to avoid the usage of punched tapes. No major reorganization of the production facility is required when DNC is installed. To apply FMS successfully the organization of the production system has to be adapted.

8.3.2 DFMS-F functionality

DFMS-F implements the full FMS functionality as described in chapter 3. DFMS-F can be considered to be an extension to DFMS-D. In addition to the DFMS-D functionality, now automatic pallet transport and optionally automatic tool transport are supported. Instead of one Control Module that implements the functionality of a number of Function Modules, separate implementations of Function Modules are used.

Due to the fact that DFMS-D has a much more elaborate functionality than a traditional DNC system, the distinction between a DFMS-D system and a DFMS-F system is much smaller than between a traditional DNC system and an FMS. A very easy grow path can be defined from DFMS-D to DFMS-F.

8.3.3 Economic considerations

In this section an estimate is made of the sales prices of a DFMS-F application. The number of DFMS-F applications that can be expected to be sold is considerably smaller than the number of DFMS-D and DFMS-J systems. A low estimate is approximately 10 systems, a high estimate is 250 systems.

Fixed costs

The fixed costs consist of the initial investment for developing the DFMS-F system. Identical to DFMS-J and DFMS-D, the initial investment is considered to consist of:

- software development costs;
- documentation costs;

- marketing costs.

DFMS-F software

The software required for a DFMS-F system consists of software for the Station Managers and software for the Control Module. The Station Manager software can be directly derived from the prototype Station Manager. The DFMS-F version of the Station Manager software is almost completely similar to the DFMS-D version. If the DFMS-D version is already available no additional costs will have to be made for the Station Managers; if no DFMS-D is available it is estimated that approximately three man years of development are required.

Due to the additional functionality of the Function Modules Station additional development is required. A conservative estimate of the required time is two man years[20]. The cumulative costs of the software development for DFMS-F amounts to 750,000 guilders

Additional costs have to be made to market the DFMS-F product and to provide customers with adequate documentation. A rough estimate of the required time for both activities is one and a half man year. This leads to a total of additional costs of 300,000 guilders[21].

The total fixed costs for the DFMS-F system are, based on the costs estimations of software development, documentation and marketing, approximately 1,050,000 guilders.

Variable Costs

The variable costs of each DFMS-F installation consists of the following components:

- hardware costs for the Station Manager implementations;
- hardware costs for the Function Module PCs;
- costs for the network (hardware and software);
- installation costs.

The required hardware is very similar to the hardware required for a DFMS-D system. The only difference is that, due to the more extensive functionality

[20] If the DFMS-D versions of the Function Modules are available, the required time will be less.

[21] The costs of DFMS-F, DFMS-D and DFMS-J marketing, documentation and development will largely overlap.

of the Function Modules, one PC will not suffice anymore. A better option is to implement each Function Module on a separate PC. Separate PCs are required for:

- the Tool Module;
- the Pallet Transport Module;
- the Operator Module;
- the Program Module;
- the Load Module.

To implement these Function Modules, moderately fast IBM-AT compatibles can be used. The total costs of 5 systems is estimated to be 20,000 guilders.

The installation costs for DFMS-F are estimated to be equal to the costs for DFMS-J and DFMS-D: 1,000 guilders per connection.

The real hardware costs for the supplier are higher than just the purchase prices of the components. An additional margin for handling and storage of 25% is added to obtain the real costs. The prices for the Station Manager implementation therefore becomes 5,750 guilders. The price of the five Function Modules becomes 25,000 guilders.

Calculated selling prices of DFMS-F

The fixed costs have been estimated to be 1,050,000 guilders. When the 100% profit is included, the fixed costs component in the prices of DFMS-F installations can be calculated to be 2,100,000 guilders.

In table 8.6 an overview of the estimated sales prices of a DFMS-F system consisting of 10 machines, using standard PCs or programmable CNCs is shown.

8.3.4 DFMS-F versus traditional FMS control systems

Functional comparison

The specifications of the DFMS-F systems are similar to the general DFMS specifications. These specifications have already been discussed in detail in chapter 3.

Number of systems to be sold	10	20	50	100	250
System development overhead	210,000	105,000	42,000	21,000	8,400
Costs of Function Modules	25,000	25,000	25,000	25,000	25,000
Costs of 10 prog. CNC's	57,500	57,500	57,500	57,500	57,500
Installation	10,000	10,000	10,000	10,000	10,000
Estimated sales price	302,500	197,500	134,500	113,500	100,900

Table 8.6: Estimated sales prices of a DFMS-F system of 10 machines using programmable CNCs. The system development overhead is amortized over the varying number of systems sold. Prices are in Dutch guilders, October 1988.

Economic comparison

The economic justification of FMS is a complex issue. The cost of the FMS control system is only a relatively minor fraction of the total system costs. A separate justification of FMS hardware - and FMS control system investments does not make sense. The economic evaluation of the DFMS-F system will therefore solely consist of a comparison with alternative FMS control systems.

An elaborate economic comparison between DFMS-F and other FMS control systems is not possible, because not much data is available. Most FMS system integrators are machine-tool manufacturers whose primary interest is to sell machines. The FMS control system is for them a requirement to sell the FMS, not a separate product. Information about the costs of FMS control systems is therefore difficult to obtain.

Siemens is one of the few manufacturers of FMS control systems that sell the control system as a separate product. The costs of the FMS-M control system of Siemens, as discussed in chapter 2 are approximately 500,000 guilders. The functionality of the FMS-M system is roughly similar to the functionality of the DFMS-F system.

In comparison, the estimated sales price of the DFMS-F system is just 200,000 guilders (when 20 systems are sold.) Based on these data DFMS-F can be considered to be a feasible alternative for traditional FMS control systems.

8.4 The DFMS control system family

A well known problem of many highly sophisticated systems is that it is not possible to introduce these systems in small steps. Introducing these systems in *one* step leads to many financial and organizational problems.

The distributed DFMS architecture and implementation have the advantage of the possibility of a highly modular introduction. A DFMS system can be introduced in many steps. Each step may be followed by a next one, but this is not required to obtain the general DFMS advantages.

In this section two cases of modular introduction of a DFMS system are discussed. The first case describes the high end (FMS) situation, the second case the low end (job shop) situation.

8.4.1 The growth path to a fully automated FMS

In the first example a new system will be installed that will eventually have full FMS functionality. The system is installed in small steps. Each extension to the system is only made after the already installed parts of the system work satisfactorily. The example system will eventually consist of 4 machines. In the final implementation an automated product transport system and an automated tool handling system will be installed.

Stage 1: Stand alone machines

The machines that are to become part of the system are equipped with programmable CNCs. The machines can be bought and installed separately. Initially two machines are installed. The other machines will only be bought when the market develops satisfactorily. The two machines function completely independent from each other. In figure 8.4 the initial situation is shown.

Stage 2: DNC functionality

The next stage is the introduction of the DFMS-D system. A simple Local Area Network (for example DecNet DOS) will be installed and the two machines will be connected. The standard DFMS-D Station Manager functionality is implemented on the programmable CNCs.

The functionality of the Function Modules is combined into one logical component: the DFMS Control Module. The Control Module contains

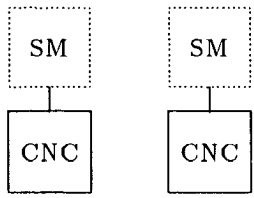


Figure 8.4: A System consisting of two separate machines. The Station Manager functionality is already available, but is not yet required.

rudimentary versions of the implementations of the Pallet Transport Module, Operator Module, Load Module, Program Module and the Tool Module. The Control Module is implemented on one system. In figure 8.5 the simple DFMS system is shown.

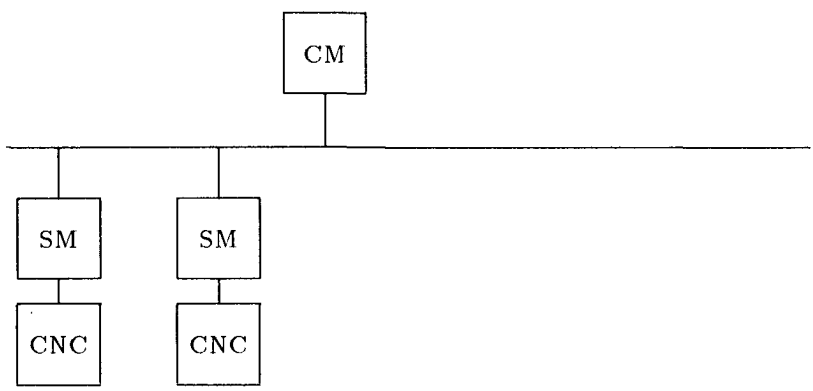


Figure 8.5: A simple DFMS-D system consisting of two machines.

Stage 3: Extension of the system

New machines can be added to the system when the market situation develops as expected. The new machines can be integrated easily into the system. If the market developments are disappointing, no unnecessary investments

have been made. The advantages of DFMS-D are noticeable even in small systems. In figure 8.6 the extended DFMS-D system is shown.

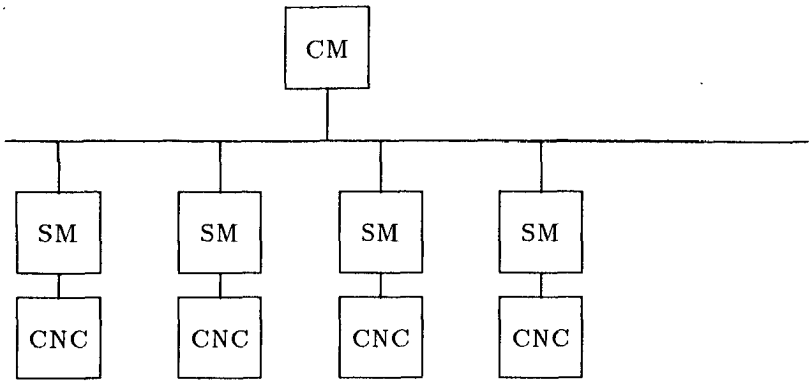


Figure 8.6: A DFMS-D system consisting of 4 machines.

Stage 4: Automated pallet transport

When the system reaches a certain size and when manual product transport leads to inefficient operation of the system, an automatic pallet transport system can be introduced[22]. The system will move gradually from a DFMS-D implementation to a DFMS-F implementation. The Station Managers are adapted slightly to handle the automatic pallet transports.

The pallet transport system is controlled by a separate pallet transport system controller. A standard implementation of the Pallet Transport Module is now required; a new control component is added to the network. The pallet transport messages that were processed by the Control Module are now processed by the Pallet Transport Module. In figure 8.7 a DFMS-F system with a separate Pallet Transport Module is shown.

Stage 5: Automated tool transport

An automated tool transport system is usually only installed in larger systems. A separate Tool Module is added to the DFMS network. The tool

[22] It is of course only possible to install a pallet transport system if the machines already installed offer the possibility of automatic pallet transport.

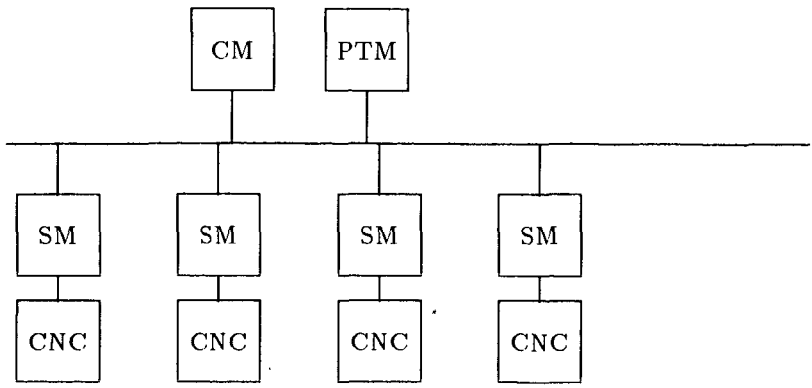


Figure 8.7: A DFMS-F system with a separate Pallet Transport Module.

messages that were previously processed by the Control Module will now be processed by the Tool Module. The Tool Module will be responsible for all tool management in the system. In figure 8.8 the extended DFMS-F system is shown.

8.4.2 From job-shop to DNC functionality

The second example shows how a traditional job-shop can be organized and can be extended to a DFMS-D system.

The first step to organize the job shop is the install a DFMS-J control system. With the DFMS-J system only manually controlled machines can be integrated into the DFMS system. To gain experience with the DFMS-J system, initially a small number of machines will be equipped with a Station Manager. The DFMS-J control module and the DFMS network will be installed. When the system works satisfactorily gradually more machines can be equipped with a Station Manager.

To include CNC controlled machines, the system has to be upgraded to the DFMS-D functionality. The manually controlled machines remain operating as they did in the DFMS-J environment. The DFMS-D Control Module handles the DFMS-J Station Managers without problems.

The CNC controlled machines can now operate in full DNC mode. When older, manually controlled machines are replaced by CNC controlled ma-

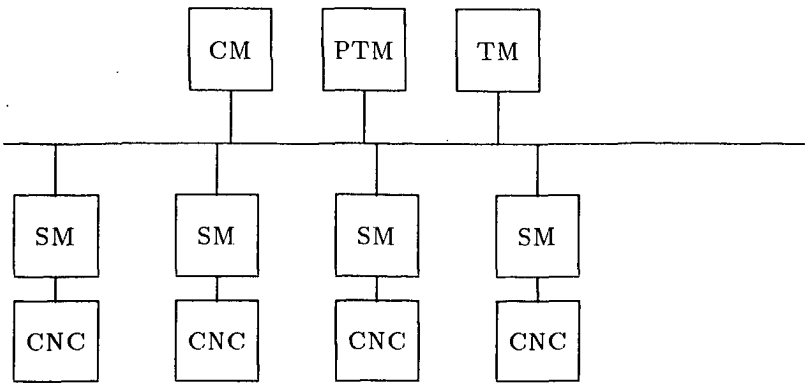


Figure 8.8: A DFMS-F system with a separate Tool Module and Pallet Transport Module.

chines, a growing part of the system will profit from the DFMS-D functionality.

Chapter 9

The quintessence of DFMS

In this chapter the essential characteristics of DFMS will be explained. The advantages and disadvantages of DFMS will be made clear.

9.1 Characteristics of DFMS

The DFMS system is characterized by the following features:

- **Separation of functionality**
  In the DFMS system a clear and easy to understand separation of functionality has been achieved. Communication between DFMS functions is performed by exchanging messages;

- **Application of client-server model**
  In the DFMS architecture some components function as *servers* that perform general services and other components function as *clients* that request those services. The Function Modules are the servers and the Station Managers are the clients in the DFMS architecture;

- **Absence of central control**
  No component in the DFMS architecture can *order* other components to perform actions. The DFMS components confer with each other to determine what should be done;

- **Distributed implementation**
  The distributed DFMS architecture can (and should preferably) be implemented distributed. A Local Area Network is required for communication between the DFMS components;

- **Absence of scheduling**
  Unlike traditional FMS control systems, the DFMS control system does not use *production schedules* that are prepared beforehand. The sequence of operations on machines is determined during the actual production.

9.2 Advantages of DFMS

The advantages of DFMS as discussed here, will only be fully obtained if a distributed implementation of the architecture is used and if use is made of programmable CNCs. Advantages are:

- **Simplicity**
  The DFMS concept is inherently simple, it can be easily explained to the people who have to work with the system. It would be possible to have a completely manual implementation of the DFMS architecture, by having each machine operator perform the Station Manager functionality. Each machine operator would perfectly well understand what actions would have to be performed and how these actions would contribute to the overall system goal;

- **Robustness against internal problems**
  Because the DFMS components function strictly separated from each other and communicate by exchanging messages, an intrinsically robust system has been created. An error situation in one component will not easily affect other components. The architectural specification allows each DFMS component to be stopped and started independently from other components without disrupting system operations;

- **Robustness against operational problems**
  In a traditional FMS where a more or less detailed schedule is executed, each disturbance of the schedule has consequences for the operations that still have to be performed. If, for example, an operation takes longer than the nominal time, the machine and product will become available later and the tools are allocated longer then was expected. A complicated dispatcher is required to minimize the deviations from the schedule. When many disturbances occur, the schedule will have to be adapted frequently.

  In a DFMS system no adaptions have to be made when problems occur during operations. This feature makes the system much more robust;

- **Broad application**
  The DFMS architecture can be used for a broad range of applications. The three variants on the DFMS architecture: DFMS-J, DFMS-D and DFMS-F cover respectively the job shop, the DNC and the FMS application. The three DFMS versions are architecturally almost identical; upgrading from one version to another is simple;

- **Suitability for low end application**
  The DFMS-D system enables the gradual introduction of FMS type technology. FMS technology can now be applied in areas where previously FMS was considered much too complicated or expensive;

- **Configurability**
  It is simple to add or remove machines in a running DFMS application. Adding a machine is performed by simply connecting the Station Manager to the DFMS Network and by letting it partake in negotiations for operations. Removing a machine is performed by 'breaking' the link between Station Manager and DFMS Network.

- **Extensibility**
  It is very simple to extend the control system, when new machines are added to the production system. Simply by connecting and activating another Station Manager to the DFMS Network the control system is extended. This is a huge commercial advantage. It is not necessary to invest in a large control system, early in a project under the assumption that the capacity may be needed sometime later. In DFMS it is possible to control capacity on a 'as needed' basis;

- **Low cost**
  It is to be expected that in the near future many CNCs will be equipped with MS-DOS compatible processing capabilities. When this feature becomes standard, it is possible to construct an FMS control system at very low additional cost.

9.3 Disadvantages of DFMS

The DFMS architecture uses simple algorithms to determine the sequence of operations on the machines. The disadvantage of using these simple algorithms is the danger that in some applications the performance of the system will not be satisfactory. For these situations the DMFS architecture can be

extended with a special *Dispatching Module.* The Dispatching Module can apply the most sophisticated dispatching rules that are currently available.

Bibliography

[Acti 83] ACTIS-DATO M., CIAFFI P., CIGNA P., 1983, A generalized job scheduling system for FMS. *Proc. 2nd. Int. Conf. Flexible Manufacturing Systems, 615-624, IFS, Bedford UK.*

[Bak 88a] BAKKER J.J.A., 1988, DFMS: A new control structure for FMS. *Computers in Industry 10, 1-9.*

[Bak 88b] BAKKER J.J.A., 1988, Een gedistribueerde FFS besturing. *MB Produktietechniek, Jrg. 54, nr. 8, september (Dutch).*

[Bat 85] BATELLE COLUMBUS LABORATORIES, 1985, Flexible Manufacturing Systems in Metal Removal, *Electric Power Research Institute, EM-4184, Research project 2613-3.*

[Ben 82] BEN-ARI M., 1982, Principles of Concurrent Programming. *Prentice Hall, Englewood Cliffs.*

[Ber 85] BER ABRAHAM, FALKENBURG DONALD R., 1985, Tool Management for FMS. *Annals of the CIRP Vol. 34/1/1985, 387-390.*

[Bie 88a] BIEMANS F.P. AND VISSERS C.A., 1988, Computational tasks in Robotics and factory Automation. *Computers in Industry, Vol 10., No 2., pp 95-112*

[Bie 88b] BIEMANS F.P. AND VISSERS C.A., 1988, A Reference Model for Manufacturing Planning and Control Systems. *Accepted for publication in the Journal of Manufacturing Systems, also published as Philips Laboratories Briarcliff Note TN-88-017.*

[Bla 82] BLACKSTONE JOHN H., PHILLIPS DON T. AND HOGG GARY. L, 1982, A state-of-the-art survey of dispatching rules for manufacturing job shop operations. *Int. J. Prod. Res., Vol. 20, no 1, 27-45.*

[Brow 68] BROSHEER BEN C. AND DESOLLAR JAMES C., 1968, Variable Mission' Machining. *American Machinist, special report no. 620, september.*

[Brow 82] BROWNE J. AND RATHMILL K., 1982, The use of simulation modelling as a design tool for FMS. *Proc. 2nd. Int. Conf. Flexible Manufacturing Systems, 197-214, IFS, Bedford UK.*

[Car 82] CARRIE A.S., ADHAMI, E., 1982, Introducing FMS by simulation. *Proc. 2nd. Int. Conf. Flexible Manufacturing Systems, 229-238, IFS, Bedford UK.*

[Chan 86] CHANDRASEKHARAN M.P. AND RAJAGOPALAN, 1986, MODROC: an extension of rank order clustering for group technology. *Int. J. Prod. Res. Vol. 24, no 5, 1221-1233.*

[Com 86] COMER DOUGLAS, 1984, Operating System Design: The XINU approach. *Prentice Hall, Englewood Cliffs.*

[Croo 85] CROOKALL JOHN R., 1985, Planning and simulation of FMS. *Annals of the CIRP Vol. 34/1/1985, 387-390.*

[Duf 87] DUFFIE NEIL A. AND PIPER REX S., 1987, Non-hierarchical control of a flexible manufacturing cell. *Robotics & Computer Integrated Manufacturing, Vol 3, No. 2, pp. 175-179.*

[Eve 86] EVERAARTS D.C., BAKKER J.J.A., 1986, Specificaties en evaluatie van het SFFS concept, *Delft University of Technology, Delft, 60 pages, Laboratory for Manufacturing Systems, Rapport WPS 86.040 (Dutch).*

[Jut 86] FIX-STERZ JUTTA, LAYE GUNTER UND SCHULTZ-WILD RAINER, 1986, Flexibele Fertigungsystemen und Fertigungzellen- Stand und Entwickelungstendenzen in der BDR. *VDI-Z, Bd. 128, Nr 11.*

[Gim 85] GIMBEL H.F. UND GRANOW ROLF, 1985, Rationaliserung der Werkstattfertigung durch integrierte NC Organisation, Gesamtkonzept eines realisierten systems. *Werkstatt und Betrieb 118, 11, 744-748.*

[Gim 86] GIMBEL H.F. UND GRANOW ROLF, 1986, Rationaliserung der Werkstattfertigung durch integrierte NC Organisation, Realiserung der grundfunktionen. *Werkstatt und Betrieb, 119, 2, 99-104.*

[Gim 87] GIMBEL H.F. UND GRANOW ROLF, 1987, Rationaliserung der Werkstattfertigung durch integrierte NC Organisation, Betriebsmittel organisation. *Werkstatt und Betrieb, 120, 9.*

[Ham 87a] HAMMER H., 1987, Flexible manufacturing cells and systems with computer intelligence. *Robotics & Computer Integrated Manufacturing, Vol 3, No. 1, pp. 39-54.*

[Ham 87b] HAMMER HELMUT, 1987, Veränderungen in der Produktionstechnik durch flexible Automatisering. *Flexibele Automatiserung beim Bohren und Fräsen, Werner und Kolb product documentation, Berlin.*

[Hat 85] HATVANY J., 1985, Intelligence and coorporation in heterarchic manufacturing systems. *Robotics & Computer Integrated Manufacturing, Vol 2, No. 2, pp. 101-104.*

[Hea 86] HEARN D.A. AND DONNELLAN M.A., 1986, The integration of FMS, Just-in-time and LAN technology into a common manufacturing system. *Proc. 5th. Int. Conf. Flexible Manufacturing Systems, 179-192, IFS, Bedford UK.*

[Hen 86] HENDERSON M.J., 1986, Economical computer integrated manufacturing. *Proc. 5th. Int. Conf. Flexible Manufacturing Systems, 167-178, IFS, Bedford UK.*

[Huy 86] HUYSENTRUYT J., 1986, CIM-OSA, a computer integrated manufacturing approach based on the open system approach. *Proc. 5th. Int. Conf. Flexible Manufacturing Systems, 215-224, IFS, Bedford UK.*

[Iwa 84] IWATA K., OBA F., YASUDA K., 1984, Simulation for Design and Operation of Manufacturing Systems. *Annals of the CIRP Vol. 33/1/1984, 335-339.*

[Kaas 87a] KAASHOEK J.C., 1987, Een gedistribueerde besturing van een Flexibel Fabricage Systeem. *Delft University of Technology,*

Delft, 72 pages, Laboratory for Manufacturing Systems, Rapport WPS 87.069 (Dutch).

[Kaas 87b] KAASHOEK J.C. AND BAKKER J.J.A., 1987, Load Module Element Performance Specifications. *Delft University of Technology, Delft, 60 pages, Laboratory for Manufacturing Systems, Rapport WPS 87.037.*

[Kaas 87c] KAASHOEK J.C. AND BAKKER J.J.A., 1987, Station Manager Element Performance Specifications. *Delft University of Technology, Delft, 130 pages, Laboratory for Manufacturing Systems, Rapport WPS 87.029.*

[Kaas 87d] KAASHOEK J.C. AND BAKKER J.J.A., 1987, User Requirements DFMS. *Delft University of Technology, Delft, 32 pages, Laboratory for Manufacturing Systems.*

[Lacey 86] LACEY KARL, 1981, What's the real definition of FMS. *Machinery and Production Engineering, November.*

[Lenz 86] LENZ J.E., 1986, General theories of flexible integration. *Proc. 5th. Int. Conf. Flexible Manufacturing Systems, 255-264, IFS, Bedford UK.*

[List 75] LISTER A.M., 1975, Fundamentals of operating systems. *MacMillan Press, London.*

[Mason 83] MASON GEORGE, 1983, 3 million pound FMS for producing turned parts. *Machinery and Production Engineering, 5 january 1983.*

[Mason 86] MASON FRED, 1986, Computerized cutting tool management. *American Machinist, special report 786, May.*

[Mol 83] MOLLO M., 1983, A distributed control architecture for a flexible manufacturing system, *Proc. 3th. Int. Conf. Flexible Manufacturing Systems, 227-240, IFS, Bedford UK.*

[Moo 88] MOONEN O.W., 1988, Implementatie van het load algorithme in het DFMS prototype. *Delft University of Technology, Delft, 80 pages, Laboratory for Manufacturing Systems, Rapport WPS 88.003 (Dutch).*

[Mur 87] MURPHY K.,KAY J., 1987, Automatic identification and information systems. *The FMS magazine, 5(4), 169-172.*

[Must 88] MUST, 1988, MUST simulation software, Manual, version 4.0. *Upward Systems, Rijswijk, The Netherlands (Dutch).*

[Ono 87] ONO Y., 1987, Cell Control System. *Robotics & Computer Integrated Manufacturing, Vol 3, No. 4, pp. 389-393.*

[Phi 87] THE CAM ARCHITECTURE PROJECT TEAM, 1987, CAM Reference Model, version 1.0. *NV Philips, Centre For Manufacturing Technology, CFT Report 13/87.*

[Pro 86] PROTIM PARTHA BOSE, 1986, Basics of AGV systems. *American Machinist, special report no. 784, March.*

[Pur 82] PURDOM PETER B., 1982, The Citroen (CCM) Flexible Manufacturing Cell, *Proc. 2nd. Int. Conf. Flexible Manufacturing Systems, 151-165, IFS, Bedford UK.*

[Ránky 83] RÁNKY DR. PAUL, 1983, The design and operation of FMS. *IFS Publications, Bedford.*

[Rau 87] RAUCH PETER, 1987, Hard- und Software zur Steuerung und Überwachung von flexibelen Fertigungsanlagen. *Flexibele Automatiserung beim Bohren und Fräsen, Werner und Kolb product documentation, Berlin.*

[Reij 88] REIJERS L.N. EN HAAS H.J.L.M. DE, 1988, Flexibele Produktie Automatisering, Deel 1: Numerieke besturing. *De Vey Mestdagh, Middelburg.*

[Roë 87a] ROËLL R.P. AND BAKKER J.J.A., 1987, Pallet Transport Module Element Performance Specification. *Delft University of Technology, Delft, 45 pages, Laboratory for Manufacturing Systems, Rapport WPS 87.030.*

[Roë 87b] ROËLL R.P. AND BAKKER J.J.A., 1987, Tool Module Element Performance Specification. *Delft University of Technology, Delft, 45 pages, Laboratory for Manufacturing Systems, Rapport WPS 87.038.*

[Roëll 87c] ROËLL R.P. AND BAKKER J.J.A., 1987, Operator Module Element Performance Specification. *Delft University of Technology, Delft, 30 pages, Laboratory for Manufacturing Systems, Rapport WPS 87.039.*

[Shaw 87] SHAW M.J., 1987, A distributed scheduling method for computer integrated manufacturing: the use of local area networks in cellular systems. *Int. J. Prod. Res., Vol. 25, no 9, 1285-1303.*

[Shaw 88] SHAW M.J., 1988, Intelligent information processing in FMS, *The FMS Magazine, 6 (3), 137-140.*

[Sie 87] SIEMENS, 1987, CIM-Komponenten: Flexible Fertigungssysteme. *Siemens Trainungs-Center für Automatiserung.*

[Tan 81] TANENBAUM ANDREW S., 1981, Computer Networks. *Prentice Hall, Englewood Cliffs.*

[Tor 82] TORRI L, 1982, Flexible Manufacturing Systems: A modern approach. *Proc. 2nd. Int. Conf. Flexible Manufacturing Systems, 279-297, IFS, Bedford UK.*

[UN 86] ECONOMIC COMMISSION FOR EUROPE (UN), 1986, Recent trends in Flexible Manufacturing. *ECE Information Office, Palais des Nations, Geneva.*

[Vis 88] VISSENBERG C.J.M., 1988, EPS: DNC Message Protocol F700. *Philips Numerical Control, Machine Tool Control, DER6-32.1-0611A004.*

[Vol 85] VOLLMER HARALD, WITTE HARALD UND KLOSE HORST, 1985, NC-Organisation. *Carl Hansen Verlag, München.*

[Wak 81] WAKERLEY JOHN. F., 1981, Microcomputer, Architecture and Programming. *John Wiley and Sons, New York.*

[Ward 85a] WARD PAUL T. AND MELLOR STEPHEN J., 1985, Structured Development for Real-Time Systems, Volume 1: Introduction and Tools. *Yourdon Press Computing Series, Prentice Hall, Englewood Cliffs.*

[Ward 85b] WARD PAUL T. AND MELLOR STEPHEN J., 1985, Structured Development for Real-Time Systems, Volume 2: Essential Modelling Techniques. *Yourdon Press Computing Series, Prentice Hall, Englewood Cliffs.*

[Ward 86] WARD PAUL T. AND MELLOR STEPHEN J., 1986, Structured Development for Real-Time Systems, Volume 3: Implementation Modelling Techniques. *Yourdon Press Computing Series, Prentice Hall, Englewood Cliffs.*

[War 85] WARNECKE H.-J. AND STEINHILPER R., 1985, Flexible Manufacturing Systems. *IFS Publications, Bedford.*

[Wer 87] WERNER, 1987, Control Station WERNER SC1 for the Computer Controlled Workpiece and Tool Supply at Maching Centers and Flexible Manufacturing Systems. *Werner und Kolb product documentation, Berlin.*

[Wil 67] WILLIAMSON D.T.N., 1967, Ein neues Fertigungsverfahren, *TZ. f. prakt. Metallbearb. 61, Heft 9.*

[Zee 88] ZEESTRATEN M.J., 1988, A look-ahead dispatching procedure. *Delft University of Technology, Delft, Submitted to the International Journal Of Production Research.*

[Zee 89] ZEESTRATEN M.J., 1989, Scheduling for Flexible Manufacturing Systems, *Dissertation, to be published, Delft University of Technology, Delft.*

Appendix A

Centralized Implementation

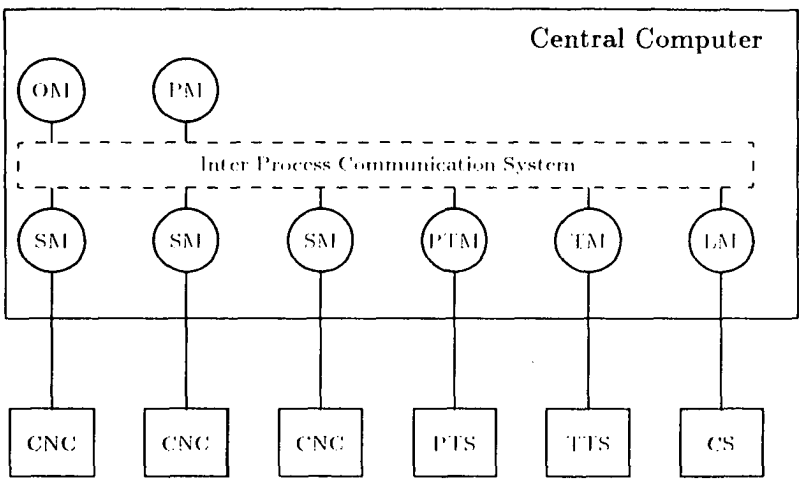
Although the DFMS architecture is essentially a *distributed* architecture, it is possible to design an implementation on one central computer system. The distributed character of the architecture can be maintained by implementing the Station Managers and Function Modules as *separate processes*. The processes communicate with each other by inter-proces communication methods supported by the operating system being used; no data are shared.

The Station Manager processes communicate with the CNCs of the machines by means of serial lines or a Local Area Network. The Pallet Transport Module communicates with the Pallet Transport System Controller, the Tool Module communicates with the Tool Handling System and the Load Module communicates with one or more clamp stations. An overview of this implementation is shown in figure A.1.

A simpler implementation can be obtained by not insisting on the distributed character of the implementation. The whole DFMS control system is now basically implemented as one process. Function Modules and Station managers are now separated at the level of software modules. Communication between Function Modules and Station Managers takes place by sharing data and by passing variables. An overview of this implementation is shown in figure A.2.

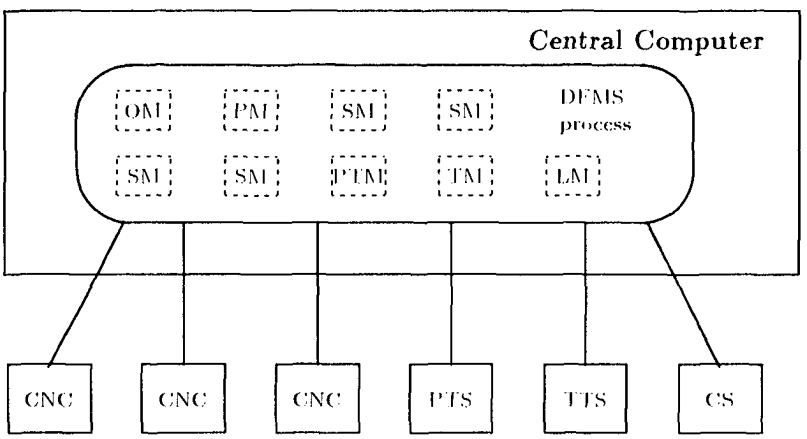
It should be stressed that there are no real advantages of a centralized implementation (over a distributed implementation) of a distributed architecture. If a central implementation was considered, a more effective architecture could have been chosen[1].

[1] Although the architecture and implementation design phases are independent, they do influence each other. The fact that a distributed implementation was considered advantageous, stimulated the development of a distributed architecture.



OM:	Operator Module	LM:	Load Module
PM:	Program Module	CNC:	Computerized Numerical Control
SM:	Station Manager	PTS:	Pallet Transport System
PTM:	Pallet Transport Module	TTS:	Tool Transport System
TM:	Tool Module	CS:	Clamping Station

Figure A.1: A centralized implementation using many processes. The circles represent (operating system) processes.



OM:	Operator Module	LM:	Load Module
PM:	Program Module	CNC:	Computerized Numerical Control
SM:	Station Manager	PTS:	Pallet Transport System
PTM:	Pallet Transport Module	TTS:	Tool Transport System
TM:	Tool Module	CS:	Clamping Station

Figure A.2: A centralized implementation using one process. The dashed boxes represent software modules.

Appendix B

Simulation systems

B.1 Continuous and discrete simulation systems

In order to gain deeper insight in the behavior of a complex system, it is possible to design a *model* of the system, that represents the essential behavior of the system. What is considered to be the essential behavior depends solely on which characteristics of the system have to be analyzed. Two classes of simulation models can be distinguished:

- Continues Simulation models;
- Discrete Event Simulation models.

In a continuous simulation model the system is described by variables that can continuously change value. The behavior of the system is described by mathematical relations, such as differential equations. The *system state* depends on the values of all system variables. Typical applications for continuous simulation models are sociological, economical and biological problems.

In a discrete event simulation model, the *state of the system* changes at *discrete* moments when an *event* occurs. Between these events it is not possible to distinguish between different system states. The set of events that is defined for a model determines which system characteristics can be analyzed.

Flexible Manufacturing Systems can best be described by using a *discrete event simulation* (DES) system[1]. Examples of events that in most FMS simulation models will be used are:

[1] It is not absolutely impossible to apply a continuous simulation method on FMS problems. However, for most purposes it is quite difficult to describe the behavior of a FMS by differential equations.

- the start of an operation;
- the completion of an operation;
- the introduction of a new product into the system.

When just these three events are used, a large part of the actual system behavior is ignored. For example, the fact that products need to be transported and that tools can break during an operation are not taken into account. When an analysis is required in which these facts are considered to be important, a more elaborate model will have to be made.

B.2 Types of discrete event simulation systems

In discrete simulation systems the entities that simulate the behavior of real world elements are called *components*. Components have variables or attributes that may change in time. The values of the variables determine the state of the components. Three different methods can be applied to describe the interactions between the components:

- **Event Description**
  In the event description a list is made of all events in the system. Where the event takes place or which components are influenced by the event is not taken into account. For each event that can occur the activities are described. Activities can be, for example, the scheduling of new actions or the changing of variables anywhere in the system. The simulation system processes event after event, until no more events are left;

- **Activity Description**
  In the activity description a list is made in which all activities and the conditions in which they occur are described[2]. An activity can only be performed when the conditions for the activity evaluate to true. The list is sorted in the sense that the high priority activities come first. When a set of conditions occurs so that the conditions of more than one activity evaluate to true, the highest priority activity is performed. The simulation system analyses all conditions and performs the activities for which the conditions evaluate to true. The performing of the

[2] An activity description simulation system is an example of a rule-based system. Rule based systems are frequently applied in artificial intelligence applications.

activity brings the system into a new state that enables other activities. When the point is reached that no activity can be performed, the simulation stops;

- **Process Description**
  In the process description method each component is described as a process that has interactions with other processes. A *process description* specifies the behavior of a process. Processes can interact, they can exchange data and they can suspend and activate each other. The simulation runs as long as there are events scheduled.

The most modern discrete event simulation method is the process description technique. The advantage of this method is that it is easy to make a very natural description of a real life system.

B.3 Requirements of simulation systems

Special languages or systems can be used to ease the development of simulation models. In this section some requirements of simulation tools will be discussed.

- **Descriptive power**
  A simulation model has to describe the behavior of a real life system. It is important to use a 'natural' description method: a method that seems straightforward to the users of the system. In this respect the process description method is more suitable than for example an activity description method. People tend to view the real life systems in terms of cooperating processes. If the simulation system uses a similar concept, the model will be easier to understand;

- **Functionality**
  To model a system, a large amount of work has to be done. Some of the activities are rather standard and can be part of the simulation system. Examples of software tools that can be applied are:

  - tools to handle the scheduling of events;
  - tools to manipulate queues;
  - statistical tools;
  - tools to present the simulation results in form of charts;

- **Simplicity**
  The amount of time that has to be spent in developing a simulation model depends for a large part on the complexity of the simulation system. A low complexity is an advantage, but should not lead to low functionality.

Stellingen

Behorend bij het proefschrift:

DFMS: Architecture and implementation of a distributed control system for FMS

J.J.A. Bakker, 20 juni 1989

1. Het gebrek aan flexibiliteit van veel flexibele produktiesystemen is niet een tijdelijk, maar een fundamenteel probleem.

2. Het succes van een Flexibel Fabricage Systeem hangt veel meer af van het produktenpakket dat het systeem moet verwerken, dan van de kwaliteit van de systeembesturing.

3. Bedrijven die terugschrikken voor het installeren van een volledig geautomatiseerd Flexibel Fabricage Systeem, kunnen het installeren van alleen een FFS besturing overwegen. De te verwachten hogere organisatiegraad zal bijdragen tot een efficientere produktie.

4. De technische complexiteit van een systeem neemt in hoge mate toe, wanneer een gedistribueerde implementatie wordt toegepast. De conceptuele complexiteit neemt daarentegen echter vaak af.

5. Het is onjuist om aan vakgroepen binnen universiteiten slechts materiële kredieten toe te kennen. Het buiten zicht houden van personele kosten leidt tot een onderwaardering van arbeid.

6. Voor slechts een zeer klein gedeelte van de werknemers zal de toename aan ervaring bij het ouder worden blijvend opwegen tegen het verlies aan daadkracht, energie en ambitie; dit dient consequenties te hebben voor salariëring en zou moeten leiden tot variabele pensioen leeftijd.

7. Zelfs van tolerante samenlevingen kan niet worden geeist dat zij zich aanpassen aan de levensstijl van culturele minderheden.

8. De taalbeheersing van de huidige generaties studenten wijst op de noodzaak van het afnemen van een taaltest als onderdeel van een universitair toelatingsexamen. Multiple-choice vragen dienen te worden vermeden.

9. De DFMS architectuur kan worden ingezet op een zeer breed toepassingsgebied: van computerarchitectuur tot het coördineren van 'networks of co-makers'.

10. Bij de opleiding van technici is het ontwikkelen van gezond verstand en van analytisch denken belangrijker dan het overdragen van specifieke kennis van het vakgebied.

11. Het draagt aan de geloofwaardigheid van milieugroeperingen bij, indien zij zelf mede de economische gevolgen ondervinden van de door hun voorgestelde maatregelen.

12. Ingenieurs hebben, als leden van een beroepsgroep, geen speciale maatschappelijke verantwoordelijkheid; zoals ieder ander hebben zij wel een persoonlijke verantwoordelijkheid.

13. Het gelijktijdig bestuderen van techniek en menselijke fysiologie geeft een ontnuchterend beeld van technische prestaties.