Massey University
School of Science
Institute of Technology and Engineering

A Distributed Shop Floor Control System Based on the Principles of Heterarchical Control and Multi Agent Paradigm

A dissertation presented in partial fulfilment of the requirements for a
PhD degree
in
Production Technology – Computer Integrated Manufacturing (CIM) Systems
at
Massey University

Goran D. Colak

Year 2004

Acknowledgement

I would like to express my appreciation to my first thesis's supervisor Dr Saeid Nahavandi for giving me an opportunity to start this research project at Massey University.

Special thanks and gratitude go to my second supervisor, Prof Donald Barnes, for his altruistic interest and support during my research work, for suggestions related to the structuring of the thesis, and for all his efforts, patience, and precious time that he put in while reviewing and editing the thesis.

This dissertation is dedicated to my son Avram and my wife Jasmina for their unconditional love, patience, and sacrifice during my study.

Preface

In progressive firms, major efforts are underway to reduce the time to design, manufacture, and deliver products. The programs have a variety of objectives, from reducing lead-time to increasing product quality. The process of improvement starts with customer requirements, which in turn lead to customer-driven manufacturing, incorporating customer requirements more directly into the manufacturing processes. Forecasting customer requirements has not become any easier, in fact, just the contrary. The implication is clear: that if demands cannot be forecast, the manufacturing function must be designed to respond to these demands. To do this rapidly, more and more of the manufacturing decisions are being delegated to the factory floor. To paraphrase; the customer is saying what is to be made, the due date is now, and the work force is figuring out how to do it online. As the manufacturing world moves toward the "zero everything" vision of the future (zero inventory, zero set-up time, zero defects, zero waste), fundamental changes will take place in the factory. These changes will necessitate changes in manufacturing planning and control systems and particularly changes in planning and control on the shop floor level. This dissertation addresses the possible direction that some of these changes might take on the shop floor.

The starting preamble of this research is that forecasting in certain type of manufacturing systems is not possible. An example might be systems in which product orders arrive randomly, such as manufacturing facilities involved in production of replacement spare parts). Additionally, in many other manufacturing systems, forecasting generates results that are of a very low level of certainty. In many occasions they are practically useless, since they are applicable only for short time horizons. As an example, small-quantity batch manufacturing systems usually operate under conditions where frequent disturbances make this production unstable at all times. Therefore, addressing these systems, the main idea embodied in this dissertation could be expressed as follows: *"Instead of focusing efforts on how to improve the old, or develop new methods for controlling material flows in manufacturing systems, methods that are solely based on the main premise of predicting the future circumstances, this research takes another course. It considers an alternative approach – developing of manufacturing control mechanisms that are "more reactive" to the changes in the systems and "less dependant on prediction" of future events.*

It is believed that the modern job shop manufacturing facilities, such as mentioned above, can further increase their competitiveness by adopting approaches for shop floor control systems that are discussed in this research study. This is because the proposed system is capable, both dynamically and in real time, of promptly responding to frequent changes in production conditions, always attempting to find the best possible solution for given circumstances.

The embodied philosophy in this project for resolving computationally difficult and complex scheduling problems in manufacturing systems is not new. However, it introduces a concept and methodology that makes development of a distributed multi-agent system a reality. It does so by using common hardware, computer operating and network systems, and programming languages and technologies. A developed test-bed application that can run on theoretically unlimited number of computers connected into a local area network (LAN), demonstrates the work of distributed multi agent systems, and proves that such a system can be developed using common computer hardware and software technologies, in a very affordable, and inexpensive way.

This dissertation represents yet another effort in the vast research endeavour directed towards the building of competitive manufacturing facilities. If any part of this work is going to be used for these purposes in the future and serve as a small contribution to this endeavour, the author will consider this study successful.

Table of Contents

A list of figures

A list of figures – in appendixes

A list of tables

Abstract

This practically oriented research study concerns the design and implementation of the core part of a distributed heterarchical shop floor control (SFC) system based on the multi-agent paradigm. The system has been designed and developed with the primary aim of supporting production operations in discrete part manufacturing systems of a job-shop type. The "modus operandi" of the system is envisaged to be beneficial, and of a particular value, for job shop manufacturing systems that are characterised with:

- A large number of machine tools (manually operated or fully automated CNC machines);
- Low volume - high variety batch production (that is, in which products are fabricated in small quantities and many different product types); and
- Random arrival of production orders of different product types and quantities, (order rates, types, and sizes are not known in advance and are hard to predict).

A classical example of an appropriate manufacturing system is one that produces spare parts as single items or in small batches.

The project used multi-agent system theory and recent developments in software technology to solve the problems that concern modelling, design, development, and implementation of a proposed heterarchical SFC system. The project demonstrated (by using integrated simulation) how difficult and complex scheduling and resource allocation problems in job shop manufacturing systems could be successfully resolved in an on-line and real time manner.

Basic production and control units in the proposed approach were organised around workstations. Manufacturing operations inside workstations were simulated while the interaction among workstations (including communication and negotiation processes) was conducted in the same manner, as it would be in the real-life systems. These simulated activities as well as the "testing capabilities" of a workstation agent were integrated seamlessly into a single software package – a workstation agent. (The term "testing capabilities" refers to the feature that enables a user, for example, to capture a message that is sent to the workstation agent and then to postpone the message processing, to change the content of the message, or even to reject the entire message to see what effect this has on the total operation.)

The dissertation describes the structure for both the proposed manufacturing system resource model, which is based on *manufacturing workstations* as basic production units, and the heterarchical shop floor control system model, which is based on *workstation agents* as the basic control units. The heterarchical control system was demonstrated on a test-bed network application that was created with the objective of validating the concepts, and verifying the feasibility of the proposed concept. The dissertation outlines the main design aspects of the application (it describes basic modules of the production workstation agents) and describes the way in which the system operates, using an example of a simple workpart travelling through the system.

Chapter 1. Introduction

Today, severe market competition puts higher demands on contemporary manufacturing systems to meet rising customer specific requirements. To increase responsiveness to changes in the market, manufacturing systems often need to be able to simultaneously produce multiple product types in small batches, even in lot sizes of one, concurrently providing lower lead times and offering the ability for quicker and less expensive system reconfigurations. To accommodate such a production, it is believed that rigid and inflexible centralised and hierarchical manufacturing control structures should be replaced with heterarchical control structures that are made up of multiple, distributed, cooperative, locally autonomous production entities. These heterarchical control structures are more adaptable, extensible, reliable, fault-tolerant, and more easily reconfigurable [Saad et al., 1997]. This thesis reports the results of an application oriented research project on a distributed, multi-agent heterarchical shop floor control (SFC) system that was undertaken at the Institute of Technology and Engineering at Massey University, Palmerston North, New Zealand.

This chapter is structured as follows.

Section 1.1 outlines market demands and requirements imposed on contemporary manufacturing systems, addresses some of the major challenges for contemporary flexible manufacturing systems, introduces heterarchical control systems, provides a few reasons why heterarchical based manufacturing systems are not widely presented in industry nowadays, and discusses issues regarding modelling and simulation of heterarchical SFC systems.

Section 1.2 discusses the reasons for conducting research in the area of heterarchical manufacturing control systems, proposes the research that is to be undertaken in the research project, and provides justification why such a proposal is made.

Section 1.3 defines the research domain.

Section 1.4 presents the targeted manufacturing environment - the manufacturing systems towards which this research project is primarily directed. It also presents two basic requirements that need to be satisfied in modern-day manufacturing systems in order to stay competitive in the market place in the years to come.

The structure of the thesis is outlined in Section 1.5.

1.1 Preliminary remarks

The reducing tariff barriers between national boundaries, has meant that the competition between manufacturing companies has significantly increased during the last two decades. The new circumstances of an open and global world market have influenced many manufacturing enterprises all over the world. To remain competitive in global markets, modern manufacturing companies need to respond to a number of ***consumer demands***. For example, there is demand for:

- Constant product innovation;
- Better design and increased product functionality (introducing more new features);

- Greater product variety (supplying more different types of the same product);
- Increased product reliability;
- Better product quality;
- Providing products at lower or constant prices;
- Low-cost customisation;
- Better service and customer support;

To satisfy the customer's increasingly specific and demanding requirements, ***modern companies need to***, for example:

- Provide rapid product realisation. By designing products for manufacture and assembly, development cycle for new products and time-to-market can be significantly reduced.
- Be able to simultaneously produce multiple product types as well as providing lower lead times. The changes in production of different product types must be quick and cost effective;
- Move away from "make-to-stock" towards "make-to-order" production. Increase the circulation of capital;
- Reduce overall stock levels including Work-In-Process (WIP) inventories. All materials in the production process represent a frozen investment till the product is released to the market. Also, due to more frequent changes in production some product components might become obsolete as changes are made to meet changing consumer demands.
- Provide greater flexibility with respect to product quantity and variety. Offering products in greater variety increases customer satisfaction and thus the chance of more sales. This means increasing machine and product flexibility. Machine flexibility is the ability to change and reconfigure machine functionality to handle a greater variety of product mixes effectively. Product flexibility is the ability to adapt parts/products selection and assembly sequencing to best suit the current system configuration and customer demands.
- Provide workforce flexibility. Having staff (teams of multi-skilled personnel) who are cross-trained increases overall flexibility of the system;
- Provide a rapid response to specific customer requirements. Manufacturing facilities have to be able to respond quickly to requirements that often cannot be forecast;
- Continuously improve the manufacturing processes. This includes replacing obsolete technologies with competitive, highly flexible facilities. New technologies have always been major enablers of competitive advantage;
- Have a commitment to total product and service quality. The quality of products needs to be high and consistent.
- Use a high level of automation. Automated processes can improve product quality, reduce the need for unskilled workers and provide greater flexibility.
- Minimise lead times. Increasing delivery reliability and shorter delivery times will help any company to keep present customers, and to win new ones.
- Reduce lot sizes. In order to reduce frozen capital and be more responsive to customer demands, it is an advantage to have manufacturing systems that can fabricate products in lower volumes, with batch sizes even approaching unity;

- Reduce overall production costs. This includes the reduction of all cost components, from material and labour, to energy and overheads.
- Increase productivity. Getting more output per input.
- Increase efficiency. Achieve higher resource utilisation.

As a result of worldwide industry competition, the interest in Computer Integrated Manufacturing (CIM) systems was rekindled in the second half of 1980's. CIM systems are viewed as an effective strategy to cope with the demanding requirements imposed upon modern manufacturing companies. They improve manufacturing responsiveness in small to medium volume discrete parts manufacturing operations. However, because CIM seeks to integrate the entire manufacturing enterprise, these systems often come in over budget, and do not provide the promised flexibility. This is particularly so for small companies that could benefit the most from CIM. Furthermore, the evolution of CIM and promised benefits has been slower than expected. This can be directly attributed to the high software development and maintenance costs, and the difficulty in achieving the required levels of integration between systems [Smith et al., 1996]. These problems are especially evident in the development of the Flexible Manufacturing Systems (FMS)[1] as an integrated part of CIM. Therefore, while the concept of FMS has been well received in industry, the ability to consistently build successful systems remains elusive. In addition to the above reasons, another primary reason for FMS failures is the lack of appropriate control structures [Tawegoum et al., 1994]. Beyond a reflection of the organisational structure, FMS control systems have been designed using the traditional hierarchical model for two primary reasons: firstly, the large size of the systems dictates a delegation of information and control and, secondly, the time-horizon demands of the system require a stratification of decision-making procedures.

Realising automatic control, where material flow and information flow are arranged and synchronised in such a way as to maximise global performance of the system, remains the major challenge to the contemporary flexible manufacturing systems [Smith et al., 1996]. To achieve this, both 1) the factory machinery (which has to accommodate many diverse manufacturing processes), and 2) the accompanying production control system (which is responsible for timely execution of these processes), must be designed in such a manner that they are able to accommodate frequent changes in the surrounding environment over time. Therefore, contemporary manufacturing systems need to offer the ability for quicker and less expensive system responses to frequent changes in production operations.

Unfortunately, in trying to accommodate such production, it has been found that conventional manufacturing control structures impose serious limitations for short-term, make-to-order production. With the increasing size and complexity of the manufacturing system, conventional manufacturing control schemes, which are based on a centralised and hierarchical structure, become prohibitively complex and unreliable. In this regard, it becomes increasingly important that the Shop Floor Control (SFC) system, which is responsible for planning, scheduling, and controlling the equipment on the shop floor, has an architecture that is modifiable, extensible, reconfigurable, adaptable, and fault tolerant [Saad et al., 1997]. In order to achieve this, manufacturing computer control architectures have to be modified, from the more rigid centralised and hierarchical, to more decentralised, distributed heterarchical architectures. (Many authors argue that the key to a successful manufacturing system is the decentralisation and distribution of information and control.) These systems can reconfigure dynamically

[1] FMS are composed of a number of computer numerically controlled (CNC) machines and material handling devices (robots and Automated Guided Vehicles (AGVs)), interconnected through high speed communication networks using sophisticated software to operate properly.

depending on production requirements and the state of the manufacturing processes [Duffie and Prabhu, 1994]. To overcome the drawbacks inherent in the conventional FMS (discussed in more detail in Section 5.4 "Overview of manufacturing control architectures and their main characteristics"), a system-based approach that incorporates distributed heterarchical control and software engineering techniques is desired, for constructing new control systems.

Therefore, to cope with the ever-changing market trends, academia and industry have been working towards employing the principles of heterarchical control structures[1]. Heterarchical systems consist of an aggregation of multiple, distributed, cooperative, locally autonomous production entities, that act on the basis of information exchanged amongst each other to achieve their individual and system goals. By adopting these distributed heterarchical control architectures, it is believed that this new generation of manufacturing control systems will provide higher levels of adaptability, extendibility, reliability and easier reconfiguration. Although any manufacturing facility would benefit from these systems, these systems are of particular importance for low-volume, high variety job shop manufacturing systems,[2] in which order rates and sizes are not known in advance.

Despite the growing interest among researchers and subsequent research [Baker, 1991, 1995, 1996, 1998, Brennan & William, 2000, Crowe & Stahlman, 1995, Dewan & Joshi, 2001, Duffie 1982, 1990, Duffie & Piper, 1986a, 1987, Duffie & Prabhu, 1994, Duffie et al. 1988, Lewis, 1981, Lewis et al., 1987, Lin & Solberg, 1992, 1994, Macchiaroli & Riemma, 2002, Maturana et al., 1999, Ottaway & Burns, 2000, Parunak, 1987, 1998a, 1998c, 2000a, 2000b, Parunak et al., 1998, Roy & Anciaux, 2001, Saad et al., 1995, 1997, Shaw, 1985a, 1985b, 1987a, 1987b, 1988, Shen & Norrie, 1998, 1999, Veeramani 1992, 1994, Veeramani et al., 1993, 1997, Wang & Usher, 2002, Wu & Sun, 2002], and that almost two decades have passed since the pioneering work in the domain of distributed heterarchical manufacturing control systems [Lewis, 1981], there have been very few implementations of heterarchical systems. Reasons that have been claimed to explain why heterarchical based manufacturing systems are not widely used in industry nowadays, are that the theoretical optima cannot be guaranteed, predictions for the system can usually be made only at the aggregate level, and autonomous entities can become computationally unstable [Parunak, 1994]. The same author has quoted some further grounds that may be more important to explain the limitations of the applications of multi agent systems in practice [Parunak, 1996b]. These limitations could be applied to the entire class of heterarchical control systems. They are:

- Lack of tools and platforms for developing agent based applications;
- Lack of formal proof theory for agent based systems;
- Reluctance to change;
- Lack of standards;
- Little support for debugging and testing distributed asynchronous systems, due to the difficulties of data acquisition;

---

[1] In the late 1980s, in parallel with the growing research efforts and successful implementation of distributed computer systems, the idea of heterarchical control architecture was developed [Dilt *et al.* 1991].

[2] It is important to note that the term "job shop" has different usage *in the technical literature* on the structure of manufacturing factories compared to usage *in general engineering practice.* The technical term "job shop" refers to the type of manufacturing systems, in which an arrangement of machine tools on a shop floor is based on a functional layout, and in which the part types being manufactured and the sequence of manufacture is highly variable. Typically in engineering usage, "job shop" describes a small "one off" engineering facility. In this dissertation the technical term for "job shop" is used.

- Slow technology transfer from academia to industry.

Yet another disadvantage in dealing with heterarchical shop floor control (SFC) systems is a lack of appropriate research tools that could be used for investigation of heterarchical manufacturing control structures.

Investigating behaviour of any manufacturing systems can be a difficult task, because of the complex interactions among many entities (variables) involved in the system. Modelling and simulation techniques are one of the most appropriate tools for such evaluations. However, it should be noted that in modelling manufacturing systems, traditional modelling-simulation approaches typically suffer from several weaknesses. Firstly, although they provide powerful constructs for modelling basic manufacturing system behaviours, the abstractions used in the modelling often do not accurately map to the system being modelled. Secondly, they do not support human supervisory control, which is an important feature of most manufacturing systems. Finally, it is difficult to represent control and decision-making processes. This difficulty is even more emphasised when considering heterarchical control systems. Because the heterarchical SFC systems include co-operation (negotiation) processes among autonomous, distributed production entities evaluating different controlling strategies and the merits of heterarchical control systems is an even more difficult task. Often it is almost impossible to perform this task with most of the existing simulation packages.

In addition, many experiments that have been undertaken on the performance of heterarchical control systems are conducted by using commercial modelling and simulation software packages. Since these packages are not specifically designed to support such simulation it leaves doubt as to whether or not the performance of these systems is biased by the simulation. (That is, performance of heterarchically controlled systems may be influenced by capabilities of simulation packages rather than by the control software itself). Furthermore, many problems generated by communication issues might be omitted.

At the time when this project was commenced, commercially available simulation packages that were able to simulate heterarchically controlled manufacturing systems did not exist. A problem that modern "desktop" simulation packages face when dealing with heterarchically controlled manufacturing systems is the problem of simulating relationships that exists among distributed entities that are in real systems, conducted via a network of computers. A desktop application would probably, though not necessarily, simplify these relationships (in terms of imposing limitations on applying different negotiation/scheduling strategies and simulating the network circumstances - congestions). Hence, the variety and validity of the simulations conducted would be in question. Most importantly, when simulations of manufacturing processes are run "faster than real time" these negotiation processes should also be run "faster than real time". In such a case, if the simulation does not include simulation of real network traffic conditions, the accuracy and validity of the results of such a simulation would be jeopardised. Therefore, developing a desktop application for the simulation of heterarchical shop floor control systems is a very difficult and challenging project. The absence of such desktop applications was one of the primary motives (as discussed in the next section) for development in this project of a "network" application for simulation (testing) of heterarchical shop floor control systems.

1.2 Motivation for conducting this research project

Recent market trends and demanding requirements imposed on the contemporary manufacturing systems (addressed at the beginning of the previous section) can be viewed as general motives for conducting the research work covered by this dissertation. Further research was motivated by the next two issues.

1) The issue of controlling large computerised manufacturing systems. Due to inadequate preparation[1] the performance of the earliest FMSs were disappointing. Consequently, Flexible Manufacturing Cells (FMC) emerged as smaller, less expensive and more tractable installations. However, in this conservative approach to building manufacturing systems, each consisting of several FMCs, *"the optimisation of individual FMCs will only lead to sub-optimal performance of the entire manufacturing system"* [Veeramani et al. 1993]. The same authors argued that to attain more efficient performance it is *"necessary to integrate the FMCs into one system"*, and that this realisation has encouraged studies into the *"feasibility of large FMSs that could possibly consist of 50 or more intelligent Computer Numerically Controlled (CNC) machines. ...The wisdom of investigation into the operation and control of large FMSs lies in the increased flexibility and efficient sharing of resources that would be facilitated by such systems."*

2) Absence of a sound software model for modelling and simulation of heterarchically controlled systems. The reasons for the lack of implementations of heterarchical systems, in particular, lack of tools and platforms for developing agent based applications, and lack of formal proof theory for agent based systems, provided further encouragement for conducting this research. Because of loosely coupled elements and distribution that are involved within heterarchically controlled FMSs, using conventional modelling and simulation techniques to investigate these systems is difficult. Scheduling and control problems are in any case hard to address in classical centralised and hierarchical control schemas, because they are complex and context-dependant. Heterarchical distribution makes this situation even worse.

The above considerations resulted in proposing a practically oriented research project:

- To construct a complete, functional heterarchical control model (limited at the shop floor level) as for a real manufacturing environment. The model would provide a formal method for the development of heterarchical SFC software.
- To develop a software system model (an accompanying test-bed network application), that could be used as a research tool for studying complex control problems, and conducting experiments in the field of distributed, heterarchical manufacturing control systems. The application would prove the feasibility of the proposed concept and, also, with a possibility to combine real and virtual production units, would provide a suitable modelling and simulation environment.

The successful implementation of heterarchical control systems requires a sound software model. Because of the advances that have been recently achieved in computer science (note that we are referring at the time before the project commenced – 1997), it was believed that many of the technical, hardware and software computer resource limitations, encountered by previous researchers over many years, have been overcome (or will be in very near future). The following two developments were justification for this standpoint. Firstly, there was the emergence of powerful hardware devices and the development of new operating systems that made PCs more powerful and reliable, than for instance, the mainframe computers were just a few years ago. This is the best-illustrated by comparing common PC platforms and developing environments at the time when the project was commenced in 1997 and the time when final amendments on the thesis were made, in 2003. Secondly, there were many advances in software in recent years. These included improvements in high-level programming languages, database technologies, and appearance of new programming systems for rapid application development (RAD) such as -

[1] This refers to the accompanying production functions that go together with immediate execution of processing operations on the shop floor (for example, timely production planning and scheduling, complex and rigid control software, lack of software integration with the other production subsystems, high maintenance costs, and so forth.) All these lead to the paradox of actual "inflexibility" of flexible manufacturing systems.

Visual Basic and Delphi. Examples included new accompanying specifications, standards, and technologies, such as ODBC (Open Database Connectivity), ADO (ActiveX Data Object), OLE (Object Linking and Embedding), ActiveX controls, etc. These all provided researchers with powerful programming tools for developing new, complex prototype applications.

It has been a challenge, at least in the research domain, to apply those advances in the area of heterarchical manufacturing control systems, and to transform the relevant theory of heterarchical manufacturing systems into tangible, practical systems. One of the intriguing questions was: *What can be done in terms of developing a practical test-bed network application of a distributed, heterarchical shop floor control system using standard PCs, commonly available computer operating and network systems, one of the prevalent programming languages, and by employing "off-shelf" modern software development tools?* Therefore, a decision to attempt the development of an appropriate operational model (a test-bed network application) of a distributed heterarchical SFC system based on a multi-agent paradigm was made. The reason for making such a decision was based on four main considerations that are discussed next.

Firstly, it has been envisaged that through the development of such an application (in an interactive process of designing, coding, testing, and analysis of the constituent system modules), the finest details of heterarchical SFC system would be perceived, investigated, and thoroughly understood. Such an approach leads to an "open system architecture", which allows the system to be upgraded at a later stage with new features, and to make constant, ongoing improvements over time. It was anticipated that this development would help to better understand the overall issues of designing heterarchical control systems, for example, how to devolve a system appropriately, how to distribute control and responsibilities among system entities, and how to establish relationships among these entities in heterarchical manner. Also, it was expected that we would be able to precisely discern the key attributes of the distributed system entities and the way in which these attributes are to be locally organised. The application would enable a definition of the detailed requirements of the distributed SFC system entities. *In another words, it was believed that there is no other way to get full comprehension and the details of a distributed heterarchical shop floor control system but to try to develop a prototype.* Both full insight of overall requirements, and the scope of work required for practical realisation of such a project would be perceived through such development.

Secondly, because control and data are distributed in the heterarchical systems, it is believed that *the modelling and simulation of these systems also has to be distributed in a manner similar to the systems being simulated.* Since the cooperation among distributed elements (workstations) in these systems involves passing messages, it would be difficult to simulate it with most existing simulation packages.

Thirdly, *such an application could be used for conducting a whole set of experiments related to the usage of different dispatching rules, applied in a distributed manner, in heterarchical SFC systems.* Also, measuring the performance of the manufacturing system (such as due date deviations, sojourn time, etc.) can be performed more accurately[1] under circumstances that are much closer to those that exist in the real systems. Therefore, since the application is based on a method of "integrated simulation", then, in comparison with commercial software packages for general simulation, the experiments would provide results much closer to the actual situation in a

[1] The accuracy of a simulation depends on the accuracy of the simulation model and the input data. To ensure the accuracy of the simulation results, the simulation model employs workstation agents that are designed as a "clone" of the real agents. In other words, the same software package that represents a workstation agent in a simulation model can be used (with some minor modifications and extensions) in a real manufacturing system.

manufacturing operation. (In such a simulation, workstation agents are used as they would be in the real systems, that is, control signals and messages are sent exactly as they would be in the real system but with the difference that the workstation's manufacturing activities are simulated). The simulation closely reflects the current state of the heterarchical SFC, thereby ensuring the accuracy of the representation of the simulated data.

Finally, and more importantly, it is believed that the application could offer *the possibility of conducting experiments with different negotiation algorithms, determining the effect of these algorithms on the requirements, and the performance characteristics of a distributed SFC system from a communication point of view.*

Such an experimental system was successfully designed and constructed, and it is discussed in detail in this thesis.

1.3 Research domain

Before progressing further with the matter of interest in this project, the boundaries of a research area should be defined. Therefore, this section determines a research domain from the perspective of a broader FMS framework. A review of the most important issues related to FMS is given in [Gunasekaran et al., 1995]. These issues can be divided into five phases [Ben-Daya, 1995]:

(1) Design. The design problem starts when the need for automation and flexibility in making work-parts and products is realised. The decisions involved, at this stage, include:

- The system hardware;
- Computer system and control mechanisms;
- FMS layout[1];
- Part family selection.
- The selection of a production planning philosophy;
- A decision hierarchy for the complex FMS environment;
- The design and implementation of a computerised system and its human/machine interfaces.

(2) Aggregate production planning. This phase includes the long-range manufacturing and procurement decisions related to the material resources and plant capacities. The result of a production planning process, which is coordinated with sales objectives (forecasts), resource availabilities, and financial budgets, is a production plan. The plan has to be accomplished if the company's overall objectives are to be met. By providing a production plan, management can plan and control subsequent master production scheduling decisions.

The main output of the aggregate production planning phase is a master production schedule (MPS). The MPS is based on the calculation of the overall requirement for end-product production, which is then adjusted, in the light of the capacity levels available and other restraints, to result in a feasible amount of work to be done over the time under consideration. MPS defines the set of work-parts (part mix), production rates, and lot sizes that will be produced during the planning period. Being a statement of production and not a forecast, MPS

[1] In the case of job shop flexible manufacturing systems (FMS) that are based on processing (functional) layout, this function deals with the grouping of machines of similar types into identical machine groups. Each machine in a particular group is then able to perform the same operations.

provides the basis for utilising plant capacity effectively, attaining the firm's strategic objectives as reflected in the production plan, and resolving trade-offs between manufacturing and marketing.

(3) Short term planning or system setup. This phase interfaces the aggregate production planning outputs with the day-to-day operation of an FMS. The decisions involved in this phase include:

- *Part selection.* Part selection deals with the problem of determining a subset of the parts, from the given set of part types that are to be processed immediately and simultaneously during the short-range planning horizon. The selected set of parts must only use a feasible allocation of resources. The need for the part type determination arises because the system has capacity limitations. For example, the tool magazine capacity restricts the number of tools that can be mounted on the magazines, and hence this limits the number of different parts that can be processed.
- *The production ratio problem.* Given the part types selection, this function determines the relative part type mix ratios at which the selected part types should be produced over time.
- *The resource allocation problem.* This process deals with the allocation of a limited number of machines, tools, jigs, fixtures, and pallets of each type among the selected part types [Gunasekaran et al., 1995].
- *The machine loading problem.* When the work-parts to be processed simultaneously have been selected, the machine loading function is concerned with the problem of allocating work-part operations and the required tools amongst available machines for a given product mix, so that some selected system performance criterion is optimised. Constraints of the machine loading problem typically include the number of cutting tools that can be simultaneously located on the tool magazine and the capacity of the machine [Gunasekaran et al., 1995].

These four problems areas are linked to each other. There are different strategies for handling their interdependence. Some authors combine a few of these problems in one formulation; others disaggregate these problems and develop sequential or iterative solution procedures.

The output of the short term planning phase is based on detailed material and capacity planning activities. It defines for example an assignment of processing operations to machines, the tools required for processing the selected part types to be loaded into the machine's tool magazines, and an allocation of fixtures to parts.

(4) Shop floor scheduling. This phase determines the detailed routing of work-parts through the machines and determines the start and completion times for each activity.

(5) Shop floor control. This phase deals with the actual operation of the system. The decisions made at this level include:

- The design and implementation of procedures for handling machine tool and other breakdowns;
- Periodic and preventive maintenance;
- Quality control;
- On-line data collection and processing.

The research domain of this project is a SFC system with the focus on the important scheduling problem in an FMS. Figure 1-1 shows the research area in the context of the overall Production Planning and Control (PPC) system.

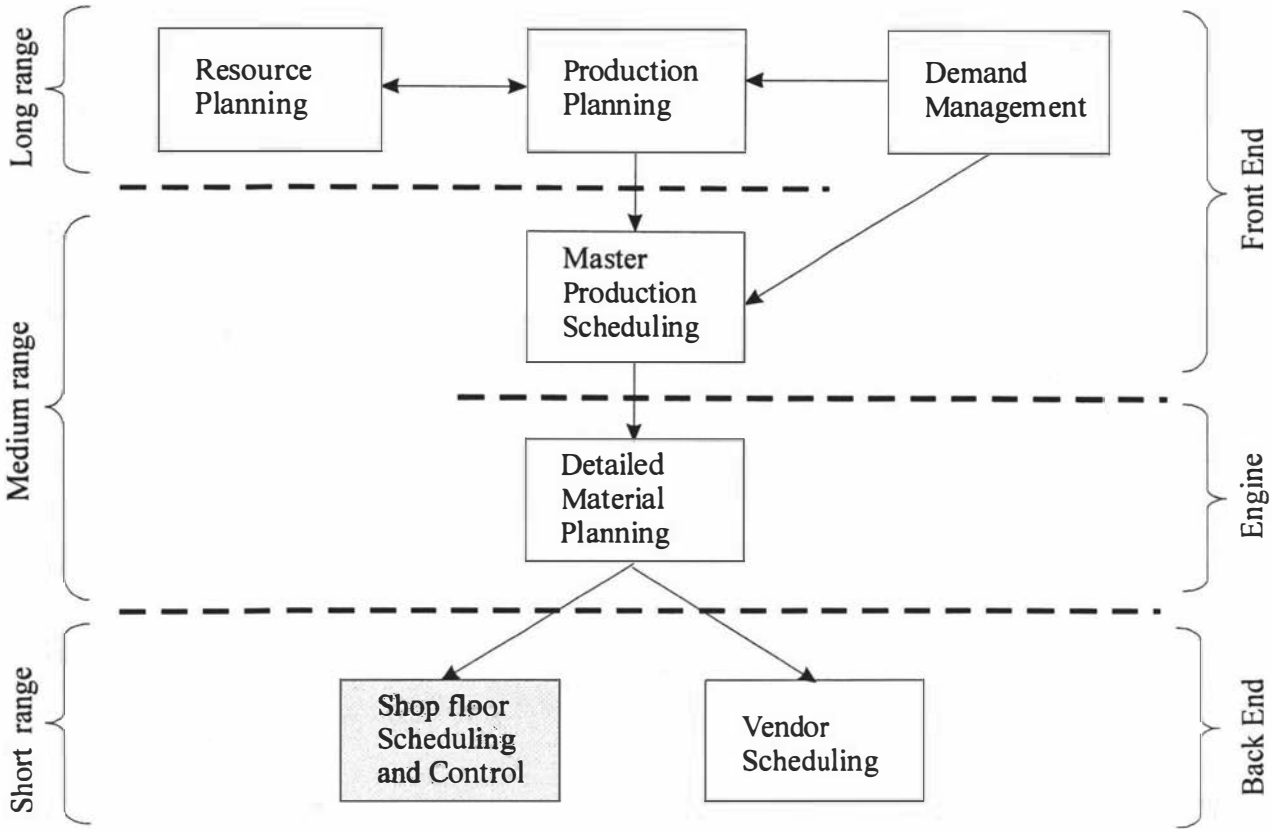


Figure 1-1. The research domain –The Shop Floor Control (SFC) system

Bearing in mind the need to limit the scope of this project, it was not intended to embrace the strategic and tactical planning decision-making process levels of the PPC systems.

1.4 The manufacturing systems targeted in this research

The general market demands, listed at the beginning of this chapter, have caused increasing interest in the research community on the batch, job shop, manufacturing systems. It is believed that these systems, which produce parts in small lots but in a broad variety of different types, will play a very important role in the coming years (see Section 2.1.2 "Importance of batch production"). For this reason the targeted manufacturing environment of this dissertation is primarily on ***low-volume, high variety, job shop manufacturing systems.*** These types of manufacturing systems can be viewed from several different aspects.

1) With respect to a flow of materials, two diametrically opposed shop environments can be distinguished:

- *Flow shop.* In a pure flow shop system, each job flows from upstream to downstream over the machines without any return flow.

- *Job shop.* In a job shop, a job can begin and end at any machine. The flow of material is complex, many jobs may flow in a reverse direction in some part of their route through the factory, and paths are of variable length.

The primary interest of this dissertation is on ***job shops manufacturing systems,*** though the system, which has been developed, can be successfully implemented in any type of flow pattern from a flow shop to a job shop.

2) With respect to the variety of product types that are produced, manufacturing systems can be classified as:

- *General-purpose manufacturing systems.* These systems are capable of producing a broad variety of different product types, the details of which, typically, are not known in advance at the time when a system is designed.
- *Dedicated manufacturing systems.* These systems are specifically designed to produce a narrow set of product types, which are well known at the time when the system is designed and installed. These systems usually employ highly specialised machine tools designed to produce large quantities of a specific product or work-part.

Our focus is on ***general-purpose manufacturing systems,*** which represent an important new trend in manufacturing system design that provide a number of customer service possibilities.

3) From the viewpoint of the stability of the production environment and its effect on production planning and scheduling, two classes of manufacturing systems can be identified:

- *Manufacturing systems that operate in a predictable environment.* Because market demands for products are either reasonably constant over a long time period, or follow some form of periodic time curves, part types and their quantities can be foreseen. In such systems, customer orders arrive continuously, but current product types and quantities may vary in time. In these systems the Manufacturing Resources Planning (MRPII) approach can successfully address scheduling problems. The MRPII coordinates production on the basis of components' due dates, production capacity, etc. In this context, MRPII is seen as an effective scheduling technique.
- *Random manufacturing systems.* These systems produce products that arrive periodically at random time intervals. In these systems, customer orders cannot be predicted, fluctuation of material requirements is high, and the production environment is very changeable.

The focus of the system developed in this project is on ***random manufacturing systems***.

4) From a viewpoint of complexity of products that are produced, manufacturing systems can be divided in two classes:

- *Manufacturing systems that produce products requiring assembly of components.* Production is based on the manufacture of work-parts that are a part of subassemblies. Subassemblies are part of assemblies, which are in turn part of final products. In these systems, the scheduling problem is to deal with sets of jobs (parts) where each component of an assembly must be scheduled to ensure that all are completed at the same time. This avoids an inventory of parts waiting for other parts of an assembly.
- *Manufacturing systems that produce components of an individual or "one off" type.* The only scheduling problem is to deal with a number of different jobs. These systems are usually designed to produce a much larger variety of work-parts than the systems from the above group.

This thesis' primary interest is in the ***manufacturing systems that produce products of an individual or "one off" type.***

5) Finally, depending on the extent to which components of a manufacturing system are automated, computerised, and integrated, two extreme types of manufacturing systems can be distinguished:

- *Conventional manufacturing systems* that are characterised by a small degree of automation and computer control of material processing, material handling, and material storage operations.
- *Flexible manufacturing systems (FMSs),* which can be viewed as highly automated, Computer Integrated Manufacturing shops with some added constructs, such as central or distributed full computer control, automatic tool changers and tool magazines, sophisticated fixtures and pallets, common material handling system(s), and automated loading and unloading capabilities [Millar and Yang, 1996].

This research embraces ***both conventional and flexible manufacturing systems***, though the emphasis is on FMSs.

Following the above classifications of manufacturing systems and summaries of their preferences, we can now more precisely describe the manufacturing system of our interest. Therefore, the manufacturing system which is the subject of this research is distinguished by the following characteristics:

- It is a job shop, general purpose, random manufacturing system, which operates in an unstable production environment.
- The manufacturing system can produce work-parts in small batches and wide variety. Small batch sizes can even approach one item.
- The time of arrival of incoming orders in the manufacturing system cannot be foreseen. Orders of different sizes and product types arrive in the system in somewhat random intervals.
- Each particular job in the system might have a unique processing route. Jobs can go from one workstation to another in a somewhat random pattern.
- The operational time that is required at a particular workstation for performing manufacturing operations is highly variable (processing time for each operation is different). Also the time for a given work-part may vary from workstation to workstation (one workstation can process the same work-part faster or slower than other workstations).
- The system should provide a wide range of manufacturing processes. It needs to be highly flexible, so that it can cope with both the required volume changes – (can handle surges in volume), and product mix changes – (can handle considerable product variety).
- The need for rapid response should be highly emphasised.
- The system should be able to accept almost "any" customer order (in terms of feasibility to fabricate a work-part), and to produce it as soon as possible.
- The system should have the capability to route work-parts through the system by using alternative processing paths.
- The system should be able to efficiently operate under conditions in which short lead times and minimal inventory levels are present.

To establish competitive manufacturing systems of such characteristics in unstable manufacturing environments, two requirements are of particular importance:

- Manufacturing systems should have highly automated and flexible production facilities. They should employ state-of-the-art machinery that uses the latest achievements in material processing technology. This includes conventional (drilling, milling, boring, etc.) and unconventional (electro-erosive, laser, etc.) material cutting/shaping techniques. FMSs play an important role in meeting these requirements.
- Manufacturing systems need to have an effective production planning, scheduling and control system. For the successful application of FMSs, the latest processing technology on the shop floor should be employed. Additionally, the other supportive, accompanying production and logistic functions of the production system (product design, process planning, etc) has to be well integrated with the FMS. This is particularly true for the production planning and control (PPC) functions performed at the shop floor level.

More detailed discussion on manufacturing systems and on production planning, scheduling and control systems are provided in chapters 2 and 3 respectively.

1.5 The structure of thesis

A dissertation is structured around the following chapters:

Chapter 01 (this chapter) introduces the research work presented in this thesis. Initially, the chapter lists market demands and addresses some challenges imposed on contemporary manufacturing systems. The chapter introduces heterarchical manufacturing control systems, and explains why this type of control has not been widely presented in industry when this project started (1997). Also, it advocates the motives for conducting this research study, proposes the research work and defines its domain. The chapter finishes by highlighting characteristics of manufacturing systems to which this research project is primarily directed.

Chapter 02 covers some fundamental aspects of manufacturing. It encompasses a brief historical overview of manufacturing systems development, the basic principles of group technology, cellular manufacturing and flexible manufacturing systems. The chapter then highlights the importance of batch manufacturing and identifies some issues related to batch production. At the end, it describes some desirable characteristics of the proposed model of a manufacturing system.

Chapter 03 presents an overall framework of a production planning and control (PPC) system and points out the place and importance of a shop floor control (SFC) system in this framework. The most important approaches that have been traditionally used in production planning and control of manufacturing systems: Material Requirements Planning (MRP), Manufacturing Resources Planning (MRPII), and Just-In-Time (JIT) are covered in more detail. The chapter describes the basic concepts of SFC used for controlling manufacturing systems under MRP/MRPII and points out at complexity of such a system. At the end, the chapter highlights the drawbacks of traditional PPC systems when they are applied in random job shop manufacturing systems, lists desirable features of manufacturing control systems, and suggests the use of heterarchical control architectures for overcoming underlying SFC problems.

Chapter 04 centres around four topics: a scheduling framework, a review of some scheduling techniques, a review of a few simulation studies related to traditional scheduling techniques, and distributed scheduling. A section dedicated to a scheduling framework provides the key definitions of scheduling, highlights the complexity and importance of a scheduling problem, describes the main facets of the scheduling framework, and presents some criteria for judging

scheduling performance. The next section overviews the commonly used scheduling techniques and introduces briefly some relevant, non-traditional scheduling techniques that emerged in recent years as a result of efforts spent in finding effective tools to cope with complex scheduling problems. A section that reviews a few relevant studies regarding scheduling jobs in flexible manufacturing cells and flexible manufacturing systems is provided with the aim of identifying and extracting those methods which were reported to contribute significantly to the increase of the overall performance of manufacturing systems. The last section briefly introduces an agent-based scheduling approach that is covered in more detail in the next chapter.

Chapter 05 is dedicated to the new generation of manufacturing systems. Initially it addresses market trends and changes in manufacturing systems and briefly introduces concepts of virtual organisations. After that the chapter summarises the main requirements imposed on modern manufacturing systems and provides an overview of the most important manufacturing control architectures. Further, the chapter introduces non-traditional approaches and concepts in organising, designing and controlling manufacturing systems such as: multi agent, holonic, bionic, and fractal manufacturing systems. The chapter finishes by discussing similarities and differences between multiagent and holonic manufacturing systems and at the end it provides a summary of the main design principles for highly distributed manufacturing systems.

Chapter 06 is dedicated entirely to the review on research work conducted in the area of multi agent systems. A few agent definitions, a summary of common agent properties, and some agent classification schemas are provided at the beginning of the chapter. In addition, the following topics are covered in the chapter: commonly used coordination mechanisms and protocols among autonomous agents, the reasons for using agents in manufacturing systems, discussion on what entities in manufacturing systems should be mapped as agents, desirable requirements for successful implementation of agent-based shop floor scheduling and control systems, and multi-agent architectures in manufacturing control systems. The chapter finishes with a brief review of some tools and standards used for developing multi agent applications, and some concluding comments on multi-agent systems.

Chapter 07, after providing a retrospective discussion on relevant background material and discussion on a few points that influenced design of the proposed system of heterarchical shop floor control, defines the problem statement, the goals, and the objectives of the research work conducted in this project.

Chapter 08 describes the basic structure and operational principles of the adopted reference model of a job shop manufacturing system which was used as a basis for the development of a proposed distributed heterarchical shop floor control system. The chapter addresses relationships among system components and finishes by providing a list of the main characteristics of the manufacturing system model.

Chapter 09 is dedicated to the selection of a development platform used for creating a test-bed application, of the distributed heterarchical shop floor control system. The selection of computers, a computer operating system, a network system, network interface cards, and the selection of programming tools for the development of agent software are the main topics discussed in this chapter.

Chapter 10 presents the proposed architecture of distributed heterarchical shop floor control (SFC) system. The chapter describes the functional model of a production workstation agent, explains the co-operation and communication between workstation agents, and describes the methods used for scheduling jobs in the proposed model of a distributed heterarchical SFC system.

Using an example of the production of a simple work-part ***Chapter 11*** describes the way in which the experimental heterarchical SFC system operates.

Discussion of research results is presented in ***Chapter 12***, concluding comments are given in ***Chapter 13*** and possible directions for the future work are discussed in ***Chapter 14***.

The final part of a dissertation consists of the bibliography and two appendixes.

Appendix 01 discusses issues related to the Threading and Apartments in Visual Basic applications.

Appendix 02 is a copy of the letter received from the Massey University New Technology Developments Operations Group as part of an application for funding for further development. As only very limited funds were available the Board of Group was unable to assist at that time although they agreed that the system has very considerable potential. The letter provides evidence that the system was operational and demonstrated, running on a small Local Area Network, in 1999.

In addition to this thesis, further information about the program setup and its operation can be found on the accompanying ***Compact Disk (CD)***. The CD contains video clips about: a) Defining the program's data; b) Setting the program's parameters; and c) Video clips taken during the system operation.

Chapter 2. Overview of traditional manufacturing systems

The purpose of this chapter is to provide some background material that will contribute to better understanding the complexity of manufacturing and the accompanying issues, some of which will be addressed in this project.

This chapter deals with the structural/organisational aspects of manufacturing systems and represents introductory material to the next two chapters that provide further background related to the technological aspects (production planning, scheduling and control) in traditional job shop manufacturing systems.

The chapter includes a brief review of manufacturing systems development, explains the principles of group technology and the reasons that led to the appearance of cellular manufacturing systems, and points out, for instance, that a job shop manufacturing system can be a large system that consists of tens (in some cases even more than hundreds) of machine tools.

The chapter highlights the importance of batch manufacturing, identifies some issues related to batch production, and finally proposes a desirable structure for manufacturing systems to be used in the future. Later in Chapter 8, the principles embodied in this proposal will be used as guidelines for creating a model of manufacturing system that in turn is used as a basis for applying the principles of distributed heterarchical shop floor control and dynamic real-time scheduling in this study.

2.1 Defining the technological structure of a manufacturing system

As stated in Chapter 1, our interest in this research project is to develop a model of an efficient and affordable shop floor control system that is based on the principles of heterarchical control and multi agent systems. However, before proceeding with this task we need first to decide and adopt the technological structure of the manufacturing system upon which the model will be based.

Generally speaking, the selection of an appropriate system structure is a very important task. It is important to define what is meant by structure. The Oxford English Dictionary defines structure as *"the mutual relation of the constituent parts or elements of a whole as determining its peculiar nature or character."* That is, structure is concerned with how the individual components of a system relate and how the nature of these relationships determines the overall system behaviour. From the organisational perspective of the system a structure can be viewed as *"the manner in which the organisation divides up its labour into specific tasks and achieves co-ordination among these tasks"* [Johns, 1992]. The term ***technological structure*** refers to the manufacturing system's "hardware". Technological structure is represented by physical components (capital assets) of the system, including operators that operate this hardware. In contrast, the term ***methodological structure*** refers to the manufacturing system's "software". Methodological structure is represented by approaches, methods, and procedures used for

managing a manufacturing facility to produce the desired output (some of these approaches are discussed in Chapter 3).

The selected structure will influence, for example, the ability of a system to effectively and efficiently produce products, the way in which resources are used, and the ability of the overall system to cope with variability and disturbances. Also, while defining an appropriate structure, it is necessary to consider a whole set of many other attributes regarding relationships among constituent elements of the system (for example, the interaction of the material transport system with the production units inside the manufacturing system, etc.).

To facilitate this task (of defining an appropriate system structure), we review first what has happened in the past. The next section provides a brief historical overview on the development of traditional manufacturing systems.

2.1.1 A brief review of development of manufacturing systems with emphasis on job shop production

At the beginning of 20th century, four types of manufacturing system were distinguished: job shop, flow shop, project shop, and continuous process.

Job shop manufacturing systems had a functional lay out and were characterised by the ability to produce a wide variety of products in small manufacturing lot sizes. However, these systems were very inefficient because the production process closely followed customer orders and due to frequent machine preparation (changing tools, jigs and fixtures) for different work-parts (customer orders) a considerable amount of time was wasted on non-added value activates.

During and after the two World Wars abnormal markets were created as a consequence of the productive effort involved in peacetime reconstruction and economic recovery. Almost any type of product was required and could be sold in the huge quantities. At that point of time, a competitive focus among manufacturers shifted from ***industrialisation*** (which was characteristic for the 19th centaury) to ***productivity*** (getting more output per input) and ***efficiency*** (increasing resources utilisation). The main goal was to produce as many products as possible for a given time horizon by having highly efficient systems.

To increase productivity, the job shop systems evolved into ***production job shop systems***. These systems continued to fabricate larger number of versatile products (like job shops) but started to produce them in batches. In this way the number of set-ups on many of the machines for each new order for diverse products were reduced. Although these systems continued to use the same type of equipment (general purpose), ***machine utilisation*** was improved, as only one set-up was required per batch. Because the production rate usually exceeded the customer demand rate, the job shop started to build an inventory of the products, before switching to other orders. Under the persistent pressure for increasing machine utilisation, the production batch sizes soon started to become larger (comprising several thousands of items instead of initially several hundreds). In other words, the aim was to further reduce the number of batch changeover-procedures and, hence, avoid the long set-up times. In addition, larger batch sizes contributed to reduction of production costs per unit since the cost of lost production time (required for machine preparation to produce new product) was spread over a larger number of units. However, the production job shop became extremely difficult to manage as it grew, resulting in long product throughput times and very large work-in-process inventory levels. With high inventory, production was solely dependent upon labour and machine utilisation. Therefore, high productivity in such systems was maintained by providing machines and labour with a backlog of work. High inventories provide factories with efficient use of machines and people. Both were always busy producing parts. However, *factories became efficient producers of parts but inefficient producers of*

products [Lenz, 1994] due to higher inherent production costs. Inventory became the "culprit" for the high cost of production.

During the seventies, increasing interest was paid to the ***reduction of production cost*** per unit. During that time focus was directed to methods for ***reducing inventories***. Hundreds of techniques were developed during this period. The principles of ***group technologies*** and concept of ***cellular manufacturing*** received a significant emphasis during this period. They were generally recognised as being useful in a batch-manufacturing environment, especially if there was a large number of small batches.

Group technology is based on the principle that similar things (including product design, process planning, fabrication, and assembly) should be done together. In the group technology approach, machines and people that are required to produce a family of parts with similar processing requirements are grouped together in one or more dedicated production areas, referred to as manufacturing cells. This is a more efficient organisation compared with process layout, and generates benefits in production, process planning and control, materials handling, and employee satisfaction. Factories based upon cell layouts have experienced a number of improvements. For example, more effective production control can be applied (scheduling procedures are greatly simplified since the number of units is reduced), material handling costs are lower (material handling is dramatically reduced because most of work-parts flow through a single cell), work-in-process inventory is reduced (the material flow is simplified), the number of set-ups required is reduced (because similar parts are grouped), response to changing conditions is quicker (set-up times are also reduced), quality of products is improved (quality is controlled within the cell), productivity is increased (throughput time is shortened), the reliability of equipment is increased (equipment within the cell is routinely maintained by the workers), and job satisfaction and status are improved. Guun [Guun, 1987] argued that due to similarity between the manufacturing and design characteristics of the range of products in a family, the following percentage reduction can be achieved (results from a survey of 35 cell based manufacturing systems): average WIP 43%, set-up time 17%, manufacturing lead time 55%, average batch travel distance 79% and on-time deliveries increase 61%. By applying principles of group technology the main causes of high inventory levels are reduced and factories focus more on producing major components and products and less on producing parts.

Development of numerical control (NC) machine tools and robots has significantly contributed to further reduction of set-up times. Programmable machine tools and robots allowed the possibility of manufacturing different parts on the same equipment without substantial set-up interruption. The integration of Computer Numerical Controlled (CNC) machines and robots into flexible manufacturing systems was the next logical step in the further development of manufacturing systems.

Traditionally, production facilities have two conflicting objectives: flexibility and productivity. *Flexibility* refers to producing a number of distinct products while *productivity* relates to higher speed production[1] or increased output for a given input of time or per person. Therefore, ***increasing job shop productivity while maintaining production flexibility*** has been goal of the manufacturing industries. The emergence of ***flexible manufacturing systems (FMSs)*** in the early seventies was a development in this direction.

---

[1] The term productivity means more than "high speed production". It is a measure of the ability to create goods and services from a given amount of resources (labour, capital, materials, land, knowledge, time, or any combination of those). In manufacturing, productivity refers to the ratio of the output quantity of any production process divided by the unit of input quantity. Most often the term productivity refers to the ratio of the quantity of units produced per unit of time.

FMSs typically consist of a number of processing workstations (CNC machines, robots and in-process storage facilities) that are interconnected by means of an automated material handling system (AGV and robots), which are all integrated via computer networks and operated under an overall computer control architecture. FMSs are capable of producing a variety of products or work-parts with minimal lost time for changeovers from one product to the next. (With FMS, the idea was to combine the high efficiency, productivity and quality of high-volume mass production with the programmable flexibility of the CNC machines, robots and automated guided vehicles (AGV) in order to be able to produce a variety of work-parts effectively and efficiently). As demand for different products change, the system can be readily rescheduled without significant disruptions and delay. In FMSs productivity is increased - production rate and output per labour hour are greater. Quality is improved through greater consistency in processing. Flexibility to changeover quickly from one product to the next and flexibility to accommodate new products is increased (comparing with the flow shop systems). Unit costs of products are reduced because the production rate is increased and the labour cost is reduced. By employing CNC machines the set-up times are reduced because the number of different types of machines needed to produce a part is reduced and the changeover of fixtures and tools is performed faster.

The main distinguishing feature of an FMS from traditional manufacturing systems is "flexibility". However, the term “flexibility” does not have a precise definition and has many specific meanings in manufacturing. Sethi and Sethi identified about 50 terms relating to 11 flexibility types [Sethi & Sethi, 1990]. Some of those types such as routing flexibility, machine flexibility, operational flexibility, etc. are well manifested in FMS. A summary of various types of flexibility that may be utilised in increasing the performance of flexible manufacturing systems include:

- ***Machine flexibility*** (of a machine) refers to the various types of operations that the machine can perform without requiring a prohibitive effort in switching from one operation to another.
- ***Material handling flexibility*** is the ability to move different work-part types efficiently for proper positioning and processing through the manufacturing facility it serves.
- ***Operational flexibility*** of a work-part refers to its ability to be produced in different ways, commonly by changing the sequence of operations.
- ***Process flexibility*** of a manufacturing system relates to the set of work-part types that the system can produce without many major set-ups.
- ***Product flexibility*** refers to the situation in which new part types can be added or substituted for existing part types.
- ***Routing flexibility*** of a manufacturing system is the ability to produce a work-part by alternate routes through the system.
- ***Volume flexibility*** of a manufacturing system is its ability to be operated profitably at different overall output levels.
- ***Expansion flexibility*** of a manufacturing system is the ease with which its capacity and capability can be increased when needed.
- ***Program flexibility*** is the ability of the system to run virtually untended for a long period while processing a variable range of parts without operator intervention.
- ***Production flexibility*** is the universe of part types that the manufacturing system can produce without adding major capital equipment.

- ***Market flexibility*** refers to the case where the manufacturing system can adapt to a changing market environment.

Depending on the relationship between the *number of different products (P)* and the *production quantity (Q)*, the spatial structure of a flexible manufacturing system can take one of the following layout patterns [Hitomi, 1994]: 1) Process (Functional) Layout - in the case of small Q/P ratio, 2) Product (Flow-line or Production-Line) Layout - in the case of a large Q/P ratio, and 3) Cellular (Group Technology) Layout - in the case of an average Q/P ratio.

Despite the fact that the concept of FMS was well accepted by industry, the number of successfully implemented FMS was very low. There are many reasons that explain such a state but the most important is that these systems require very high investments, often with disappointing performance.

In the late 1970s, the idea of starting with small, easy to control ***flexible manufacturing cells (FMCs)*** (obeying principles of group technology) as the first phase of implementation of FMSs was regarded as a feasible approach. It was envisaged that once the reliability of the cells had been proven, linking FMCs into FMSs by material and information flow, integration could be economically justified. Such a concept has also been more acceptable and applicable for small companies with limited investment funds. In the beginning of 1980s, flexible manufacturing and assembly cells, defined as the *"sum of all devices required for automatically producing a family of different parts or components on one physical capacity unit"* [Spur et al., 1986] emerged as smaller, less expensive and more tractable installations. The difference between a FMS and FMC is sometimes defined in terms of the number of workstations included. Usually a FMS includes four or more machines, while a FMC contains three or fewer. Of course, this is not a universally accepted dividing line and other divisions are possible.

However, the approach of building manufacturing systems each consisting of several FMCs has its disadvantages. The main issue addressed in cell manufacturing systems is ***restricted system flexibility***. Because of the manufacturing cells' design and resulting organisational structure, a cellular manufacturing system is not as flexible as a functional arrangement in that the cells are design to fabricate narrower set of product types. In addition, it may be difficult to route different parts in different ways through the system, for example from one cell to another and then back to the first cell. This in turn can restrict the volume of parts that can be handled by the system at one time. Another issue is ***machine utilisation***. With lower inventories inside cells, efficient use of tools and machines is usually reduced as well. In addition, machine utilisation among cells can be uneven. Machines may be overloaded in one cell and idle in another. The next disadvantage is related to the ***overall system optimisation***. Both, experience and theory suggest that optimisation of individual cells will only lead to a sub optimal performance of the entire system. Finally, the entire manufacturing system is ***difficult to integrate*** so that flexible manufacturing cells often remain islands of expensive automation.

2.1.2 Importance of batch production

In spite of the superiority of "flow line manufacturing systems" (high productivity and efficiency), due to market demands for variety in products, systems organised into a process layout and batch production have been dominant for many years. It was estimated that, at the end of eighties, over 70%-90% of all engineering manufacture is carried out on a batch basis [Weatherall, 1988, pp36] and that almost 90% of all items produced in discrete part manufacturing are made in batches of less than 50 units, [Vakharia & Selim, 1994]). Batch manufacturing has, therefore, become the focus of the work of many academic and industrial research institutions with the aim of tackling its inherent problems by applying the latest achievements in manufacturing technology, computer science, and control techniques.

2.1.2.1 Some advantages and disadvantages of job shop systems and batch production

As pointed in Section 2.1.1 "A brief review of development of manufacturing systems with emphasis on job shop production", batch production originally was carried out in manufacturing facilities with a functional layout (job shop manufacturing systems). The advantage of such an organisation is that a wide range of different products with different process requirements can be supported simply by changing the routing through the plant. These manufacturing facilities have significant system flexibility in terms of product flexibility, volume flexibility, routing flexibility etc.). The major disadvantages include time-consuming material handling operations, long set-up times, high levels of work-in-process inventories, and the difficulty of reconciling different scheduling requirements of different products on various sections.

In batch production, at any given time only part of the batch is being processed while the remainder lies idle. Also no work is done on work-parts as they are transported between machines. It has been estimated that in many traditional batch production jobs, parts spend less than 5% of their time in the factory being processed [Weatherall, 1988, pp.36] and the rest of the time is spent waiting or being moved from one functional area to the next. Further, once the part is on the machine, it is actually being processed only about 30-40% of the time [DeGamo, et al., 1988, pp.9]. The rest of the time is spent adjusting the settings of the machine, loading and unloading of parts, tool changing, inspection, and so on. Consequently, work in progress costs are high, order delivery lead times are often extended, and material control problems are complex.

2.1.2.2 Attempts to overcome issues related to batch production

To cope with the inherent problems of batch production, cell manufacturing and flexible manufacturing systems emerged.

FMSs, were generally recognised as being useful in batch production, especially if there is a large number of small batches. However, FMSs were engineered to switch rapidly but only among predefined members of a family of products. The complexity of FMSs (which are fundamentally job shop systems, though some of them are organised on the principles of cell manufacturing) made control problems even worse. Reliability of these systems was impaired, and the required throughputs were often not achieved due to the lack of adequate means of production planning and scheduling. To summarise, there were undoubtedly successful implementations of FMSs (for example, some Japanese companies that were engaged in the production of automobiles, machine tools, and robots using FMSs). However, in spite of the expenditure of considerable amounts of time and money, in overall terms, the performance of FMSs was far below expectations. FMSs were poorly specified and mostly perceived as technological solutions that involved only the integration of appropriate hardware and software without the understanding of the essential and desirable prerequisites for their effective implementation. Grossly inadequate effort was put into planning and controlling FMSs. For that reason, the implemented FMSs were even dismantled in many companies [Kochhar, 1997].

Flexible manufacturing cells (FMCs) emerged as more manageable, reliable and less expensive installations. However, cellular manufacturing and particularly the concept of linked FMCs also has some drawbacks. As discussed earlier, some of the key disadvantages of FMCs are as follows: restricted system flexibility, uneven machine utilisation, it is very hard to optimise and integrate an entire manufacturing system, and the efficiency of sharing system resources is restricted.

2.1.3 A desirable structure for a model of manufacturing system

From the material presented in this section (and the relevant literature), it is possible to extract and define some guidelines for the creation of a desirable structure for future manufacturing systems. There are two most distinctive elements on which decision has to be made: ***type*** and ***size*** of the manufacturing system structure.

At its simplest, the choice is between a job shop and a flow line structure. Keeping on mind 1) customer demands (discussed in more detail in Chapter 1), and 2) harsh requirements imposed nowadays by the market on contemporary manufacturing systems (namely, the system should be able to cope with circumstances in which the products need to be manufactured in small quantities but with many varieties and in which the manufacturing orders, of unpredictable types and sizes, arrive in the system randomly with short notice), it seems that ***the structure of the manufacturing system should follow that of the job shop type***.

Another question is related to the size of the manufacturing system. This aspect is considered in the following text. Beside the reduction of set-up times, the routing flexibility is viewed as another important element for reduction flow times (increasing system efficiency). Namely, flow time can be reduced by providing multiple alternative processing paths for work-parts to follow during the production process. All the paths cannot be online continuously (they require set-up) but the existence of multiple paths might allow work-parts to take less time to complete their process - less flow time. For example, if one of the machines in the main processing path becomes overloaded or non-operational, then the work-part can take alternative processing route. The completion time will not depend upon the availability of one path and in some cases might be shorter than the average time when using a single path. However, with this type of system, flexibility (routing flexibility) in small flexible manufacturing systems (FMS) and flexible manufacturing cells (FMC) is severely limited. This is due to the small number of machine tools (in the cells usually less than five) and there is a very small number of alternative processing routes (if they exist at all). On the other hand a functional layout found in the classical job shop manufacturing systems provides many more processing paths through the manufacturing facility. Functional (job shop) layouts are insensitive to changes in product mix and product design and as a result these changes do not generally cause major disruptions. Similarly, changes in volume can be accommodated more smoothly with more even machine utilisation distribution.

The above discussion suggests that having cells with a higher number of machines would be an advantage. Namely, to provide multiple potential processing paths in the cell, the cell must be equipped with a few alternative machines capable of performing the same processing operations (machine flexibility) and must be operated by workers that are capable of working on multiple machines (labour flexibility). Therefore, a compromise solution can be seen by ***extending the size of the manufacturing cells*** with an increased number of machine tools (This is contradictory to the reasons why manufacturing cells were created in the first place) arranged inside the cells on the basis of a functional layout. In this concept, the benefits from globally arranged material flows as in classical cell manufacturing (product layout) – on a macro level - could be combined with the benefits of the manufacturing systems with the functional layout (increased production flexibility and more even sharing of resources) – at the micro level. This approach would lead to transformation of manufacturing cells into larger production facilities capable of producing a number of product families. Furthermore, product families could be more loosely established (widening criteria for creating product families) with a higher number of parts within a product family and greater variability of products in them. FMCs could consist of several tens of CNC machines so that FMCs of such a size would more or less correspond in production capacity to the traditional machine shops comprising hundreds of conventional

standalone machines. The need for moving parts between cells would be reduced and thus the problem of interrelationships among cells would be minimised or in some cases totally eliminated. Integration of more FMCs into one system would be easier to accomplish because: 1) the number of cells in the system would be significantly reduced (for example 2-3 cells for production parts of prismatic, disk, and shaft types), and 2) this integration would take place on a much higher, abstract level. Each cell could operate in more autonomous manner with a great deal of managerial and control issues solved at the cell level.

However, for successful operation of such large manufacturing facilities (consisting of several tens of CNC machines) ***it is necessary to have an efficient manufacturing control architecture, particularly in the case of shop floor control systems.*** The absence of appropriate control architecture has already been regarded as one of the main causes for the failure of previously implemented FMSs. Indeed, *"it has been repeatedly documented in the literature that information systems are the primary source of problems in an FMS implementation"* [Gowan & Mathieu, 1993]. Manufacturing control architectures are discussed in Chapter 5. The model of manufacturing control system advocated in this thesis provides a framework for development of such a control system that would be capable to control manufacturing facilities that might consist of a very large number of machine tools.

2.1.3.1 Characteristics of the adopted model of manufacturing system

The main features that characterise the proposed model are:

- A highly efficient and flexible structure, low-volume job shop manufacturing system that is assumed to be comprised of sophisticated ***general-purpose CNC machine tools***. (Sophisticated CNC machines are typically equipped with the large tool magazines and automatic tool changing systems. Because tool movements are programmed, and the machines can automatically change tools and load or unload work-parts, the set-up and processing times can be significantly reduced when compared with conventional machines).
- Material handling operations between workstations are performed by ***general-purpose material handling equipment***, such as forklift trucks or automated guide vehicle (AGV) systems. Operators or industrial robots are used for material handling operations over shorter distances (loading and unloading parts from machine tools and pallets).
- Manufacturing systems of this type are usually ***equipped with automated storage and retrieval systems*** for rapid and accurate storing of raw materials, work-parts, tools, fixtures, and finished products. Their main advantage is control over the inventory stored in them. Work-parts in process are usually stacked on pallets.
- ***Staff are assumed to be versatile and highly skilled*** so that they can perform a range of different work assignments. Cross training and other investments in personnel development are of importance. With improved training operational staff can be given more flexibility in decisions relating to task timing and sequences at sophisticated workstations when compared with the situation in a highly planned and rigidly controlled system using low skilled, direct labour for component production as in car assembly lines of the past.
- The systems should provide a high degree of ***equipment and labour alternatives***. By having several alternatives in equipment and people, changes in schedules and design are much easier to handle. For example, the provision of routing flexibility (alternative paths) in process plans can increase overall manufacturing performance particularly in terms of capacity utilisation. In a rapidly changing manufacturing environment

responsiveness to fickle customer demands requires both equipment and labour capacity to be available to handle surges.

Chapter 8 will discuss some aspects of a demonstration implementation of a practical heterarchically controlled manufacturing system in which the above characteristics are considered as highly desirable prerequisites.

Chapter 3. Production Planning and Control system – traditional approaches

Since the focus of this research is on distributed heterarchical (non-conventional) shop floor control (SFC) systems, the intention of this chapter is to point out the place and role that SFC systems have in the context of the much wider range of systems used for Production Planning and Control (PPC) of manufacturing systems. Also, to investigate the possibilities of integration of the proposed heterarchical SFC system with the other parts of a PPC system, it is important to understand 1) the inherent relationships that exist between SFC and other subsystems of PPC and 2) elementary theory on which traditional methodologies used in classical SFC systems are based. Therefore, the purpose of this chapter is to present the overall framework of PPC systems, to describe the most important approaches in PPC systems that have been traditionally used for controlling manufacturing systems (in particular, Materials Requirements Planning (MRP), Manufacturing Resources Planning (MRPII) and Just-In-Time (JIT)), and to describe basic functions and the most important techniques used in the traditional SFC systems.

The chapter is structured around the following main topics:

The need for a effective shop floor management system and the purpose of this chapter are emphasised in the introductory part.

Section 3.1 gives some definitions and a simplified framework of modern production planning and control (PPC) systems.

Section 3.2 stresses the role and importance of PPC systems used in the development of modern, competitive manufacturing companies.

Well-known approaches that are used in building large-scale systems: Materials Requirements Planning (MRP), Manufacturing Resource Planning (MRPII), and Just-In-Time (JIT) are discussed in more detail in Section 3.3.

A summary of conventional approaches (MRP and JIT) in the design of Production Planning and Control systems is given in Section 3.4.

Section 3.5 describes basic concepts underlying shop-floor control techniques under MRP/MRPII approach and points out at complexity of such a system.

Section 3.6 provides a list of desirable features of manufacturing control systems that are applicable to low-volume job shop manufacturing systems.

Finally, Section 3.7 highlights the drawbacks of traditional PPC systems when they are applied in random manufacturing systems and suggests the use of distributed control architectures for overcoming underlying problems.

3.1 General framework for a Production Planning and Control system

A Production Planning and Control (PPC) system, as its name suggests, is concerned with the planning and controlling of manufacturing processes, including materials, machines, people, and suppliers. Thus, PPC involves organising and managing the process of converting raw materials into pre-designed finished products. The goal of PPC is to achieve the best possible correspondence between the external market demands of a company (customer demands and/or marketing sales plans) and internal value-adding possibilities of the company. It is believed that an effective PPC system provides a substantial competitive advantage for a company.

There is no standard definition of production planning and control and the operational meaning usually depends on the particular application. However, as an example, a generic definition given by Burbidge, and by Vollmann et al. will be used as the basis for understanding production planning and control.

"Production control is the function of management which plans, directs, and controls the materials supply and processing activities of an enterprise. Where, planning is the process of deciding what to do in the future, directing comprises the operation of issuing orders, and control can be described as the constraining events to follow plans" [Burbidge, 1978, 1989].

"Basically the Manufacturing Planning and Control (MPC) system provides information to efficiently manage the flow of materials, effectively utilise people and equipment, coordinate internal activities with those of suppliers, and communicate with customers about market requirements." [Vollmann et al., 1992]. Also, the authors noted, *"A key in this definition is managers' need to use the information to make intelligent decisions. The MPC system does not make decisions or manage the operations - managers perform those activities. The system provides the support for them to do so wisely."*

Clearly, production control involves making certain decisions toward meeting a certain objective, implying that it is primarily a decision-making activity.

One of the earliest integrated views of the whole process of production planning and control system was presented by Holstein whose graphical definition of production planning and control system is depicted Figure 3-1 [Holstein, 1968]. The definition identifies eight classes of activities (eight boxes in the figure) as well as the major information flows that connect these activities. The majority of current production control frameworks and flow charts are variations of these basic functions.

Another simplified framework of modern PPC system is given in Figure 3-2 [Vollmann et al., 1992]. It is the skeletal framework while the full system depends on the company's specific needs. The full system is characterised with different degrees of detail in different areas and includes data inputs, other system modules, and feedback connections.

According to Vollmann et al., Manufacturing Planning and Control systems in any firm encompass three distinct phases. Figure 3-2 depicts these three parts.

- The first phase, or front-end, is the ***creation of the overall manufacturing plan.*** General directions for Manufacturing Planning and Control are provided by ***a company game plan.*** In this plan overall company objectives are established. The game plan, which is top management's responsibility, should always be consistent with strategic plans, departmental budgets, and the firm's capabilities. As a result of activities at this phase, as an integral part of the game plan, ***a manufacturing plan*** is generated. A manufacturing plan specifies the production output required to achieve the overall

company objectives. A manufacturing plan must be stated in production terms, such as end items or product options. Finally, the front end of the MPC system produces the ***Master Production Schedule (MPS)***.

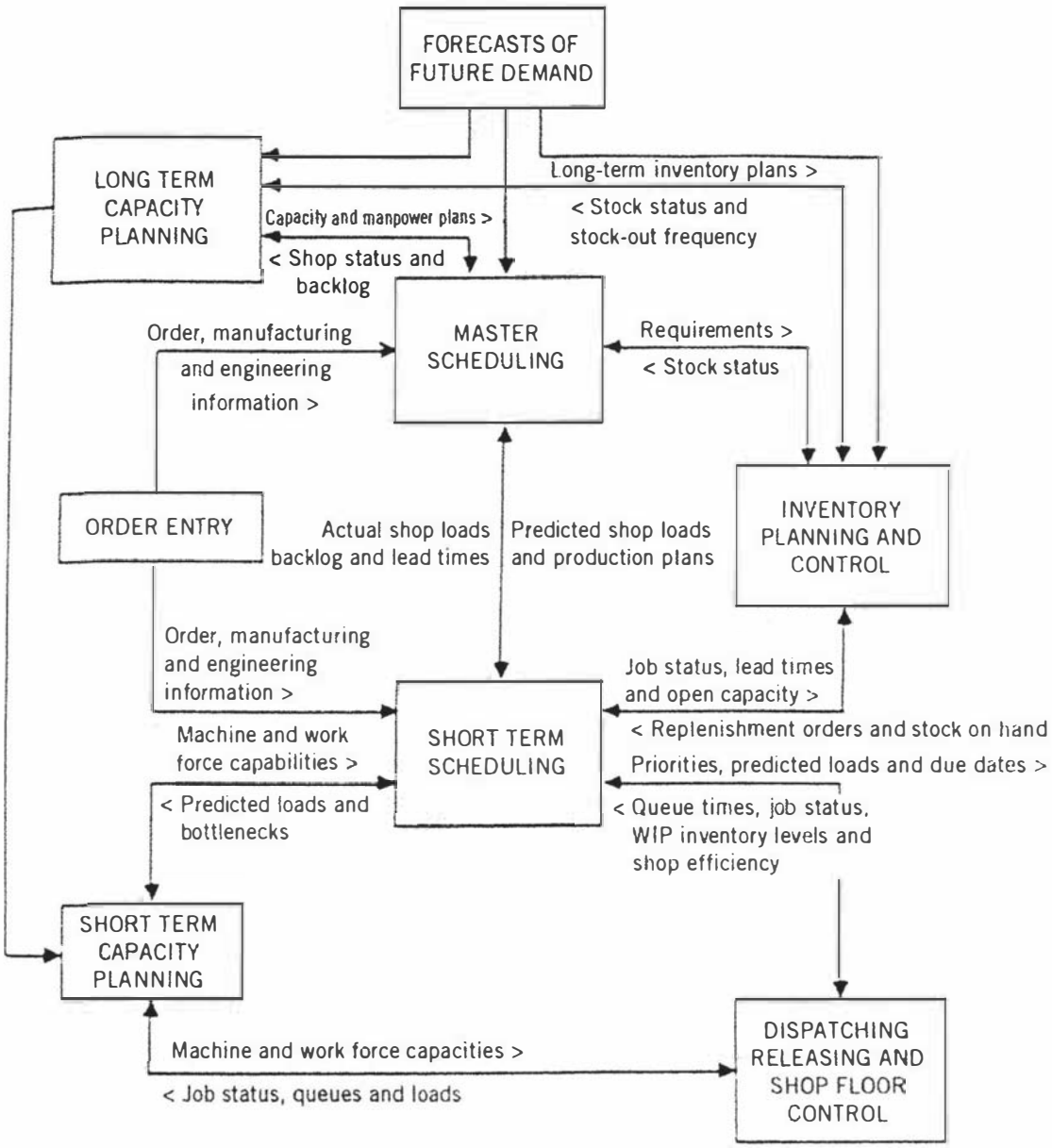


Figure 3-1. Graphical definition of a Production Planning and Control (PPC) system

- The second phase, called the engine, involves ***performing the detailed planning of material and capacity needs to support the overall plans.*** The master production schedule feeds directly into the detailed material planning module. For firms producing a wide variety of products with many parts per product, detailed material planning can involve calculating requirements for thousands of parts and components, using a formal logic called Materials Requirements Planning (MRP). MRP determines (expands) the period-by-period (time-phased) plans for all component parts and raw materials required to produce all the products in the MPS. This material plan can thereafter be utilised in the detailed capacity planning systems to compute labour or machine capacity required to manufacture all the component parts.

- The final PPC phase, or back-end, involves ***executing the detailed material and capacities plans on the shop floor and in purchasing.*** Therefore, the back end, or execution system, deals with shop floor scheduling of the factory and with managing materials coming from vendor plants. The configuration of shop floor systems depends on the process's needs. For example, firms with process layout will have shop-floor control systems that establish priorities for all shop orders at each process department so the orders can be properly scheduled. Other firms with production cells layout will probably find that the Just-In-Time based shop-floor systems are a more appropriate solution. Purchasing systems provide detailed planning information for vendor scheduling. In essence, purchasing is the procurement of outside machine capacity. It must be planned and scheduled well to minimise the final customers' overall cost.

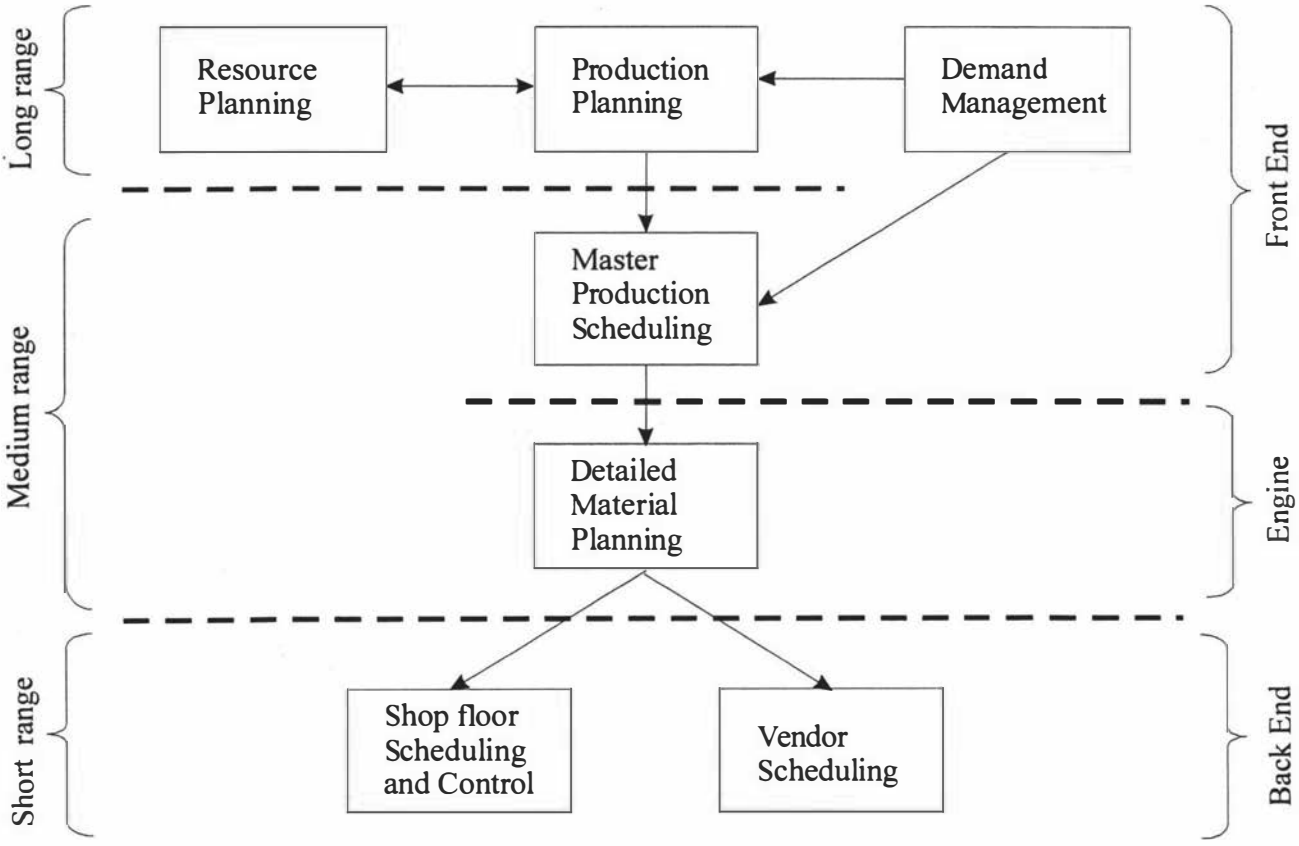


Figure 3-2. Simplified framework of a Production Planning and Control (PPC) system

The PPC scheme shown in Figure 3-2 presents a set of functions that must be accomplished in every manufacturing company, whether small or large. That is, there must be front-end, engine, and back-end activities.

Though the scheme does not capture the emphasis or importance that each module has in a particular company, nor the different systems or techniques that are used, it can be viewed as the standard for the general functions to be performed and for evaluating and comparing alternative systems. The scheme can be used as a checklist for auditing a PPC system. The crucial questions are whether each activity is performed, how well they are integrated, and how well each subsystem works.

There is another view on production planning and control system which is supported by many authorities. They distinguish between long, medium, and short-range PPC horizons that are also indicated in Figure 3-2. These three levels range from large aggregations of planning for long

time periods to very detailed machine scheduling for time intervals of an hour or less. This is a useful view, but the time dimension inside the same range varies substantially from company to company depending on the firm's specific needs (for example, in the long range, this time varies from a few months to several years in the future). Three levels of hierarchy are:

- ***Long term plan*** (corporate game plan) considers the overall manufacturing concerns such as the development of new products, changes in the work force, procurement of new plant, etc. with a planning horizon measured in years (0-5).
- ***Medium-term plan*** (master production scheduling - MPS) considers the output from the long term plan, as well as the expected demand, inventory and capacity levels, to produce a viable work load in terms of end-product production. This is often in weekly time periods over the one year. For assembly type industries it is desirable that the output from MPS is placed into a Materials Requirement Planning (MRP) system. The weekly workload is then broken down further into requirements for sub-assemblies, parts, and raw material.
- ***Short-term plan*** (detailed or job shop scheduling) takes the output from the MRP system, or the output from the MPS for non-assembly type industries, and then performs the capacity loading and sequencing of each job on each machine or process.

Finally, we will conclude this section by providing yet another view to production planning and control system. Namely, in a hierarchy of conventional production planning and control architecture the following five levels of planning, scheduling and control functions can be distinguished: production planning, process planning, scheduling, shop floor control, machine control, and device control [McFarlane & Bussmann, 2000]. A brief description of each is given in the following sequel.

- **Production planning** identifies products required to fulfil an order, defines parts and subassemblies, together with the required numbers of each item, that are required for products to be manufactured and defines the number and size of production batches. Production planning also governs decisions such as what product orders should be manufactured, when new jobs should be started (order release times), what level of inventory should be carried, and when machine maintenance should be performed. Sometimes the production planning process involves activities that belong to the process planning function. The production planning approach in that case results in production batches with defined production tasks and sequence.
- **Process planning** defines production tasks/operations and their sequence (production routes) for work-parts being processed in the factory. During a process planning process, firstly an order is decomposed into a sequence of production tasks/operations, and secondly, the nominal allocation of operations to resource types is determined. Note that during this process specific resources or production times are do not determined.
- **Scheduling** in a discrete manufacturing environment involves the allocation of production operations to specific resources and the specification /determination of the timing (start, duration, completion) for those operations. In scheduling the batches are assigned to specific resources (machines) and the required work-hours and the batch completion time are determined. Detailed schedules contain start and completion dates for all operations to show when these must be done if the manufacturing orders are to be completed on time.
- ***Shop Floor Control / execution*** represents that part of manufacturing control where theoretical expectations and physical production realities meet. Shop floor control involves: 1) the initiation of tasks (production, transport, etc) involving actual start times

and actual production settings; 2) the control of the execution of the tasks; 3) the monitoring of task status; 4) the termination of the tasks.

- ***Machine control*** involves the initiation, co-ordination, and monitoring of the different machine functions or devices required to support the execution of production tasks by an individual machine (e.g. control of a NC machine or multi axis robot).
- ***Device control*** involves actuation, sensing, and feedback control of the physical operations which support a machine or process unit. (e.g. control of a pump or servo motor).

One of the key issues in modern manufacturing control systems is to provide a high degree of flexibility and interoperability between these levels. A model of shop floor control system discussed in this thesis partially addresses this issue, in particular, linkages among process planning, scheduling and shop floor control levels.

3.2 The importance of a Production Planning and Control system

Nowadays, Production Planning and Control (PPC) is seen as the main subsystem in developing modern manufacturing facilities, to provide a substantial competitive advantage for a company in its markets. At the time when the application of the latest technologies and highly automated and flexible structures become common in factories, the company with the more successful production planning and control system will gain the lead in the market.

The main aims of contemporary production planning and control systems are to: 1) reduce lead times, 2) keep low levels of work-in-process (WIP) inventory, 3) achieve high system usage, and to 4) provide greater production flexibility. Reduction of manufacturing lead-time (from a few weeks to a "few" hours) is a must for a company to be competitive. Achieving very short lead times supports better customer service and responsiveness. The lead-time (time needed for part production) is made up of processing and queue time. By engaging automated equipment the processing time is considerably reduced compared with manually operated machinery. To reduce queuing time (and production costs), Work-In-Process (WIP) inventory levels must be kept low. The shorter processing time and significantly reduced levels of WIP require the production control system to provide careful monitoring and better co-ordination of all production activities. Also, due to high capital investments in automated machinery, a high rate of machine utilisation is required. For highly automated manufacturing facilities an average usage of machinery across the whole system is often aimed at 80% or more. It means that system usage becomes an increasingly important factor that puts an additional burden on production control systems. Achieving flexibility in all aspects of production operations is one of the primary goals of the future. The capacity of a system to achieve this flexibility to a large degree depends on the production planning and control system. Namely, the flexibility of the system is created in two phases. *"Firstly, the system is designed to be flexible and then it is managed to express that flexibility"* [Vollmann et al., 1992]. Installed production equipment makes it possible to achieve the first phase. Achieving the second phase, specifically process flexibility, operational flexibility, routing flexibility, and volume flexibility (refer to Section 2.1.1. "A brief review of development of manufacturing systems with emphasis on job shop production" for explanations on these different types of flexibility), will depend primarily on the capability of the production control system.

In highly automated manufacturing systems the control decisions can be made with more certainty about the actual state than in the conventional systems. The following factors provide

better conditions for PPC: 1) *the availability of data* (the presence of a large amount of data which are automatically generated); 2) *the quality of data* (data are more accurate, since they are produced automatically); 3) *the immediacy of data* (there is little elapsed time between data being generated and being received. A FMS contains some provision for communication within the system).

Despite these better conditions highly automated systems require even more complex problems to be solved. For example PPC systems need to: 1) *Provide more comprehensive on-line control.* The speed at which jobs move from operation to operation is much higher than in conventional production systems. 2) *Consider more engineering details.* To achieve low Work-In-Progress and short lead times, tooling and jig/fixture requirements may have to be included in the PPS. 3) *Integrate with existing software systems.* Production Planning and Control software should link in readily with the organisation's existing software packages such as Materials Requirements Planning (MRP), Computer Aided Design (CAD) and Computer Aided Process Planning (CAPP). 4) *Generate detailed instructions.* Automated manufacturing systems require detailed instructions about machining and sequencing operations.

The importance of Production Planning and Control systems was recognised in the 1970's (the first integrated framework to PPC, Materials Requirements Planning – MRP system, was established in 1975 [Orlicky, 1975]) and significantly increased during the 1980s (the extended version of MRP, Manufacturing Resource Planning – MRPII, was created in 1984 [Wright, 1984]). For example, Mather argued that the lack of an effective control system would impede progress toward the "factory of the future"[1] [Mather, 1986], while both Hayes and Wheelwright [Hayes & Wheelright, 1986] and Skinner [Skinner, 1985] described Production Planning and Control as critical infra-structural elements for the development of a competitive manufacturing strategy. However, till recently the "right overall system solution" for contemporary manufacturing systems has not been found. As it will be discussed later in this thesis, the concepts of distributed manufacturing systems have recently set new directions for system designs and implementations.

3.3 Integrated - traditional - approaches to Production Planning and Control

Traditionally, production control modelling has focused on one or two of the functions identified by the general framework given in Section 3.1 (refer to Figure 3-1 and Figure 3-2). In most cases these functions are executed separately and are loosely interfaced with minimal automatic data transfer. During the 1970s, the interdependence of activities within and outside the shop floor was recognised. To achieve integration in production control, all the functions in the model of a production planning and control system have to be operationally associated (all modules have to be fully and thoroughly integrated (on operational rather than abstract levels) with the possibility of mutually interchanging data in real-time). Since the 1980s, these

[1] *"In the factory of the future, automated systems will facilitate production at every step, beginning with electronic receipt of the customer order. The product will be designed uniquely for the customer on a CAD system, and routing will be created on a computer-aided process planning (CAPP) system in accordance with the resource capacities and workloads. Instructions will be given to the automated storage and retrieval system (ASRS) to pick materials and tools. An automated guided vehicle (AGV) will deliver these materials to robots and direct numerical control (DNC) machines, which will perform the current processes. The vehicle will then deliver the product to the shipping dock. Artificial intelligence systems will tie all these systems together and make adjustments for any conflicts. Data will move from customers to supplier and through their respective facilities smoothly, without the stopping and starting that exists when people are the interface"* [Mather, H., 1986].

interdependencies have been integrated into single large-scale computerised systems. In an attempt to develop integrated approaches, several production control frameworks have emerged. However they are typically all based on two well-known approaches, 1) Materials Requirements Planning (MRP) and its extension Manufacturing Resource Planning (MRPII), and 2) Just-In-Time (JIT). For detailed description of these approaches a reader is referred to some of readily available literature sources (e.g. [Vollmann et al., 1992]). The next three sections only provide a brief summary of their characteristics.

Traditional frameworks for manufacturing planning and control are nowadays supported by a widely available set of production planning and control systems and appropriate software, from master production scheduling to the back-end systems (refer to Figure 3-2). Moreover, the more recent software is becoming more integrated. For example, the master production scheduling produces the right input for the development of detailed material and capacity plans, which in turn provides the right input to the operational systems. An example of several commercially available packages includes: MAPICS (IBM), SCHEDULEX (Numetrix, Ltd.), MCS-3 (Micro Manufacturing Systems), Cullinet (Cullinet Software, Inc.), AMAPS (Camsen Corp.), PMS (Boeing Computer Services), Factorial (Factorial Systems Inc.), and MAC-PAC (Arthur Anderson). However, these packages are usually designed for specific types of manufacturing environments and discussion of the logic on which these packages are based (details of how the software operates) does not exist [Subhash & Sanchoy, 1994].

3.3.1 Materials Requirements Planning (MRP)

The earliest Production Planning and Control (PPC) frameworks were based on the Bill Of Materials, Gantt charts, and parts expansion. During the seventies these early systems evolved into Orlicky's Materials Requirements Planning (MRP) computer based software system [Orlicky, 1975] that is one of two fundamental approaches to PPC. Section 3.3.3 describes the other – the Just-In-Time (JIT) approach.

An MRP has a central role in a PPC system. It translates the overall plans for production (Master Production Schedule - MPS) into the detailed individual steps necessary to accomplish those plans on a shop floor. MRP schedules and controls the total flow of materials from the raw materials to the finished products on a time basis (usually, weekly). The linking (integrating) mechanism is the dependent demand concept, which results in the time-phased requirements of materials and parts.

MRP is designed to make purchased and in-house manufactured parts available for dispatch before they are needed by the next stage of production. MRP executes the production-order scheduling and purchase-order scheduling activities and it is primarily a mechanism for initiating production and assembly of parts. MRP requires three basic inputs:

- The Master Production Schedule (MPS), which is the plan for products to be offered to the customers. MPS is an aggregate plan showing required amounts of products versus planning periods for multiple end items to be produced. MRP takes a time-phased set of master production schedule requirements and produces a resultant time-phased set of component parts and raw material requirements.
- A Bill Of Materials, which shows for each part number what other part numbers are required as direct components, and
- The inventory status. To know how many parts to make, we must know how many are on hand, how many of those are already allocated to existing needs, and how many have already been ordered.

The core of the MRP system is the detailed material planning function, which is based on the ***time-phased planning*** and the associated time-phased (period-by-period) requirement records. Time-phased detailed material planning is carried out on a level-by-level basis corresponding to the levels in the Bill Of Materials (BOM) – expansion of requirements. Detailed planning is required for each level in the BOM, and lead-time offsetting is utilised at each level. Time-phased planning also facilitates schedule changes and revisions in customer delivery dates as well as changes in product mix. Under MRP, plans are typically updated on a periodic (daily[1] or weekly) basis to develop priorities for scheduling manufacturing and supplier operations.

MRP approach is appropriate to apply ***for products with a quite complex product structure*** and markets that are characterised by a high rate of new product introductions, rapid shifts in product technology, and custom-engineered products. As such ***MRP systems are particularly well suited for batch job shop manufacturing systems which are based on a functional layout and which typically produce a wide variety of products in low volumes***. This is one of the principal reasons for the high rate of adoption of these systems. MRP is based on the premise that parts are routed to different parts of the factory for processing steps, high work-in-process inventories, relatively long lead times, and high utilisation of workstation capacities. This is why the MRP approach is often favoured in manufacturing facilities employing expensive equipment. Activities in the MRP-based systems are triggered by paper work, authorising production quantities, routings, due dates, and so forth.

Criticisms of MRP are common in the literature. The problems with MRP can be classified as relating to technical factors, process factors, and inner environmental factors. The state date is usually back calculated from a required delivery date on the basis of present lead times. (MRP is a "backward" scheduling method with a "push" type production). This back calculation is made ***without considering resource availability***. MRP is often described as an infinite loader without capacity consideration. Another problem with MRP is its static nature. Lead times and schedules are intimately correlated, but ***MRP assumes that schedules depend on lead times and not the converse***. There is no feedback mechanism in MRP, and it functions primarily as a planning methodology and not a controlling methodology. MRP is a centralised planning system and it attempts to integrate the plant through its centralised plan. ***MRP does not consider the time-phased requirements of machines, labour, tools, and other resources***. Its primary purpose is to organise the complex activities occurring on the plant floor through the use of database technology. The benefits of MRP are primarily due to this organisation and resulting information flow.

3.3.2 Manufacturing Resource Planning (MRP-II)

To account for some of the problems stated at the end of the previous section, Material Requirement Planning (MRP) was enhanced to Manufacturing Resource Planning (MRP-II) [Wright, 1984]. In addition to MRP, MRP-II consists of two additional subsystems: The first includes long-range planning and master scheduling; the second focuses on ***shop floor control.***

The major enhancement of MRP-II is that ***it includes capacity planning functions.*** By involving detailed capacity planning the operations on shop floor have been improved. Later, it was realised that due dates for shop orders can be updated by MRP re-planning, and that a viable master production schedule (one that could be executed) can be maintained. At that time these systems started to be described as closed loop MRP systems. Because of the improved execution on the shop floor, in MRP-II the resultant plans become more and more believable. In addition,

[1] This assumes that the MRP elements can be re-calculated rapidly. That is, that there is sufficient computing power to undertake these recalculations in a timely manner.

simulation possibilities were added along with various ways to examine "what-if" scenarios. Because of all these changes, when compared with the original concept of MRP, MRP-II is described as being the total management system for integrating and coordinating the activities of all functions in manufacturing.

The capacity planning functions is comprised of five modules that range from long-range resource planning to day-to-day control of capacity utilisation. Figure 3-3 shows the relationship between Production Planning and Control (PPC) framework and various capacity planning modules. Each capacity planning technique's basic intent is to project the capacity needs implied by the material plan, so that timely actions can be taken to balance capacity needs with the capacity available. Also, the figure depicts the place of a shop floor control (SFC) system in the broader context of the overall PPC framework. SFC is discussed in Section 3.5.

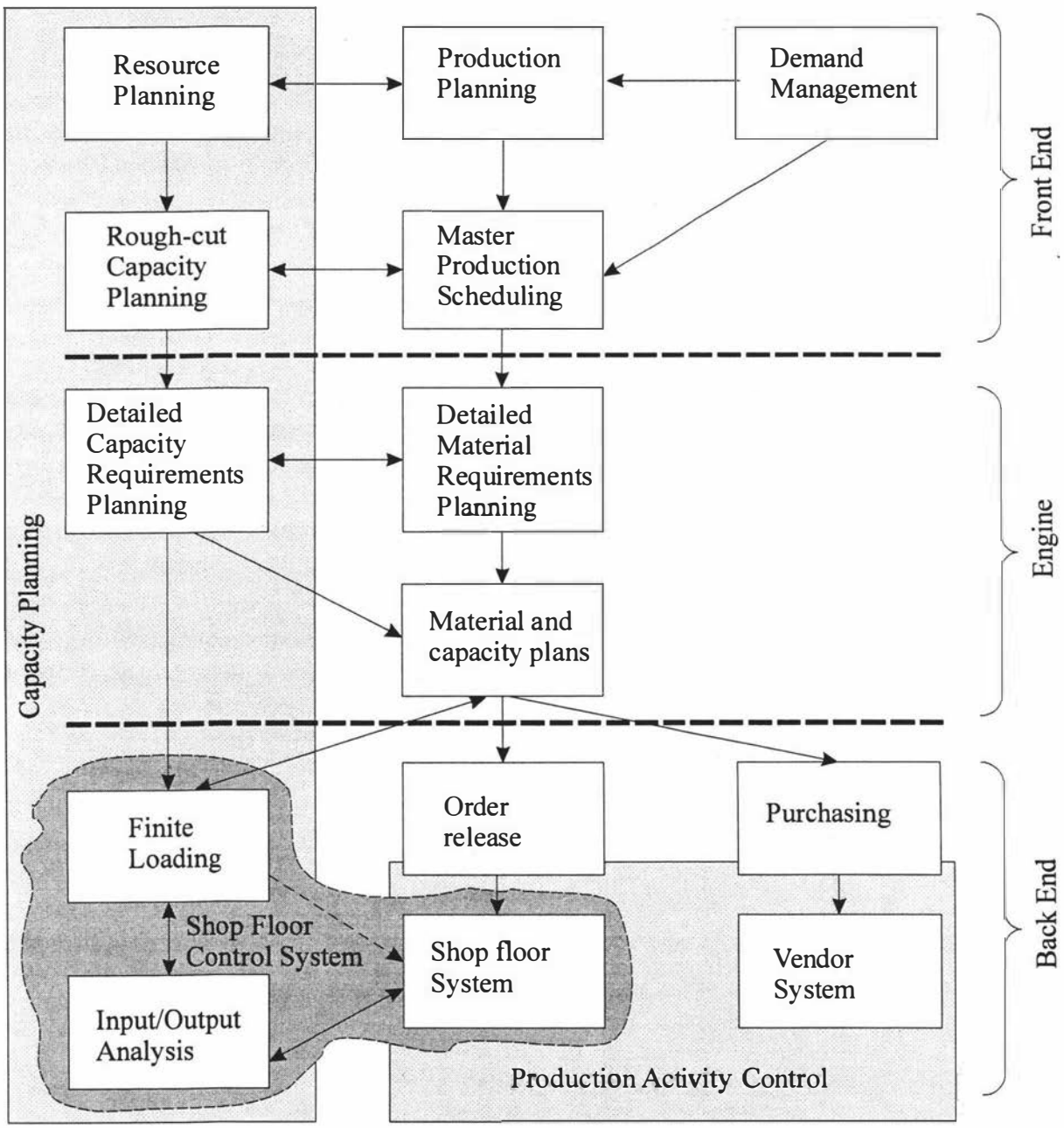


Figure 3-3. Capacity planning in the Production Planning and Control (PPC) system

Capacity plans must be developed concurrently with material plans if the material plans are to be realistic. Only as much material can be utilised as there is capacity for its production, regardless of the material plan. In the same vein, the relationship between flexibility and capacity must be discussed. ***It is not possible to have perfectly balanced material and capacity plans and, at the same time, be able to easily produce emergency orders.***

Nowadays, many companies use and adapt the MRP approach because time-phased records are basic to the understanding of many other aspects of the PPC system. Many firms have PPC systems based on a weekly cycle for MRP planning which serves their purpose well.

The effectiveness of MRP-based systems depends on the accuracy of the data. Obsolete information and inertia of MRP system were the main drawbacks (culprits for inefficiency) of MRP. Companies that use detailed procedures for Capacity Requirements Planning (CRP) ***require additional information from the detailed material planning and shop floor control systems*** to account for the exact timing, quantities, and status of component parts and end item production orders. Referring to Figure 3-3 (MRPII), going from the top downwards, each technique for material and capacity planning requires increasing amounts of data. There is general relationship between the amount of data required and the quality and detail of the capacity estimation. However, the level of detail appropriate for capacity management implies a corresponding level of detail in the database and in database maintenance. It means that ***these techniques incur successively increasing computational cost and computational time.*** In conclusion, ***MRP systems require detailed tracking of orders and materials***. Because of a number of transactions involved they require considerable computer support for interchanging and processing data. These activities can create a significant overhead costs.

3.3.3 Just-In-Time (JIT)

Just-In-Time (JIT) is an approach designed to eliminate waste in time, material, and energy in manufacturing. JIT was pioneered in the Japanese companies. The well known Kanban method of JIT was developed by the Toyota Corporation to plan and control complex manufacturing systems and to reduce inventory levels in the production process. The JIT approach in execution is focused on simplicity. Based on manufacturing in product lines (usually formed into U shaped manufacturing cells), the intent is to design a system where products flow through the system routinely without quality and disturbance problems.

JIT as a philosophy affects all areas of the Production Planning and Control framework shown in Figure 3-2, (and much more, influencing, for example, product design, process design, and human/organisational elements).

At the front end of a JIT system, it will still be necessary to do rate-based master production scheduling (MPS), production planning, capacity planning, and material requirements planning based on component expansion. A monthly production plan is usually converted into a daily build MPS. This schedule is matched against the backlog of customer orders and forecasted orders and an adjusted schedule is produced. The main concept behind the JIT philosophy is to significantly reduce the detailed material planning and associated transaction costs. This reduction can be achieved by reducing the Bill Of Materials to two or three levels and by involving production operators in the detailed tracking of materials through the system.

The primary application area for JIT is in the "back end" execution. In the back end, JIT offers the potential for eliminating large portions of standard shop-floor control systems. Detailed material planning under JIT concept is carried out with ***rate-based planning***. The primary intention of rate-based scheduling is to establish rates of production for each part in the factory

(for example, 10 parts per hour). Realising these rates allows the company to move material through the manufacturing system without stopping, in the shortest time possible.

As a technique, JIT is based on the principle of pulling materials through the manufacturing facility (rather than pushing it by issuing production orders). JIT-based systems produce in response to the downstream use of the item, which may be workstation by workstation or may be in response to demand for the overall end item. All movements and production are authorised by a signal from a downstream workstation when it has a need for component parts. The signals for communicating a demand from a downstream workstation vary widely and can include: 1) sending an empty container back to be filled 2) using painted space on the floor that holds a specified number of parts and 3) using cards (Kanbans) to say more components are needed. Based on a philosophy of pursuing zero inventories, fast material movement, and routine execution of schedule, JIT greatly reduces the complexity and costs of detailed shop material planning and scheduling done by central staff, work-in-process inventories and lead times, and the transactions associated with shop floor and purchasing systems. Detailed status information on work-in-process items (detailed shop-floor tracking records) are not needed because of high rates of material flow (jobs flow through the system in short cycle times), negligible work-in-process inventory levels, and short manufacturing lead times. Capacity planning is unnecessary under JIT operations since minimal work-in-process levels means there is no need to estimate the impact on capacity requirements of partially processed work. There are no work orders and their associated data. Also, input/output and backlog control is not an issue under JIT because the actual input becomes the actual output with an insignificant delay. All of this reduces the manufacturing database's size, a great number of otherwise requested transactions are eliminated, and the number of material planning personnel in comparison with time-phased detailed material planning is reduced.

However, the JIT-based approach with minimal inventory levels ***requires a high degree of stability.*** To operate effectively, relatively constant demands and strong cooperative relationships between companies and their suppliers are required. Stabilised production rates and in some cases levelled production schedules are of critical importance and prerequisites to effective JIT systems. ***Limited product variety*** with accompanying high requirements for setting up machines and level loading are other drawbacks of JIT philosophy. JIT system is much more limited in its ability to cope with rapid changes in the product mix. Also it does not support intensive utilisation of capacities in the same way as time-phased approaches do. As such, ***JIT systems are well accommodated in companies with production line processes, that is, companies that produce a relatively narrow range of standard products in high volumes with stable product designs and simple structure.***

3.4 Summary of conventional approaches in the design of Production Planning and Control systems

Undoubtedly, Production Planning and Control (PPC) systems play a significant role in achieving companies' goals. Because the magnitude of the investment in these systems and the time required to implement changes in them is not negligible, adequate attention should be devoted to their design.

Design options of PPC system must be matched with the ongoing needs of a company's market, the task in manufacturing, and the manufacturing process. Since a PPC system represents a major investment in a business, it must be designed to support the firm's competitive strategy.

Two basic approaches in designing the integrated PPC systems are those that are based on MRP and JIT concepts. An overview of the main characteristics of both MRP and JIT approaches, and

their linkages with market requirements and the manufacturing strategy is given in Table 3-1. There are, of course, other approaches such as periodic control systems, and Optimised Production Technology (OPT), but they can all be consider more or less as derivations and extensions of the fundamental ideas implemented in these basic two concepts. Which of these two approaches will be implemented depends on the marketplace requirements, the manufacturing task and manufacturing process in which PPC systems operate.

Table 3-1. Main characteristics of MRP and JIT approaches

	Strategic variables		***MRP based approach***	***JIT based approach***
Market requirements	Product design		Custom	Standard
	Product variety		Wide	Narrow
	Ability to cope with product mix		High potential	Limited
	Individual product volume per period		Low	High
	Delivery	Speed	Achieved through schedule changing	Achieved through inventory
		Schedule changes	More difficult	Straightforward
	Accommodating demand changes	Total volume	Easy/incremental	Difficult/stepped
		Product Mix	High	Low
Manufacturing	Process choice		Low volume batch	High volume batch or Line
	Changeover cost		High	Low
	Work in process		High	Low
	Organisational control		Centralised	Decentralised (Shop floor based)
	Source of cost reduction	Overhead	No	Yes
		Inventory	No	Yes
		Capacity utilisation	Yes	No
		Planning staff and planning computer time	No	Yes

3.5 Traditional approaches to Shop Floor Control (SFC) systems

Nowadays, a wide variety of manual and computer-based shop-floor scheduling and control techniques exist. The two basic approaches (material planning driven by MRP and material planning driven by JIT) depend greatly on the manufacturing process's characteristics. From the

previous discussion (refer to Sections 3.3.2 and 3.3.3) it is clear that a Shop Floor Control (SFC) system based on a detailed material planning system and a time-phased planning approach (MRP/MRPII concept) is more suited to a job shop manufacturing systems that are based on functional layout and which fabricate a wide variety of products of a complex product structure (several levels in BOM) in low volume batches. On the other hand, an SFC system based on rate-based approach (JIT concept), which require a much more stable production environment with the small extent of deviations in product mix and disturbances in the arrival of work orders, is more suited to manufacturing systems that are based on product lines layout and which produce a narrow range of products with a simple structure. Since our interest is on the job shop manufacturing systems that produce versatile product types in small quantities under unstable production conditions, we will focus our attention in this section to SFC systems that operate under MRP/MRPII approach only. The JIT-based approach is definitely not suitable for this type of manufacturing facilities.

3.5.1 Comments on Shop Floor Control system under MRP/MRPII

Shop floor control (SFC) system under MRP/MRPII (Materials Requirements Planning / Manufacturing Resource Planning) is quite complex and centrally driven. ***SFC concerns scheduling and control of individual jobs at all machines on the shop floor.*** One objective of these systems is to utilise each workstation's capacity effectively. This form of manufacturing control is based on relatively large batches of each component and significant work-in-process inventories to support independence among the workstations. The MRP/MRPII shop-floor approach is based on scheduling shop orders that dictate the set of detailed steps or operations necessary to make each component part. The flow of materials is controlled with dispatching rules establishing the order in which all jobs in a particular workstation are to be processed. The schedule for any workstation varies depending on the batches that arrive at that workstation. The primary criterion in establishing this order is the due dates for the parts, which are continually re-established through MRP/MRPII planning. Shop orders are tracked as they progress through the factory by processing detailed transactions of work at every workstation. Relationships between shop floor control and other parts of production planning and control systems are bidirectional. Orders are issued by the PPC to the shop floor (top-down flow of information) and feedback information from the shop floor (about the status of machinery, etc) is retrieved back to the PPC system for updating data (bottom-up flow of information). Shop orders are opened as part of MRP/MRPII planning, and they are closed out as components are received into a stockroom. Problems are highlighted through input/output analysis and shop load reports.

The most common scheduling algorithms used in industry are variations of forward/backwards scheduling algorithms that are found in the centralised and hierarchical approaches. These algorithms form the backbone of all commercial MRP/MRPII systems[1] [Parunak, 1996a]. These systems push jobs forward or backward in time as they are scheduled to keep capacity utilisation close to 100%. Current state-of-the-art forward/backwards scheduling systems support a number of features. Most are finite-capacity systems that can schedule by job, by task within the job, or by machine. Many have net-change scheduling methods (also called dynamic or network rescheduling) that reschedule only those parts of the schedule affected by changes. Some can

[1] MRP systems iteratively schedule orders forward from the time they are released into the factory, or backwards from their assigned due-dates. To simplify their operation while developing a forward/backward schedule, capacity constraints were originally ignored (the MRP concept was introduced almost 30 years ago when computers had limited computational capabilities). Such systems are called infinite-capacity MRP systems. Advances in computer hardware that took place in the last twenty years allowed vendors to change their algorithms to finite-capacity MRPII systems.

schedule both people and machines to perform a single task. Many of these systems can schedule by the most highly constrained resource first to achieve Goldratt's concept of synchronous manufacturing [Goldratt, 1990].

MRP/MRPII systems are accompanied by relatively high overhead and work-in-process inventory costs. MRP/MRPII requires a substantial volume of shop transactions to provide control reports for order tracking, dispatching, and workstation monitoring. To perform this task a large manufacturing database in a high-speed computer needs to be maintained and a significant number of planning staff needs to be employed.

The capacity of a manufacturing system is of critical importance for a successful SFC system. In essence, the capacity represents resource availabilities for meeting material plans. *If insufficient capacity is provided, no SFC system will be able to decrease backlogs, improve delivery performance, or improve output. On the other hand, if more than enough capacity exists to meet peak loads, almost any SFC system will achieve material flow objectives. It's in cases with bottleneck areas and where effective utilisation of capacity is important, that we see the utility of good SFC systems* [Vollmann, et al., 1992, pp.170].

3.6 A list of desirable features of manufacturing control systems

From the background discussion given in this chapter a list of desirable features of manufacturing control systems, which apply to low-volume job shop manufacturing systems, can be assembled. Some of these features are listed below.

An effective production control system should be able to provide ***an integrated control of all required production resources*** (machine tools, parts, operators, necessary tooling, appropriate instructions, and transportation resources) to ensure balanced use of resources and to reduce surprises. Traditional approaches to production control typically control only a subset of these resources and, as a result, are often unable to meet the production schedule due to constraints imposed by the resources that have not been unaccounted for.

Minimisation of the Work-In-Process (WIP) inventory has always been a major objective of production control systems. In future control systems it will be almost imperative to maintain a low WIP level. In the past, the practice has been to increase the WIP to cover other more fundamental problems. To achieve a lower level of WIP will put a strain on production control.

Due attention should be paid to the ***utilisation of bottleneck workstations***. If the bottleneck capacities can be more intensely utilised the overall schedule performance can be improved in several ways. Conversely, utilisation of non-bottlenecks is not a high-priority action (unless they become bottlenecks themselves as the throughput of the original bottleneck is improved) [Vollmann et al., 1992].

Most importantly, manufacturing systems need to be able to respond rapidly to the customer demands that more often cannot be forecast. To do this, more and more of the ***manufacturing decisions need to be transferred down to the factory floor***. Theoretically, the system that operates without a need for forecasting would perform the best in the circumstances in which the production is characterised as extremely unstable and changeable. To make a system more responsive, the majority of short term planning and control decisions need to be moved from the higher levels of manufacturing control to the shop floor. That is, the manufacturing control systems should be design in such a way to rely on forecast information as little as possible.

A control system needs to be able ***to deal with a very short scheduling time frame***. The scheduling time frame (in MRP/MRPII terminology known as a "time bucket" or "planning

period") is the time interval after which the schedule is regenerated. In the future, scheduling periods need to be, and can be expected to be much less than those of many current systems due to the real-time processing capabilities of modern computer based control systems.

The control system should ***provide real-time control***. Real-time control is a term for the production control mechanism that enables immediate response (directives to the system) after receiving new information about unexpected events (equipment breakdowns, order changes, vendor failures, priority changes, etc.). We are talking about *real-time control* if "good" control decisions can be generated at a rate much greater than that at which uncertainties occur in the system. Such control can be expected highly to be effective and will lead to an acceptable system performance. There are two traditional approaches to achieve real-time control of unexpected, but relevant events. In the first approach, the scheduling interval is reduced to such an extent that the schedule of all activities in the system can be completed each time an unexpected event occurs. In the second approach, when an unexpected event occurs, the controller needs to readjust the existing schedule and to perform rescheduling only of those resources that are affected. This also needs to be completed very quickly so as not to disrupt the overall schedule. Ideally the preference would be to achieve the first option, but realistically, the second option is more likely to be achievable because of the lower and hence achievable computing requirements in terms of speed and power.

3.7 Overcoming issues of conventional Shop Floor Control systems

Modern production planning and control needs to be able to cope with ***frequent changes and uncertainty of events*** in the manufacturing environment. In a low-volume job shop manufacturing system, ***forecasting of required end items is usually difficult*** because customers are more actively involved in product definition. (A trend is to choose from a long list of possible product options through, for example the Internet, and then have the product matched to the customer requirements rather than taken from stock). ***Growing complexity*** is another characteristic of today's manufacturing which manifests itself not only in manufacturing systems, but also in the products to be manufactured, in the processes, and in the company structures. Difficulties arise from a multitude of interactions in attempting to control various activities in dynamic shop floors. Furthermore, ***it is difficult to buffer customer demands*** with large inventories since the components in these inventories may or may not be required as customer requirements change with time. To avoid being left with outdated items ***production moves towards one part production***. At the shop floor level capital utilisation also has to be re-examined in light of the organisation's overall objectives. The ***equipment utilisation is relatively less important when compared with the ability to respond to surges*** brought on by changes in requirements. As a result, planning and scheduling should be executed without levelling the machine utilisation (as in JIT approach) or the use of complex scheduling systems (as in MRPII approach). This means that fixed assets (both capital and people) will probably be less intensively utilised to increase material velocity but more to increase the overall system responsiveness.

A Shop Floor Control (SFC) system which is based on one of traditional approaches, either MRP or JIT, performs satisfactorily if the system is applied in a manufacturing facility in which the arrival of the new work orders is predictable and known in advance (for example, when the work orders are repeated over the time following more or less the same time patterns). However, ***both technologies, MRP and JIT, experience difficulties and often cannot produce satisfactory results when they need to be applied in "random" manufacturing systems***. In practice, in such systems in which a high variety of product orders arrive randomly, the JIT approach is not

applicable at all (because of the unstable demand and highly changeable product mix) while systems based on MRPII approach endeavour to cope with the frequent changes and unpredictable demand but often become bogged down in the rescheduling calculations. This complexity and uncertainty seriously limit the effectiveness of conventional control and scheduling approaches.

MRPII systems are primarily developed for companies that base their production on the "produce to stock" principle. These systems employ a centralised scheduling approach based on forecasting and historic data so that when activities are to be executed on the shop floor, typically it is extremely difficult to get a real-time overview of the production schedule. The ***MRPII systems are characterised by high Work-In-Process inventories***, work-parts usually have long lead times, workstation capacities are highly utilised, and plans are typically updated on a periodic (daily or weekly) basis. In addition, ***shop floor control systems under MRPII are very complex***. The shop floor control system covers scheduling and control of individual jobs at all machines on the shop floor. The characteristics listed above for MRPII systems are exactly the reverse of the desirable features of modern control systems in "random" manufacturing. MRPII systems are often inadequate when manufacturing becomes more order-driven and dynamic and when it is more important than ever for batch manufacturers to be able to adapt their production to changes, e.g. unforeseen interrupts, lack of tools, order changes, rush orders, machine breakdowns, etc.

On the other hand, distributed heterarchical SFC systems (discussed in Chapter 05) are considered as systems adaptable to frequent and abrupt changes, capable of coping with a broad spectrum of uncertainties such as those present in a production environment. Therefore, distributed heterarchical SFC systems are seen as a convenient approach for solving control and scheduling issues in random manufacturing systems. In addition, it is believed that all other manufacturing systems that are using SFC systems based on MRP also can yield benefit. The main gain is expected through the elimination of the need for detail shop floor scheduling and, therefore, significant reduction of shop floor information and data base transactions that are requested by the MRP system for tracking orders, dispatching work-parts, and monitoring workstation's activities.

The research presented in this thesis concerns the design and development of the core part of an experimental distributed heterarchical shop floor control system believed to be able to overcome many of the problems discussed above in SFC systems for random manufacturing systems. The design and architecture of this system are discussed in Chapter 10, while Chapter 11 describes how the experimental system operates.

Chapter 4. Production scheduling

An overview of the two basic technologies that are used in production planning and control of manufacturing systems (MRP/MRPII and JIT) is given in the previous chapter. This chapter addresses issues that occur when production plans need to be implemented by scheduling tasks on the shop floor.

The chapter provides review of classical scheduling research conducted mostly in the domain of job shop manufacturing systems and plots a course for the distributed scheduling approach adopted in this project. Findings from this review enabled the extraction of those methods (for example, providing routing flexibility, overlapping operations, selecting "good" scheduling rules) which were reported to contribute significantly to the increase in the overall performance of manufacturing systems. These techniques were implemented in the design and development of a distributed shop floor control system advocated in this research project. For instance, the processing route tables were used to significantly increase system flexibility, the minimum operation completion time was used for selecting workstations to perform the next processing tasks, the shortest processing time (SPT) and the earliest due date (EDD) rules were used for scheduling jobs on workstations. This chapter also provides background information that is relevant for the future work in this project; namely, to facilitate the measurement of the performance of the proposed system, several performance criteria (measures) that were used in the simulation experiments for evaluation of scheduling rules were also addressed. On the basis of the review it was noted, for example, that:

- Minimum make-span and machine utilisation were the most common performance criteria, but others such as: average order flow time and average order lateness, total number of early orders, total number of late orders, and total number of operations processed, also could be used; and
- Jobs arrive in the system at random time intervals, most often with a Poisson distribution.

Only the basic, most common techniques and approaches used for scheduling jobs in manufacturing systems (optimisation fabrication processes) are outlined. It is beyond the scope of this chapter to summarise the vast amount of research on this topic. Since some of the first books on scheduling appeared, more than 20,000 articles about the scheduling problem have been published [Dessouky et al., 1995]. Therefore, the interest here is to focus on some basic concepts and results and to relate them to the distributed scheduling approach, which is the topic under investigation in this dissertation. The chapter finishes by pointing out a multi-agent scheduling methodology that emerged recently as promising techniques for solving complex scheduling problems.

This chapter is organised around the following topics:

Section 4.1 introduces shop floor scheduling and accentuates its importance for the successful operation of a manufacturing facility.

Section 4.2 provides the key definitions of scheduling and highlights the complexity of the scheduling problem. Further, the section addresses a few approaches that are used in practice to cope with the difficulty of the scheduling problem and discusses the main facets of the scheduling framework (such as: scheduling objectives, machine and job characteristics, product

structure, and criteria for judging scheduling performance) that need to be defined for any particular organisation.

Section 4.3 overviews the commonly used scheduling techniques and introduces briefly some relevant, non-traditional scheduling techniques, such as simulated annealing, tabu search, and genetic algorithms, that emerged in recent years as a result of efforts spent in finding effective tools to cope with complex scheduling problems.

Section 4.4 addresses single and two machine scheduling problems and reviews a few studies dedicated to scheduling jobs in flexible manufacturing systems.

Since the main focus of this research project is on the philosophy based on a multi-agent paradigm for solving difficult and complex scheduling problems in large-scale job shop manufacturing systems, Section 4.5 briefly introduces the agent-based approach. However, the whole Chapter 6 is dedicated to the review of research work conducted in the field of Multi-Agent Systems.

4.1 Introduction

Scheduling is generally described as ***allocating a set of resources to perform a set of tasks***. There are many different kinds of scheduling problems, and consequently, many ways to think about scheduling. It emerges in various business domains (airlines, hospitals, etc.) but in this study the focus is on production scheduling or more specifically on scheduling at the shop floor level of job shop manufacturing facility.

It should be noted that the terms planning and scheduling sometimes have overlapping meanings in technical literature. Some authors are talking about "scheduling" when they referring to planning functions while others use terms such as "fine planning", "operating planning" or "short-term planning" when referring to scheduling functions. This is because the production scheduling in its broadest sense spans all the levels of production planning and control from the forecasting of future sales volumes and market demand through master production scheduling (MPS) and down to detailed materials requirements planning and shop floor scheduling. At each of these levels scheduling involves the matching of work against resources on a phased time basis. For the purposes of this project, we will use term scheduling to indicate the process of determining start times and resource assignments for part production ***at the shop floor level.*** Shop floor scheduling, hereinafter referred to as scheduling only, involves the sequencing of jobs against resources on the shop floor in an effort to meet the due-dates in the released orders.

Scheduling has been considered as an important part of manufacturing for reducing the cost of manufactured products and the time of production and delivery of those products to their customers. The effective scheduling of a facility reduces its work-in-process (WIP) inventory, and increases equipment and labour utilisation, throughput, and productivity – effectively increasing the return on investment. This is why improvements in the scheduling of manufacturing operations can often contribute significantly to the overall success and profitability of a company.

Because in many cases scheduling determines how rapidly engineering changes can move through a system, and thus how responsive a company can be to changing markets, scheduling techniques that can adjust to real-time changes in the manufacturing environment are of particular importance. (The proposed model of heterarchical shop floor control system has this ability. Chapter 10 explains how jobs are scheduled to machines in real time taking into account current circumstances on the shop floor – including availability of tools, jigs, and fixtures on workstations).

4.2 A scheduling framework

A job-shop scheduling model is one of the most popular models in scheduling theory. It is considered to be a good representation of the general domain and has earned a reputation for being notoriously difficult to solve. The job-shop model is probably the most studied and well-developed model in deterministic scheduling theory, serving as a comparative test-bed for different solution techniques, old and new [Jain & Meeran, 1999]. What follows is a brief framework for scheduling, which includes key definitions, criteria for judging scheduling performance, and some other facets that are important to be defined when considering scheduling problems. More detailed discussion on the scheduling framework is provided in [Vollmann et al., 1992].

4.2.1 Definitions of scheduling

In manufacturing systems, scheduling is defined as *"allocating a set of machines to perform a set of work orders within a certain time period"*. The result of scheduling is a timed and sequenced set of jobs, a schedule, which is defined as: *"a plan with reference to the sequence of and time allocated for each item or operation necessary to complete the item"* [Vollmann et al., 1992]. Scheduling can also be viewed as *"a dynamic and adaptive process of iterative decision making and problem solving, involving information acquisition from a number of sources, and with the decisions affecting a number of production facets in reaction to immediate or anticipated problems."* [McKay & Wiers, 1999].

The primary scheduling task is ***to ensure that every part is produced on time***. Therefore, it is important to control the time a part spends on the shop floor. This time is made up of four components, *processing time, set-up time, travel time, and queuing time.* Based on these four components, due-dates, routing data, and equipment status, a shop floor scheduling system ***determines when and which actual machines should be used for which products.***

Scheduling can be viewed from several different aspects:

- ***Static scheduling*** deals with the scheduling of a fixed set of jobs in which all jobs are simultaneously available for processing. There are two assumptions in static scheduling:
  - all jobs must enter the system at the same time and
  - all machines must be available for processing at that time.
- ***Dynamic scheduling*** refers to the situation in which new jobs keep entering the system constantly over time. Dynamic scheduling also refers to the situation when a scheduling decision must be made in real time whenever an unexpected event happens in a manufacturing environment.
- In ***deterministic scheduling*** it is implicitly assumed that all aspects of the scheduling problem are known when scheduling takes place. This is a strong assumption, which may cause severe difficulties in practice.
- In ***stochastic scheduling*** it is anticipated that some aspects of the scheduling problem are unknown prior to scheduling. Strictly speaking, all real-life scheduling is stochastic, as the complete problem will never be known beforehand (machines may break down, workers may get sick, deliveries may be delayed, jobs may be cancelled, due dates may be changed, and processing times can only be estimated as being within a certain interval).

Many studies in a domain of job shop scheduling are based on static and deterministic problems (regardless of their very limited practical values) as simplified versions of real life scheduling problems, which are stochastic and dynamic. The stochastic aspect is then covered by recalculating the problem whenever an unexpected event occurs using the new constraints imposed by this event.

In the past five decades, scheduling has been mostly researched from a mathematical point of view, embodied by operations research, operations management, and artificial intelligence communities as a sequencing problem. However, scheduling is more than sequencing and there are many technological constraints that have to be satisfied to obtain feasible (realistic) schedules. Scheduling is often a manual task involving additional information and judgement when a large variety of problems emerge in a real-world situation [McKay & Wiers, 1999]. In the classical scheduling theory, machines are the main resource but a number of important additional resources are involved in the production process; like tools, jigs, fixtures, pallets and material handling devices. Yet, the best way to perceive a scheduling problem is through the schedulers' activities that are performed on daily bases. Schedulers' activities involve:

- Information collection and validation;
- Reconciliation of what happened over some time versus what was supposed to happen;
- Determination of the immediate and pending state of demands, resources, material, and personnel;
- Communication with shift supervisors and upper management;
- The processing of information with respect to machine maintenance and repair; and so on.

From the information collected and processed, as well as their experience, schedulers need to make decisions about what needs to be done next, where, and by whom. However, the answers do not appear to be the simple addition of one or two more variables to a traditional formulation of scheduling problem as a sequencing problem. Scheduling decisions particularly in a large manufacturing organisation encompass a wide range of options, and they are almost always a compromise among several hundred constraining concepts found in real scheduling systems [McKay & Wiers, 1999]. This large number of constraints combined with their dynamic and uncertain nature highlights the size and difficulty of scheduling problems in manufacturing operations.

4.2.1.1 Why the scheduling problem is difficult?

The job shop scheduling problem in most cases is a very difficult and complicated problem. The difficulty is due to a high number of work-parts that compete for time on a fixed number of machines, a number of resource-sharing conflicts which have to be resolved, a high number of constraints that need to be simultaneously satisfy, and many other reasons which some of them are briefly discussed next.

The complexity of manufacturing systems. Scheduling problems, as a special class of combinatorial optimisation problems, are classified according to their computational complexity. For the "difficult" scheduling problems (regrettably most of the scheduling problems in practice belong to this group) no polynomial type algorithms have been established yet [Finke, 1995]. Therefore, attempts to get an optimal scheduling solution of complex manufacturing systems are computationally unrealistic.

The dynamics of manufacturing systems. The complications become more severe when frequent changes, interruptions, and delays occur. For example, changes in processing times,

changes in demand rates, changes in job priority, machine breakdowns, rush orders, and other operational problems make any solution valid only for a short time period.

The uncertain nature of the scheduling problem. Not only is the appearance of most of the events in a manufacturing system unpredictable but other attributes of scheduling also cannot be predicted. For example, it is difficult to predict which objective function should be used for a specific situation on a specific day. It is difficult to know what information should be weighted more than any other information. It is difficult to know what will be the state of the material, resources, processes, and personnel, and so on.

On Line Problem Solving. Scheduling for a real world "job shop" is a problem that has to be solved on-line or, at best, in a very limited amount of time. This leaves little time to correct problems resulting from generating an infeasible schedule based on incorrect information.

Tightly Coupled Problem Space. The elements of a scheduling problem have well-understood relationships to each other, and a change in one element can have predicted results in another. However, even with the well-known relationships, because of the complex interaction among the variables, it is very difficult to accurately evaluate a schedule and to decompose scheduling problems within a single aggregation level [Pickel, 1988].

Little Human Recovery Possible. In fully automated manufacturing systems a schedule has to be nearly perfect because there is no intermediate control level to recover from errors and in complex systems human intervention can, in many circumstances, produce more problems than it solves.

Problem of temporality. Time and the meaning of time are the key components of scheduling process. Firstly, scheduling constraints, objectives, and information can vary rapidly and unexpectedly over time. Secondly, the planning horizon affects issues such as aggregation of information, precision of calculations, and the type of constraints used to make decisions. Static views of the scheduling requirements and algorithms independent of time are not capable of modelling the real situation.

4.2.1.2 Reducing the scheduling problem

To cope with the scheduling problem in day-to-day operations of manufacturing systems, efforts have gone in three directions.

Firstly, from the organisational perspective the ***entire manufacturing system is often divided into several smaller organisational units (departments).*** As a result, the problem space (the entire manufacturing system) is decomposed into smaller and less complex problems (production units - departments).

Secondly, scheduling research has also looked at the relationship of scheduling to other production control functions. In reviewing the most important approaches used in Production Planning and Control (PPC) systems in the previous chapter, it has been seen that, although an integrated solution approach to all levels of the production planning is required, there is no one single model which deals with all problems simultaneously. To reduce the complexity of the scheduling problem, the centralised structure of the overall planning and control system has been replaced with a hierarchical one. Production ***planning and control functions are decomposed into several smaller manageable modules – levels- that are hierarchically organised*** (decisions at the higher level define the constraints for the next lower level).

Thirdly, ***heuristic scheduling methods have been developed with the aim of replacing some of the sophisticated algorithms for exhaustive search for optimal solutions***, thus reducing the computational difficulty while still obtaining a fairly good solution. In practice the most

successful approach is that of using heuristics applied on the jobs waiting at the front of each machine or process. This is achieved by ranking the jobs in each queue on the basis of some simple measure and, when a machine or process becomes available, by choosing the job at the top of the ranking. Some of sequencing heuristics are presented in Section 4.3.2.2 "Priority Dispatch Rules (PDR)".

4.2.2 Scheduling objectives

Three primary objectives or goals that apply to scheduling problems are:

- *Due dates.* The objective is to get products on time that is to avoid late job completion.
- *Flow times.* The objective is to minimise the time a job spends in the system, from creation or opening of a shop order until it is closed.
- *Workstation utilisation.* The goal is to fully utilise the capacity of expensive equipment and personnel.

In this environment, a job is a work-part to be produced as a succession of operations to be carried out on different machines. These three objectives often conflict. For example, due dates can be easier met and flow time can be reduced if more capacity is provided but it reduces capacity utilisation. If extra jobs are released to the shop, they tend to have longer flow times; but the capacity can be better utilised and perhaps due date performance can be improved.

Due to inherent characteristics of a model of shop floor control system proposed in this thesis all three objectives are tried to be addressed "in the best possible way" accordingly to the current circumstances on the shop floor.

4.2.3 Performance criteria

To be able to compare and select the best scheduling alternative and to understand what particular research results mean, for each of the three primary scheduling objectives (refer to Section 4.2.2 "Scheduling objectives"), we must establish appropriate performance measures (exact manufacturing performance criteria). *"An operating scheduling system must have unambiguous definitions of performance and these measures must be congruent with the firm's business objectives."* [Vollmann, et al., 1992]. For example, due date performance can be based on a measure of "lateness" (positive or negative (earliness) deviations from the due date) or the "variability of actual completion dates against due dates". The objective in the latter case is to minimise the variance of lateness, thereby improving the reliability of delivery to customers. For a given feasible schedule, one can calculate the following quantities for each job (work-part) J_j:

Completion time (c_j) – is the time when the job (work-part) is finished.

Flow time ($f_j=c_j-r_j$) – is the difference between completion time and arrival time. Flow time gives the time the work-part is in the system, being processed and waiting.

Lateness ($l_j=c_j-d_j$) – is the difference between completion time and due date time. A positive value indicates tardiness while a negative value indicates earliness.

Tardiness ($t_j=max\ (0,\ l_j)$ - is the time of completion of the job after its due-date. Minimising tardiness improves delivery reliability and competitiveness.

Earliness ($e_j=min\ (l_j,\ 0)$ - is the time of completion of the job before its due-date. Minimising earliness has the effect of minimising the work-in-process inventory and tends to accommodate last minute engineering changes which can be crucial in industrial environments with short design-to-manufacturing lead times particularly when the last minute changes are due to

unforeseen problems in the part design that need to be corrected "at the last minute". These changes can use the earliness to avoid tardiness in the final total flow time for the product.

Slack time ($s_j=d_j-(p_j+n_j)$) – is the difference between the work-part's due date and the sum of the estimated total processing time and the present time. Slack time differs from earliness in that it is predicted or known at planning time whereas earliness occurs after the actual processing.

There are several performance measures of manufacturing systems that are used as criteria for evaluating different schedules. The most common measures that have been examined in manufacturing systems are:

Average flow time (AFT) - is the sum of all the times work-parts spend in the system divided by the number of work-parts processed by the system.

$$\mathrm{AFT} = \sum_{i=1}^{N} \mathrm{f_i} \Big/ N, \text{ where}$$

f_i = actual flow time of job J_i,

N = number of jobs (work-parts) that actually arrived.

Average flow time is a more appropriate measure of performance in a dynamic setting than make-span, which is used in most of the deterministic studies [Tsubone, & Horikawa, 1999]. Average flow time and work-in-process inventory levels are directly related measures. If one of these is increased or reduced, the other changes in the same direction. That is the longer it takes to produce a part the more parts will be present in the plant at a given time for a given throughput.

Maximum lateness $L_{max}=\max(l_j)$.

Maximum tardiness $T_{max}=\max(t_j)$.

Average tardiness is the sum of the differences between actual completion dates and scheduled due dates of the tardy work-parts, divided by the total number of parts. Note: This is the average tardiness for all parts.

Mean tardiness is the sum of the differences between actual completion dates and scheduled due dates of the tardy work-parts, divided by the number of late work-parts. Note: This is the average tardiness for the parts that are late.

Make-span is the total time to process all jobs (work-parts).

Production rate. A total number of work-parts produced divided by the make-span (parts/hour). In batch production, a total number of work-parts divided by the make-span of batch.

Machine utilisation is the number of working hours of a given machine during a selected period (actual working hours) divided by the total number of working hours in the selected period (potential working hours) commonly expressed as a percentage.

All the criteria are not equally important in practice, and therefore, for a given manufacturing facility it is necessary to define the level of importance or relative weights. Most scheduling research has been focused on optimizing only one particular performance measure. For example, most static scheduling research has used make-span time minimisation[1] to estimate

[1] Although it may not be perceived as a good theoretical objective function (make-span does not make any conclusions on due date and work centre utilisation aspects), this criterion is one of the most often used criteria in academic and industrial practice. This criterion has considerable historical significance and was the first objective applied by researchers in the early fifties.

performances of scheduling processes. The criteria applied in dynamic scheduling studies typically involve average flow time, average work-in-process (number of jobs in the system), and machine utilisation.

4.2.3.1 Evaluation strategies

There are several techniques used for evaluation scheduling solutions against the acceptance criteria. Given a candidate schedule, evaluation techniques estimate what costs would be incurred if it were run in an actual factory, but without the expense of actually dedicating the factory and its resources to running a schedule. Two common evaluation techniques are: analytical studies (for simpler scheduling problems) and simulation studies (for large-scale scheduling problems).

Analytical studies (such as mean value analyses and queuing models) estimate the behaviour of the factory by means of models that summarise the behaviour of a system without imitating its function through time. Queuing models were originally concerned with single-machine systems and later they were extended to multiple machines. However, applying queuing theory to the multiple-machine case requires so many limiting assumptions that results in general are only interesting from a research point of view.

Simulation replicates the functional model, mimics the behaviour of the manufacturing facility through time by exercising the model, observes the behaviour of the model, and interprets these observations in the context of the original facility. Simulation studies enable the examination of realistic, multiple-machine, dynamic scheduling situations, in large manufacturing systems. With simulation, various rules' for performance can be examined against several criteria. The size of problems studied can be expanded (machines and jobs) and can accommodate any kind of product structure, inter-arrival time patterns, or shop capacity.

Queuing and simulation models have been typically used for studying dynamic scheduling problems. It was hoped to use the heterarchical agent based model developed in this thesis for simulation studies.

4.2.4 Product structure

A final component of a scheduling framework considered in this section relates to product structure. There are several facets of any framework for scheduling which are related to the product structure:

- ***Type of jobs.*** One of the issues is whether the scheduling problem is dealing only with individual jobs or sets of jobs (work-parts), which are to be collected into one order (lot), or one assembly. In latter case, each component of an assembly should be scheduled to ensure that all are completed at the same time. Otherwise the work-in-process inventory will rise as the final assembly waits for the last sub-component to be completed.
- ***Alternative routings.*** Using alternative routing and defining decision rules for when to use alternative routings can be quite a complex issue. On the other hand, alternative routings can aid operating performance. If one workstation is overloaded and another is under loaded, alternative routings can improve due date performance, flow times, and workstation utilisation.
- ***Operation overlapping.*** If a job (more precisely, a part of the job represented, for example, by a transport lot of work-parts) can be started at a workstation before it is

completely finished on the previous workstation, flexibility of the system can be increased and scheduling performance can be improved.

- ***The extent to which set-up times are fixed or variable.*** In some firms, *set-up* times depend on the sequence of processing jobs. It means that a potential time saving can be made by better sequencing of jobs through a workstation.
- ***Variation of lot sizes.*** It should be determined whether lot sizes are fixed or can be variable. Better schedule performance can be achieved if lot sizes can vary. For example if they are larger for processing through some workstations, which perhaps take less time to carry out their operations than operations on other work-parts, depending on factors such as set up time. If set up time is long then processing larger batches can reduce overall process time.

4.3 An overview of some scheduling techniques

The goal of this section is to overview commonly used scheduling techniques (only their main emphasis are discussed) and show that existing techniques address only isolated parts of the problem. A reader seeking more detail in regards of any of the listed techniques is referred to relevant literature sources. A few research surveys in domain of job shop scheduling and references therein can facilitate this task. [Gupta et al. 1989] present a review of literature concerning the operations aspect of FMS. In [Parunak, 1991] author describes five challenges to the computation of schedules and then classifies existing scheduling strategies under the challenges they address. Recently, comprehensive reviews and comparison of the various scheduling techniques applied to job-shop scheduling problems is provided in [Blayzewicz et al., 1996] and [Jain & Meeran, 1999].

The efficiency of a manufacturing system depends considerably on the selected scheduling method. Figure 4-1 summarises the main techniques, grouped in two main categories, applied to solve the job shop scheduling problems (JSSP). One can either apply an *approximate method* that delivers a good solution in acceptable time (suited for more complex JSSP) or an *optimisation procedure* that yields a globally optimal solution but requires very long computing times and very powerful computers (appropriate only for very simple JSSP).

4.3.1 Optimisation procedures

It has been recognised by many researchers that scheduling problems can be solved optimally using mathematical programming techniques. One of the most common forms of mathematical formulation for JSSP is the ***mixed integer linear programming*** format. The format is simply of a linear program with a set of linear constraints and a single linear objective function, but with the additional restriction that some of the decision variables, used to implement constraints, are (binary) integers. ***Dynamic programming*** and ***branch and bound*** techniques will also give optimal results for the given decision criteria (by searching through multi-dimensional spaces for the 'best' solutions). These techniques are the only ones that can guarantee the optimality of a solution. However, there are limitations in their usefulness.

In exact procedures the time requirement increases exponentially or as a high degree polynomial for a linear increase in problem size except for selected, restricted versions (special cases) of JSSP [Jain & Meeran, 1999]. As a result these techniques are only able to solve highly simplified instances, within a reasonable amount of time. This suggests that suitable techniques for JSSP lie in other domains. Efficient (optimal) methods cannot be found for JSSP instances where $m \geq 3$ and $n \geq 3$. Consequently, the focus of optimisation research has turned to enumerative approaches. Enumerative methods generate schedules one by one using clever

elimination procedures to verify if the non optimality of one schedule implies the non optimality of many others which are not yet generated, thereby preventing the need to search the complete space of feasible solutions.

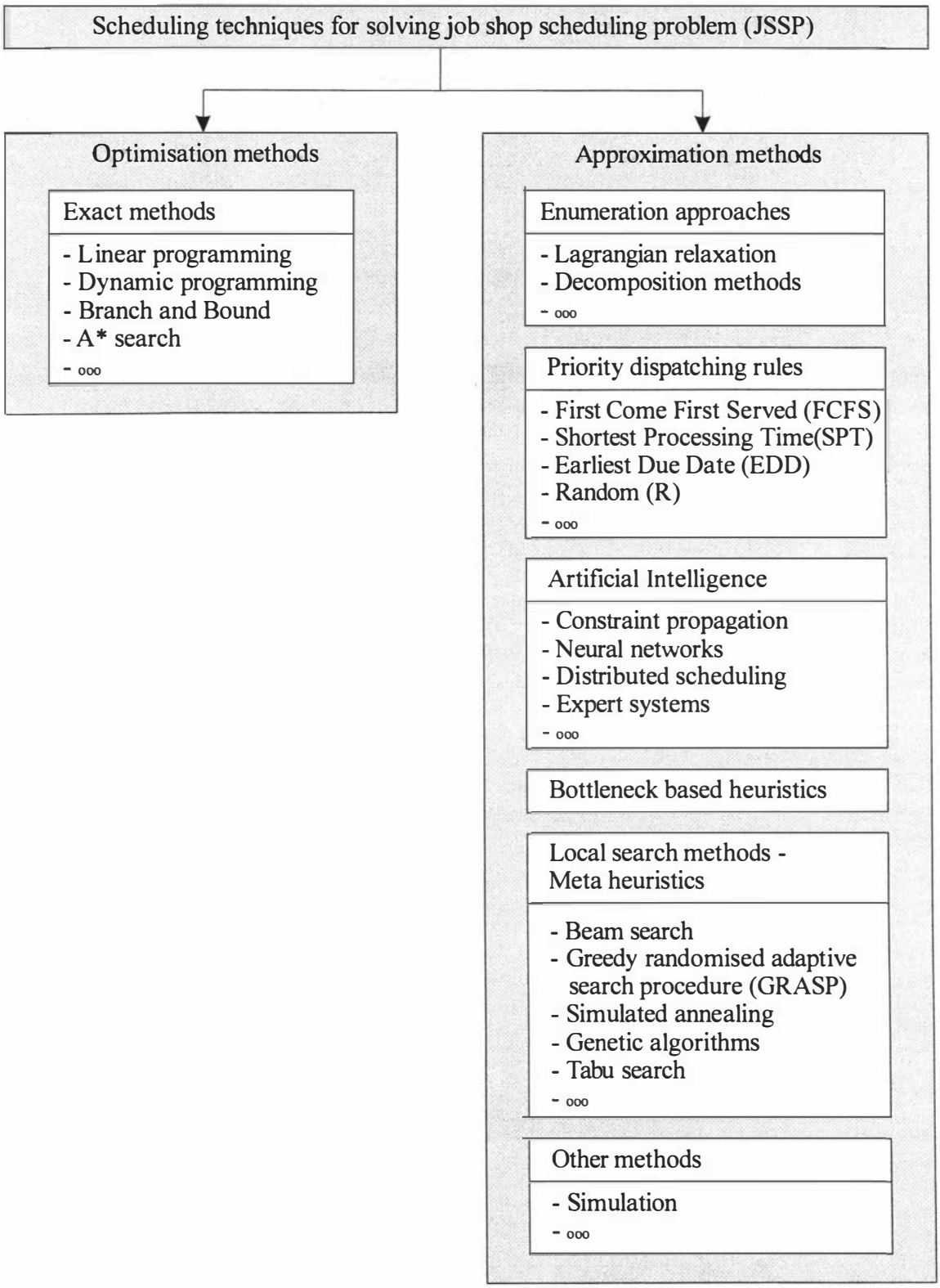


Figure 4-1. Overview of scheduling methods

4.3.2 Approximation methods

For most job-shop manufacturing systems (with large numbers of machines and many jobs) an optimal algorithmic solution to the scheduling problem is not possible ("the best" solution cannot be found within real-world time constraints). This is why many researchers have turned their attention to alternative methods. In the instances, where the solution space is too large to search exhaustively, instead of exact (optimal) approximation algorithms are used. Although approximation methods do not guarantee achieving the optimum solutions, since large regions of the solution space remain unexplored, they are able to attain "affordable" solutions within moderate computing times. For large scheduling problems this is preferred to having a random or unplanned schedule. The goal of approximation methods, which are mostly based on heuristics[1] (investigation of several alternative solutions and selecting the most appropriate one), is to develop a "good schedule" which will work, and will produce much better results than a chance selection offers. In this overview five main categories of approximation techniques are considered: enumerative methods, priority dispatch rules, artificial intelligence, bottleneck based heuristics, and local search methods.

4.3.2.1 Enumerative methods

Any success that has been achieved using mathematical formulations can be attributed to ***Lagrangian relaxation approaches (LR)*** and ***decomposition methods***. In LR methods precedence and capacity constraints are relaxed using non-negative Lagrangian Multipliers, with penalty terms also incorporated into the objective function, while decomposition approaches partition the original problem into a series of smaller, more manageable sub problems which are then solved optimally. The results indicate that even these strategies suffer from excessive computational effort while the resulting solutions are usually of poor quality, resulting in a large deviation from the optimum [Jain & Meeran, 1999]. Even when these mathematical formulations are combined with other techniques they have not performed well. It is evident that mathematical approaches are inadequate for JSSP.

Analytical approaches to scheduling (both optimising and enumerative) tend to require excessive computational resources. Consequently, scheduling algorithms are only applicable to a small number of simplified scheduling problems, but not for more complicated and complex systems typical in the modern manufacturing environment. Further, mathematical approaches make simplifying assumptions which are not always valid in practice. For example, some models assume that tool magazines, pallets and fixtures do not constrain the models in any way; some others will neglect delays in processing some operations, and so on. Finally, the models take a static view of the shop floor. It is assumed that all the planned activities will be carried out exactly, or the disruptions are infrequent enough that periodic solution of the problems will be practical.

Although, mathematical models in the literature are not efficient for reasonably sized problems, these algorithms (above all optimisation algorithms) could be used in establishment of the multi-machine requirements, for example in the circumstance when one of the machines is perceived as a bottleneck in production. The algorithms could then be applied to that machine and the rest of the schedule could be developed around this core by applying other scheduling techniques.

[1] A heuristic is an educated guess, rule of thumb, or simplification that reduces or limits the search for solutions (i.e. provides aid in the direction of solutions) in domains that are difficult and poorly understood. Unlike algorithms, heuristics do not guarantee optimal, or even feasible, solutions and are often used with no theoretical guarantee.

4.3.2.2 Priority Dispatch Rules (PDR)

Approximation procedures applied to job shop scheduling problem (JSSP) were first developed on the basis of priority dispatching rules (PDR). Because the results from application of PDR can be reasonable (if not optimal), and because they reduce substantially computational requirements and are easy to implement (in simulation and real systems), PDR are a very popular and widely used technique for resolving JSSP [Jain & Meeran, 1999].

Priority dispatching rules (also called sequencing rules) facilitate selection of the next job to be processed at a workstation. At each successive step at the workstation all the jobs (operations) which are available to be scheduled are assigned a priority and the job with the highest priority is chosen to be sequenced. Broadly speaking, scheduling rules can be used either as static or dynamic. Static rules are used for off-line scheduling and result in a fixed schedule for the period. Dynamic rules are used for real-time scheduling and they change over time.

The most well-known and comprehensive survey article on scheduling heuristics is an early report by Panwalkar & Iskander where 113 PDR from dozens of simulation studies are presented, reviewed and classified [Panwalker & Iskander, 1977]. [Blackstone et al., 1982] provide an extended discussion and summary of these and many other PDR. One of the more recent comparative studies is by [Chang et al., 1996] who evaluate the performance of 42 PDR using a linear programming model. The following is a limited review of some well-known rules that are representative of the types of rules available and which are often used in an FMS environment.

R (Random). This rule picks any job in the queue with equal probability. The rule is often used as a benchmark for other rules.

FCFS (First Come First Served). This rule is sometimes deemed to be "fair" in that jobs are processed in the order they arrive at the workstation.

SPT (Shortest Processing Time). This rule assigns the highest priority to the part with the shortest next operation (imminent processing) time. This rule ignores all due date information as well as all information about work remaining. It simply maximises the number of shop orders that go through a workstation and minimises the number of waiting parts in queue. As such, this rule tends to reduce work-in-process inventory, average job completion (flow) time, and average job lateness. However, very long jobs can be left in a queue and require some form of additional priority to get through the processes (see SPT/TOT below).

SRPT (Shortest Remaining Processing Time). This rule is an extension of SPT in that it considers all processing time remaining until the job is completed. The rule is also known as LWR (least work remaining).

SPT/TOT (Shortest Processing Time for the operation divided by the Total processing time for the work-part). The SPT/TOT rule is a combined rule that determines priority by selecting work-parts with the smallest ratio of the imminent processing time and the total processing (operational) time. In addition to giving a high priority to short operations, orders requiring a large amount of processing time also receive high scheduling priority.

LPT (Longest Processing Time first). Work-part with the longest processing time for the current operation has the highest priority.

LRPT (Longest Remaining Processing Time). Work-part with the longest remaining processing time has the highest priority.

LPT/TOT (Longest Processing Time for the operation divided by Total processing time for the work-part).

EDD (Earliest Due Date). Jobs with the nearest due date are sequenced first. No allowance is made for the expected operation time. This rule seems to work well for criteria associated with job lateness.

CR (Critical Ratio). Calculate the priority index for each part in the queue by calculating the critical ratio between "remaining time" and "remaining work", (i.e. Due Date minus the Current Time divided by the Estimated Remaining Lead Time). If the ratio is "1", the job is on time. A ratio below "1"indicates a behind-of-schedule job, while a ratio above "1" indicates an ahead-of-schedule condition. The rule is to always process that job with the smallest critical ratio next. This rule is widely used in practice.

ST (Slack Time). Jobs are run in order of the smallest amount of slack, that is, the job with the smallest slack is sequenced first. The "slack" is defined as a time difference between 1) the sum of set-up times and processing times for all remaining operations on the part, and 2) the time remaining until the part due date (due date minus present time).

ST/RO (Slack Time per Remaining Operation). A variant of ST that divides the Slack Time by the number of Remaining Operations, again sequencing jobs in order of the smallest value first.

NQ (Next Queue). A different kind of rule, NQ is based on machine utilisation. The idea is to consider queues at each of the succeeding workstations to which the jobs will go and to select the job for processing that is going to the smallest queue (measured either in hours or perhaps in jobs).

LSU (Least Set-up). Still another rule is to pick the job that minimises changeover time on the machine. In this way, capacity utilisation is maximised. Note this rule explicitly recognises dependencies between set-up times and job sequence.

FOR (Fewest Operations Remaining). This is another SPT variant that considers the number of successive operations. Jobs with the fewest remaining operations are sequenced first.

MOR (Most Operations Remaining). Jobs with the most remaining operations are sequenced first.

As noted, this list is by no means exhaustive. Many other rules, variants of these rules, and combinations of these rules have been studied.

Although results of the individual PDR are achieved extremely quickly, studies indicate that individual rules perform poorly with respect to a selected criterion of performance (in general, solution quality degrades as problem dimensionality increases). This is due to the highly myopic nature of PDR, as they only evaluate one possible operation at each decision point, thereby just considering the current state of the machine and its immediate surroundings. Researchers have tried to improve solution by combining PDR.

4.3.2.3 Artificial Intelligence

Artificial intelligence (AI) is the subfield of computer science concerned with integrating biological and computer intelligence. It has fundamental origins from biological understanding and uses principles in nature to find solutions. Using this natural understanding one of the primary objectives of AI is to make computers more useful in problem solving. Three main AI methodologies are overviewed here: beam search, constraint satisfaction approaches, and neural network methods. Many other AI techniques have been applied to JSSP, such as a Vibrating Potential method and an Ant System optimisation method; however their effect has been limited.

4.3.2.3.1 Constraint propagation

Several heuristics recognise that all constraints cannot be satisfied simultaneously, so they identify interactions among constraints and modify constraints selectively to reduce disparity among them, and thus reduce the number of constraints that must be examined. Constraint satisfaction techniques aim at reducing the effective size of the search space by applying constraints that restrict the order in which variables are selected and the sequence in which possible values are assigned to each variable. The constraint satisfaction problem is solved when a complete allocation of variables is specified that does not violate the constraints of the problem. Although considered within the domain of AI, many constraint based scheduling methods apply a systematic tree search and have close links with branch and bound algorithms.

When some of the recent constraint satisfaction techniques have been tested against a number of benchmark problems they achieved only solutions of adequate quality. Also, because close similarities with branch and bound methods, they were extremely costly. Some studies demonstrated that the incorporation of other (either heuristics or exact) techniques into the constraint satisfaction method improves results. In general, by incorporating more problem specific information, these techniques have a potential to produce better results and most importantly reduce computing times. It can be concluded that much research has yet to be done if good constraint satisfaction approaches are to be developed for JSSP [Jain & Meeran, 1999].

4.3.2.3.2 Neural networks

Neural networks are organised in a framework based on the brain structure of simple living entities. In these techniques information processing is carried out through a massively interconnected network of parallel processing units. Their simplicity, along with their capability to perform distributed computing, as well as their propensity to learn and generalise has made neural networks a popular methodology, allowing them to be used in many real life applications [Zhang & Huang, 1995].

Reported studies indicate that only the searching technique of [Sabuncuoglu & Gurgun, 1996] provides adequate results to benchmark problems [Jain & Meeran, 1999]. As these models are encoded using a mathematical model they suffer from a requirement of excessive numbers of constraints, variables and interconnections, hence they can deal with small problems only and as they are often trapped in local minima they do not guarantee optimal solutions. In addition for several of the methods the problems have to be of a particular dimensionality in order to be solved (e.g. n has to equal m) and the system can malfunction if it is not applied to problems for which it is designed for. Consequently, neural networks are not currently considered to be competitive with the best heuristics for any class of optimisation problem [Osman and Kelly, 1996].

4.3.2.3.3 Distributed scheduling

Distributed scheduling is yet another approach which has gained significant attention of research community in the last 25 years. Distributed scheduling tries to incorporate many real-life and up to date parameters/constraints that reflect current circumstances on the shop floor and, therefore, to deal with the scheduling problem on much a bona fide basis. Distributed scheduling is introduced in Section 4.5 "Emergence of distributed scheduling".

4.3.2.4 Bottleneck based heuristic

The shifting the bottleneck procedure is a well designed, analysed and implemented procedure which has been incorporated in many other works. The strategy of the shifting bottleneck procedure of [Adams et al., 1988] involves sub problem identification, bottleneck selection, sub

problem solution, and schedule reoptimisation. The JSSP is relaxed into *m* one machine problems which are iteratively solved one at a time (the branch and bound algorithm is used when searching for the optimum). Each one-machine solution is compared with all the others and the machines are ranked on the basis of their solution. The unsequenced machine having the largest solution value is identified as the bottleneck machine. Shifting bottleneck procedure sequences the bottleneck machine based on the machines already scheduled, with the remaining unsequenced machines ignored. Selection of the bottleneck machine is motivated by the conjecture that scheduling it at a later stage would deteriorate the performance criteria (makespan) of a system further. Every time the machine identified as the bottleneck is scheduled all the previously scheduled machines, susceptible to improvement, are locally reoptimised by solving the one machine problem again. The main contribution of this approach is the way the one machine relaxation is used to decide the order in which machines should be scheduled.

Shifting bottleneck procedure has improved the upper and lower bounds of several hard problems. A general weakness of these approaches is the level of programmer sophistication required and the whole procedure has to be completed before a solution is obtained [Jain and Meeran, 1999].

4.3.2.5 Local search methods and Meta-heuristics - Stochastic search

As a single technique cannot provide a satisfactory solution to job shop scheduling problem (JSSP) of reasonable size, recently, much effort has been concentrated on hybrid methods that combine myopic problem specific methods and a meta-strategy which guides the search out of local optima. Such hybrid techniques are known as iterated local search algorithms or meta-heuristics. These approaches, of which the most popular are based on techniques such as simulated annealing, tabu search, and genetic algorithms, currently provide the best results in solving JSSP. From these non-traditional optimisation techniques some breakthrough progress has been made in solving some of the more difficult job shop scheduling problems. A few comparative studies that have been performed with regard to meta-solvers are indicated in [Pirlot, 1996] (from his analysis genetic algorithms appear to be the weakest of these three methods both empirically and analytically). The main principles of these techniques are presented next.

4.3.2.5.1 Beam search

It is evident that PDR just choose one possible operation to add to the current partial sequence, while branch and bound techniques evaluate all possible operations, either implicitly or explicitly. The technique of beam search provides a balance between these approaches. Beam search evaluates only the most promising nodes at any given level. Only the best nodes are selected as nodes to branch from. The remaining nodes at that level are disregarded permanently. The number of nodes retained is called the *beam width* of the search. The results indicate that the beam search technique is able to improve on the myopic selections made by PDR. However, the deviations from optimum are still high, especially compared with other approaches, and with respect to the individual PDR the computing times are three orders of magnitude greater [Jain & Meeran, 1999].

4.3.2.5.2 The simulated annealing optimisation technique

Simulated annealing (SA) is a random oriented search technique that has gained wide attention in solving many combinatorial optimisation problems. The approach has been proposed by Kirkpatrick et al [Kirkpatrick et al., 1983]. Simulated annealing imitates the tendency of thermodynamic systems to seek energy minima. Applied to the job shop scheduling problem, for example, where the measure of performance is make-span, the simulated annealing approach

works as follows: Given a sequence, obtained from some other heuristic, a new sequence is generated by randomly interchanging two jobs. The new sequence is accepted if its make-span is better than that of the original sequence, otherwise (to avoid possibility to be trapped in local minima) it is accepted with some probability, which decreases as the process evolves.

The basic feature of the annealing approach is that the process is unlikely to get stuck at a local optimum. This is due to "the hill climbing moves" that are controlled by a selected parameter. The method is less and less likely to move away from the optimal solution towards the end of the process. The classical job shop scheduling problem has been addressed in many papers using simulated annealing. A reader seeking more information on this approach is referred to some of recent studies (and references there in). Examples include: [Kolonko, 1999, Steinhofel, et al., 1999].

4.3.2.5.3 The tabu search technique

Tabu search is now a well-established optimisation technique for solving combinatorial problems in many fields. Many papers have appeared recently dealing with the application of a tabu search to solve JSSP. A precise and complete description of this method is given in the papers of [Glover, 1989, 1990]. The basic idea of tabu search has been described in [Dell'Amico & Trubian, 1993] as follows. Any instance of a combinatorial optimisation problem is associated with a finite set of feasible solutions, each of which is characterised by a cost. The goal is to find a solution with minimum (or maximum) cost. Starting from an initial solution generated independently, a local search algorithm repeatedly replaces the current solution by a neighbouring one until a superimposed stopping criterion becomes true. The algorithm returns the best solution found, with respect to the cost function.

There are many papers in which tabu search has been used to address the classical job shop scheduling problems. Some of examples include: [Armentano & Scrich, 2000, Brandimarte, 1993, Mooney & Rardin, 1993, Watson et al., 2003, Widmer, 1991].

4.3.2.5.4 The genetic algorithm

The genetic algorithm is another strategy for managing a stochastic search that copies the biological paradigm of mutation and natural selection. The detailed description of the genetic algorithm is given in the book by Goldberg [Goldberg, 1989]. A summary of the basic idea of a genetic algorithm is presented as follows:

(1) Start with a set of initial feasible solutions that can be generated randomly. This set is called a *population*. The length of the set should be an even number. Evaluate the population by evaluating each solution and storing the value of the best.

(2) Generate another population by first randomly *mating* the solutions (i.e. each two solutions will become *mates*), then apply a crossover procedure for each mate. The crossover is done by randomly generating two integer numbers (*I* and *J*). *I* is in the range from 1 to *n*, and *J* is in the range from *I* to *n*. The operations in the positions from *I* to *J* are switched for each mate.

(3) A Partially Mapped Crossover (PMX) is then applied to correct any infeasible sequences (in the sense of repeating or missing operations) that might result.

(4) Apply the evaluation criterion explained above to the new population and update. Repeat until no substantial improvement is made.

Some of the recent papers where a scheduling problem is addressed by using the genetic algorithm are: [Biegel & Davern, 1990, Cheng et al., 1999, Davis, 1985, Madureira et al., 2001, Ponnambalam et al., 2001].

4.4 A review of some simulation studies related to traditional scheduling techniques

In this section we review several simulation studies with the aim to identify and extract those methods which were reported that contributed significantly to the increase in the overall performance of manufacturing systems. Intensive research has been conducted to address issues related to scheduling problem using modelling and simulation techniques. For example, simulation is used for addressing some primary research questions such as: Which dispatching rules for sequencing jobs at workstations perform best for different classes of criteria (performance measures)? Are some rules better than others for some classes of problems? Studies can be classified in three groups: one machine case, two and three machine case, and those that consider entire systems.

4.4.1 The one machine case

Research on single-machine scheduling has been largely based on the static problem, that is, how to best schedule a fixed set of jobs through a single machine when all jobs are available at the start of the scheduling period. Without delving into one-machine scheduling research, the most important conclusions that can be drawn from the research into the sequencing of jobs on one machine are [Vollmann, et al., 1992]:

- If the objective (system criterion) is to minimise average time per job, to minimise average number of jobs in the system (minimise work-in-process inventories), or to minimise average job lateness, the shortest processing time (SPT) rule represents the best way to pick the next job to run.
- If the objective is to minimise either the maximum lateness of any job or the lateness variance, then jobs should run in due date sequence.

4.4.2 The two and three machine case

Developing the two-machine case scheduling procedures is, as expected, more complex. Job routings have to be considered because the two machines must be scheduled, the minimum make-span depends on job sequencing, and moreover, *"if total time to run the entire batch of jobs is to be minimised, this doesn't ensure either the average time each job spends in the system or the average number of jobs in the system will also be minimised."* [Vollmann et al., 1992]. For analytically based research, some assumptions, similar to those for the one-machine case, have been made. For example, it is assumed that each job always goes from one particular machine to another particular machine, all jobs are available at the start of the schedule, and set-up times are independent of the order in which jobs are undertaken. Without providing a detail review in the research of the two and three machine-problems, several important observations can be made.

- The size of problems we can treat with analytical methods is small and is of limited applicability in the "real world."
- The computer time required to solve scheduling problems with analytical methods grows exponentially with the number of jobs and/or machines to be scheduled.
- The performance measure, minimising the make span, is not the same as minimising average time in the system or average number of jobs in the system.
- The static scheduling assumptions (beginning with all machines idle, all jobs available, ending with all jobs processed and all machines idle) clearly influence the results.

- The machine processing times do not reflect randomness, which could reduce the techniques' applicability.
- The application of SPT (shortest processing time) rule in the two and three machine case is not exactly the same as it was in the single-machine case, but it is clearly an essential element in producing the desired scheduling performance in both problem situations.

4.4.3 Scheduling in Job-shop Flexible Manufacturing Systems

Scheduling problems in Flexible Manufacturing Systems (FMS) are far more difficult than for job shops. Some of the reasons for this are as follows:

- Each machine is capable of performing many different operations giving rise to a larger number of decision variables;
- The space available for storing unfinished parts between machines is limited;
- There is a need to synchronise machines and the material handling system;
- Scheduling has to encompass support equipment such as pallets, fixtures and tools.

Given these differences, the methodology and techniques developed for job shops may not be appropriate for an FMS. However, because developing shop-floor systems for scheduling day-to-day operations in a flexible manufacturing system is still being researched, scheduling such systems can be approached by using many of the scheduling concepts already discussed in this chapter. Indeed, in practice many FMSs are operated using dispatching rules (discussed in Section 4.3.2.2 "Priority Dispatch Rules (PDR)") that specify which job is processed next from the queue of jobs requiring processing. In a similar way, a scheduling problem can be viewed as either:

- A static scheduling problem, where a fixed set of orders are to be scheduled, either using optimisation or priority scheduling heuristics; or alternatively as
- A dynamic scheduling problem, where orders arrive periodically for scheduling (for example, as daily order releases from an MRP/MRPII (Materials Requirements Planning/Manufacturing Resources Planning) system or as individual customer orders). As a static scheduling problem, performance criteria such as minimising the make-span or minimising the mean order flow time may be of interest, while additional criteria, such as the mean and variance of the order lateness and tardiness, may be of concern in the dynamic version of the problem.

This section considers how the FMS performance can be influenced by the choice of the scheduling method used to make these decisions at the shop-floor level. It is assumed, therefore, that the higher-level decisions have been made. Furthermore, it is assumed that other parameters that control the level of work-in-process (such as number of material handling fixtures for parts, and the use of special or general purpose part holding fixtures) and therefore can impact FMS performance have been addressed and determined.

Research in studying the performance of various scheduling rules in FMS really began at the beginning of eighties, as a result of the increased number of FMS in use at that time.

Mosier et al. examined the performance of sequencing heuristics for the job shop manufacturing cell that contains four machines, each machine having three queues of orders, with a separate queue for each work-part family [Mosier et al., 1984]. Orders arrived at the cell at random time intervals, according to the Poisson distribution, and the due dates for the cell were assigned on the basis of Total Work Content, using the TWK procedure. The study indicated important differences between different sequencing rules applied to manufacturing cells. Also, one clear

principle emerges, namely that *important set-up savings can be achieved by taking family groupings into consideration in scheduling* work and while designing manufacturing cells.

A study by [Shanker & Tzen, 1985] reported some early work on the methods for planning workloads in a random FMS – a system designed to process a large variety of parts that arrive continuously at random time intervals. Their work considered planning balanced workloads at different machines in an FMS when setting due dates for jobs to be processed in an FMS. They evaluated methods for planning FMS workloads in conjunction with four different rules for dispatching jobs to machines: First Come First Served, Shortest Processing Time, Longest Processing Time, and Most Operations Remaining first. The reported simulation experimental results indicated that *two dispatching rules (Shortest Processing Time and Most Operations Remaining first) performed effectively when using machine utilisation as the performance criterion.* Furthermore, FMS performance improved substantially with a workload planning procedure that achieved a balanced workload between machines. Shanker & Tzen's work *confirmed the effectiveness of the SPT (shortest processing time) rule and indicated this rule's robust nature under a wide variety of production process configurations.*

Denzier & Boe reported a different FMS scheduling approach in a study of a 16-machine plant [Denzier & Boe, 1987]. Instead of focusing on the dispatching of parts to machines, this study analysed different part loading procedures, determined the best number of in-process work-parts (fixtures), and took into account congestion in the material handling system. A material handling system, with 37 carts and 51 dedicated work-part fixtures, was used to move the eight prismatic work-parts through the system. Instead of a centralised buffer storage queue, their FMS had room for two incoming and two outgoing work-part fixtures at each machine, and room on the material handling system track for additional work-part fixtures awaiting assignment to a machine for the next operation on the route. A simulation model of the FMS was used to evaluate the work-part loading methods, using the minimum make-span and machine utilisation as the performance criteria. Several experimental factors were varied, including the work-part loading rule, the number of work-part fixtures in the system at any given time, and alternative routing flexibility for the work-parts. The timing rule for work-part loading used in these experiments was the Next Empty Fixture (NEF) rule, indicating that a new work-part was loaded whenever a completed work-part left the FMS. Work-parts were dispatched to the machines using the Least Work in Next Queue (LWNQ) rule, since small work queues normally existed at each machine. The simulation results indicated that the Smallest Proportion of Jobs Launched (SPJL) and the Next Empty Fixture (NEF) work-part loading rules performed significantly better than the other procedures, producing a shorter schedule length (minimum make-span) with higher machine utilisation. The results also indicated an interesting interaction between the two experimental factors: alternate routing flexibility and number of work-part fixtures in the system. *A 12% gain in machine utilisation occurred when a high alternative routing flexibility and a large number of in process work-part fixtures were used* in contrast to the opposite case of low flexibility and a small number of work-part fixtures. At the same time the results also indicated decreasing marginal improvement in the machine utilisation, as the number of in-process work-part fixtures increased.

Chandra and Tilavage presented another study that concentrated on the examination of combination rules [Chandra & Tilavage, 1991]. They studied a specific scheduling algorithm and compared its performance with four traditional rules. This study considered routing flexibility, shop loading, machine type preferences, criticality of orders, and two shop floor objectives i.e., maximising the work progress rate and minimising the number of tardy parts. They concluded that *combination rules might provide significant performance enhancements over traditional scheduling rules.* As with many previous studies, this study considered only a few part types.

[Hutchinson et al. 1991] examined the effects of decision-making policy and routing flexibility on FMS performance. They considered three decision-making policies (i.e., two off-line and one real-time) and 11 levels of routing flexibility. They found that off-line policies outperformed the real-time policy by 30-34% with respect to make-span. They also showed that, as routing flexibility increased, the relative performance of the real-time policy deteriorated. Their off-line policies may have been favoured since parts were processed through a relatively small number of operations.

[Chen & Chung, 1991] evaluate loading formulations and routing policies in a simulated environment. *Their main finding was that FMS is not superior to a job shop if the routing flexibility is not utilised.*

Analysis reported by [Chang et al., 1996], who evaluated the performance of 42 priority dispatch rules using a linear programming model, indicates that the *shortest processing time (SPT) related rules consistently perform well* while the longest processing time (LPT) based rules consistently perform badly.

A simulation study of [Mahmoodi et al., 1999] examined the effects of scheduling rules and routing flexibility on the performance of a constrained, random FMS. The model of the system was composed of 8 machines (three machine types), three load/unload stations, 16 AGVs, an infinite central work-in-process buffer, a transfer station, and a computer-controlled network. Each work-part was randomly assigned to 1 of 120 work-part types, each with between six to nine operations. Work-part arrival rates were based on a Poisson distribution. The number of operations to be performed on each work-part was selected from a uniform distribution ranging from six to nine operations. Routings were not fixed in all cases, creating a mechanism for varying work-part routing flexibility. No tooling or pallet constraints were considered. Machines were allowed to process only one work-part at a time and could not perform any operations during loading/unloading or when a machine breakdown occurred. Due dates were assigned by the total work content (TWK) rule. All material movements not using the AGV system are assumed to be negligible (i.e., travel between the pick-up and drop-off points, buffers, within a machine). Transfer and load/unload times were assumed to be deterministic with the AGV velocity of 100 feet per minute and a load/unload time of two minutes. They compared the performance of four scheduling rules against three performance measures: average flow time, average percentage tardy, and average tardiness. Their results showed that the impact of the *choice of scheduling rules depended heavily on the level of routing flexibility present in the system and the performance measures considered.* In the presence of total routing flexibility (processing could take place at any machine of the same type) the effects of shop load, system breakdowns, shop configuration, and scheduling rules were significantly dampened. They showed that when total routing flexibility existed, the choice of scheduling rules was not critical although, *the existence of routing flexibility enhanced the performance of all scheduling rules, especially that of SPT.* Furthermore, they found that under most experimental conditions, the shop congestion factor has little or no impact on system flow-time performance. Finally, their results indicate that the behaviour of scheduling rules in a more constrained FMS environment (i.e., where system breakdowns occur and material handling capacity is limited) is very similar to those of less constrained environments.

Taken together, these studies provide an overall view of the impact on performance when different scheduling procedures and criteria are used in FMSs. Using various lateness criteria, simulation results shows that the Slack Time per Operation (ST/O) rule is best in terms of reducing the percentage of jobs late. The Shortest Processing Time (SPT) rule is a close second, and it is considerably better than the Earliest Due Date (EDD) rule. The EDD rule has a much lower variance of job lateness than SPT, but the average lateness measure of SPT might be more

than enough to compensate for the high job lateness variance associated with SPT. For the criteria "average time in the system" (which is directly related to average number of jobs in the system, work-in-process inventory, and average job lateness) the SPT rule performs quite well. However, as mentioned earlier, SPT can allow some jobs with long processing times to wait in queue for a substantial time, causing severe due-date problems for a few jobs. This is way the combinations of the SPT rule with other due-date oriented scheduling rules have been used in some approaches to obtain most of SPTs benefits, for example, by employing hybrid rules that are made up of SPT in combination with the ST/O or CR (Critical Ratio) rules. STP in combination with the FCFS (First Come, First Served) rule has also been used at periodic intervals to "clean out the workstations", so that no jobs can get "lost" due to long processing times at some operations. Development and implementation of hybrid rules is typically found in the firms with complex job shop processes. In conclusion, simulation experiments have consistently demonstrated that the *Shortest Processing Time is a very good dispatching rule for many performance criteria since it reduces lead times and work-in-process inventories.* This conclusion is robust and is supported by a variety of simulation studies Also, the results of studies indicate that *existence of routing flexibility and use of combined scheduling rules might significantly enhance the performance of FMS.*

4.4.4 Concluding comments on classical scheduling research

The current state of scheduling research is well described in a recent study by Mahmoodi et al.: *"Most of the literature to date is limited in scope. That is, basic assumptions or simplifications have been made to examine various aspects of the scheduling problem. Very little has been done to combine the approaches of several researchers and examine the effects and interactions of many realistic factors."* [Mahmoodi et al., 1999].

However, based on the scheduling concepts and the literature review partially described in this chapter, a few general principles can be drawn:

- The objectives to be achieved in scheduling must be determined before selecting a scheduling approach, since different approaches produce different results for different optimisation objectives.
- On a basis of overview of scheduling techniques presented here it can be concluded that (in spite of some breakthroughs that have been achieved recently) a "satisfactory solution" for the job shop scheduling problem has not be found. The studies reported by [Jain and Meeran, 1998, 1999] provide a comprehensive review and comparison of the various scheduling techniques applied to job-shop scheduling problem (only some of which are mentioned in this overview). Their studies indicate that the best results (with respect to time and solution quality) are achieved from the shifting bottleneck algorithm and hybrid meta-heuristics involving tabu search, genetic local search, and simulated annealing approaches. These methods equal and surpass the best solutions for the majority of available benchmark instances by combining several generic and problem specific techniques. However, since many constraints and considerations related to scheduling problem have not been addressed (some of them are discussed in Section 4.2.1.1 "Why the scheduling problem is difficult?") the aforementioned scheduling approaches produce results that are still far away from an overall satisfactory solution to job shop scheduling problems (refer to Section 4.5 "Emergence of distributed scheduling"). Job shop scheduling remains a very active and challenging research field.
- The modern shop floor in particular is so complex that many scheduling algorithms take too much time to run. To cope with the complexity of the scheduling problem, the problem needs to be decomposed (partitioned) into smaller units. However, a problem

arises here as to whether the individual optimum solutions create an overall optimum solution. Negotiation is one particular form of partitioning [Parunak, 1991] and it is used in this project.

- There is a major difficulty in comparing different scheduling rules and scheduling schemes due to the lack of standardisation among system definitions [Gunasekaran et al., 1995]. Results usually depend on the system configuration and cannot be generalised to other systems.
- The Shortest Processing Time (SPT) sequencing rule and its combination with other rules (e.g. SPT/TOT) can produce an effective overall system performance and should be considered as one of the "standard" rules in scheduling shop-floor systems. Merits of SPT based rules are supported by many other studies, including [Chandra & Tilavage, 1991, Chang et al., 1996, Hutchinson et al., 1991, Mahmoodi et al., 1999, Shanker & Tzen, 1985, Stecke & Solberg, 1981, Vollmann et al., 1992,].
- Introducing flexibility into scheduling (e.g., through alternate routings and overlap scheduling) can be used to gain important improvements in manufacturing performance. [Denzier & Boe, 1987, Hutchinson et al., 1991, Mahmoodi et al., 1999, Tsubone & Horikawa, 1999].
- Manufacturing lead-time estimates in randomised FMS should be based on total work content and due dates must be maintained promptly (on a daily basis or ideally each time when deviations from the planned production occur) to provide improvements in manufacturing performance [Wemmerlov & Vakharia, 1991].

4.5 Emergence of distributed scheduling

A major shift in direction in scheduling research has occurred in recent years. This shift is related closely to major current changes in the design of production process at many firms. These changes were motivated by the expansion of new process technologies, such as those incorporated into Computer Integrated Manufacturing (CIM) systems, and by the intensity of worldwide competition in manufacturing. As a result, the recent scheduling research has encompassed the development of new concepts and techniques in attempt to increase the performance of modern but more complex manufacturing systems. This section introduces a distributed, dynamic, real-time scheduling method that is applied (through the concept of Multi-Agent System) in job shop manufacturing.

As discussed previously, in real life situations, the complexity and dynamics of a manufacturing system make traditional scheduling methods too inflexible, costly, and too slow to satisfy the needs of real world scheduling systems. *"The intrinsic complexity of modern manufacturing systems prohibits the use of comprehensive analytical models. Even worse, if satisfactory solutions could be achieved, they would be valid only for a very short time"* [Parunak, 2000a]. For these reasons, more and more researchers have begun to try solving the complex scheduling problems in other ways. Because the inherent nature of many industrial processes is distributed, it suggests that the complex scheduling problems should be approached in a distributed way. The main idea behind the procedure for achieving distribution is based on the following:

Firstly, the total resources and jobs are divided into local groups, and their interconnections are temporarily cut off. The resulting individual (distributed) problems are smaller and may be able to be solved by traditional methods. These "distributed" solutions are then synthesised into a "total" solution and possible conflicts between local solutions are detected and solved. An example of possible conflict that may arise is when two local solutions result in two jobs competing for one machine at the same time. From the synthesis one feasible solution to the

whole problem is obtained. The problem still remains that a system of local optimisation may not be the overall optimum solution but at least the problem is solvable with a considerable degree of optimisation.

Distributed scheduling systems are reminiscent of a real factory, where an operator is responsible for several jobs or resources. In this case, the operator can be viewed as ***an agent*** (or a holon) for these jobs and resources. The definition of an agent and its properties are discussed in Chapter 6). The agent has responsibilities to his or her own client, but is also willing to communicate and negotiate with other agents by keeping the goals of the whole organisation in mind. It is this "agent" approach, which has led to the birth of a new scheduling methodology – ***multi-agent scheduling***. Multi-Agent Systems mimic the judgement and intelligence of real world human agents, while their relatively independent problem solving and communication with one another can take advantage of the power of distributed computation. As a result multi-agent scheduling has developed quickly in the recent 10 years, paralleling the development in low cost distributed computer networks.

The proposed distributed heterarchical shop floor control system, of which a core but functional part was developed in this thesis, is based on the paradigm of Multi-Agent Systems. Chapter 6 is, therefore, dedicated entirely to the review on research work conducted in the area of Multi-Agent Systems. Section 6.3.5 "Scheduling in distributed multi-agent heterarchical control systems" in that chapter provides more information on scheduling in distributed multi-agent heterarchical control systems.

Chapter 5. The new generation of manufacturing systems

This chapter discusses some new techniques and technologies that have emerged recently in domain of manufacturing systems in response to these changes. The chapter is organised as follows:

Market trends and changes in manufacturing systems are addressed in Section 5.1.

Section 5.2 briefly introduces concepts of virtual organisations

Main requirements imposed on modern manufacturing systems are summarised and discussed in Section 5.3.

An overview of most important manufacturing control architectures with their main characteristics is presented in Section 5.4.

Agent technology is briefly introduced in Section 5.5. (The entire next chapter is dedicated to matters related to multi agent systems).

The main principles of the holonic, bionic, and fractal manufacturing systems and comparison of these concepts are given in Section 5.6. This section also reviews some selected research projects related to holonic based manufacturing.

Section 5.7 discusses similarities and differences between multiagent and holonic manufacturing systems and their integration.

Section 5.8 summarises design principles for highly distributed manufacturing systems.

5.1 Introduction

Due to creation of global, open, and highly competitive markets, manufacturing industries are currently facing a continuous change from a relatively stable supplier society (mass production) to a customised society (mass customisation). "One-size-fits-all" model is being replaced by the "one-of-a-kind" production model [Agility-forum, web]. To build an agile manufacturing system is not easy because it requires development of an entire new set of business practices as well as conducting radical changes across the whole organisation.

Another, much larger group of changes is consisted of those methodologies whose scope is mostly limited to a single organisation. In addition to many concepts that have been envisaged recently to improve the performance of manufacturing systems (such as lean manufacturing, total quality management, concurrent engineering, etc.) the changes caused by these group of methodologies have been spurred lately by an idea of adopting more decentralised, heterarchical manufacturing control architectures. Changes have been driven by the need of companies to provide less rigid and less complex manufacturing control structures that would enable rapid and inexpensive design, implementation, reconfiguration, resize and maintenance of manufacturing facilities.

5.2 Virtual organisations

Before proceeding further with the matters related to the development of modern manufacturing control systems, the concept and the development of virtual manufacturing organisations is briefly addressed in the next section.

The traditional enterprise comprises most of the phases needed to provide a product (or service), from orders through design, production to marketing. The majority of these enterprises had the policy of directly controlling all the phases of business processes, within the enterprise using internal resources. Today, many manufacturing enterprises cannot any longer be seen acting stand-alone in the global, open economy. To meet the new challenges and to reduce associated risks and costs (which are sometimes prohibitive), companies need to be more focused on their core competencies and acquire knowledge and service in non-fundamental domains from other companies by forming strategic partnerships or alliances [Sousa, et al., 1999]. On the other hand, tremendous technological developments in the area of information and networking technologies have enabled global connections that were unthinkable before. For example, the Internet is linking our world, enabling partnerships which would otherwise be impossible in all areas. Supported by these recent technological advances enterprises have started to introduce new, more dynamic and proactive, concepts of organisations. Two main paradigms have emerged: 1) the Web-Centric paradigm and 2) the Virtual Enterprise (VE) paradigm. They are briefly summarised next.

5.2.1 The Web-Centric paradigm

The Web-Centric Enterprise [Hornberger, 2001] is a novel business model that provides a foundation on which companies can build processes and procedures in such a way as to be able to accommodate unique customers requirements for products at the initial implementation of the system and also over the product life cycle. Web-centric organisations use communications extensively, but not in a way that is critical in fulfilling the goal of the organisations (e.g., a multinational corporation with dispersed parts being on the same satellite network as related companies whose use, however, is not critical for completing the production process). At the core of the Web-Centric Enterprise model is the internet-enabled software infrastructure acting as a worldwide open Dynamic Service Environment (DSE) (See: www.agentcities.org). Such an integrated framework enables the sharing of information, services and applications among suppliers, employees, partners and customers via:

- Deployment of automated, intelligent software services (e.g., internet-enabled negotiations, financial transactions, advertising and bidding; order placement / delivery, etc.)
- Complex interactions between such services (e.g., compliance policies; argumentation and persuasion via complex conversation protocols, etc.);
- Dynamic discovery and composition of services to create new value added services (e.g., dynamic virtual clustering of synergetic partnerships of collaborative organizations aiming to achieve a common goal).

5.2.2 The Virtual Enterprise paradigm

The Virtual Enterprise (VE) is a paradigm that can be defined as a temporary alliance of enterprises that come together to share skills and resources in order to better respond to business opportunities [Camarinha-Matos et al., 1997]. The term "virtual enterprise" is used because it would not be a permanent organisation. VE is a set of enterprises opportunely joined in a

logical, momentary although operational manufacturing structure, co-operating to accomplish particular, predefined and mutually accepted goals. Such enterprises comprehend multiples autonomous entities, each potentially composed by other autonomous inter-related co-operative enterprises, behaving globally as single one. Each autonomous entity tends to focus its core competencies outsourcing complementary products or even actively pursuing cooperation partners to mutually complement their activities [Camarinha-Matos & Afsarmanesh, 1997], expecting to reduce risks and costs while maximizing market opportunities. This kind of enterprises "lives" in a very dynamic and challenging environment. A new market opportunity might trigger the creation of a complete new logical enterprise structure providing the market with a required new product. An example of the Virtual Enterprise paradigm is the Boeing Company that established a virtual co-operation with several companies in order to produce its aeroplanes. Boeing designs, assembles and markets the aircraft, while an international network of suppliers makes the components. Another organisation which operates essentially in a virtual organisation manner is the worldwide Toyota automobile manufacturing operation.

As with a web-centric enterprise, a virtual enterprise is one whose members are geographically apart, independent entities while appearing to be operating as a single, unified organisation with a single and apparently real physical location. Within a virtual organisation work cannot be completed without support of an information technology infrastructure linking the organisation's parts. However, virtual organisations, contrary to web-centric organisations, use communications extensively in a way that is critical for completing the production process and fulfilling the goals of the organisations. Scheduling in a virtual enterprise poses many challenges but it was believed that the multi-agent model proposed in this study could be used not only within a single organisation but in specialised manufacturing entities linked through the Internet.

5.3 Main requirements of the new generation of manufacturing control systems

Modern manufacturing strategies need to shift to support global competitiveness, new product innovation and introduction, and rapid market responsiveness. The next generation manufacturing systems will thus be more strongly time-oriented, while still focusing on cost and quality. Because modern manufacturing depends heavily on computer systems, all requirements (driven by consumer demands / market trends) that are imposed on modern manufacturing systems apply to the manufacturing control software and the computer control system itself. Conventional manufacturing control systems have experienced substantial difficulties in coping with the new circumstances on the shop floor (increased complexity, frequent changes, unpredictably and uncertainty). The control systems required additional functionality. From the analysis of different opinions (recent examples include: [Baker, 1998, Bussmann & McFarlane, 1999, Shen & Norrie, 1999, Ulieru, 2002]) a list of most common characteristics (from shop floor to inter enterprise level) that the next generation of advanced manufacturing systems need to incorporate are given below. It should be noted that this list of requirements is not exhaustive, that several items overlap in their definition, and that most of listed features relate both to the overall system and to each individual entity.

Modern manufacturing control systems need to provide the following functionality:

- ***Reconfigurability*** – This refers to the ability of a manufacturing unit to be easily altered in a timely and cost effective manner. From the system perspective, reconfigurability is characterised by run time accommodation to the addition or removal of various components of a manufacturing system. The capacity of the control architecture needs to

be able to reflect changes in the system configuration reliably and quickly (preferably on a daily basis).

- ***Modifiability*** characterised by the ease with which changes to existing elements of the system can be made. The terms modifiability and reconfigurability are often used interchangeably.
- ***Extendibility,*** characterised by the ease with which new elements can be added to the system to increase the existing level of functionality. An important functionality (related to maintenance) of an extendible system is some form of "plug & play". It should be possible to add or replace hardware and the corresponding software modules on the fly, without stopping the system.
- ***Adaptability*** is the ability of a manufacturing system to continue to operate in the face of disturbances, changing its properties and behaviours in response to new circumstances encountered on the shop floor. Disturbances might be caused by everyday production anomalies or by substantial structural changes that the system might encounter during its "life-time". In the later case, terms ***ill-specification*** and reconfigurability are used as synonyms for adaptability.
- ***Fault-tolerance,*** which is the ability of the system to continue to function, perhaps in a degraded state, despite the occurrence of system failures. The system should be fault tolerant both at the system level and at the subsystem level so as to detect and recover from system failures at any level and minimise their impacts on the working environment.
- ***Reliability,*** which is the measure of the probability that a system (or system component) will operate continuously, conforming to its specifications, without experiencing failure.
- ***Flexibility*** - the ability that the system exhibits during operation that allows it to change processes easily and rapidly (to meet primary disturbances in production) in a predefined set of possibilities (each one specified as a routine procedure), defined ahead of time. Flexibility also relates to physical flexibility of machinery. Machines (and other resources) have to be able to execute several operations and to change quickly among different production plans according to the part type to be manufactured at a given point in time.
- ***Self-Organisation*** is the ability of manufacturing units to collect and arrange themselves in order to achieve a production goal.
- ***Scalability*** means that additional resources can be incorporated into the organisation without disrupting organisational links previously established. This capability should be available at any working node in the system and at any level within the nodes.
- ***Robustness,*** manifested through the ability of manufacturing systems to respond rapidly and in a cost effective manner to uncertainties in the manufacturing process (e.g. defects, delays, and variable yields). Terms robustness and adaptability are used interchangeably.
- ***Reaction*** refers to the ability of an entity to adjust its plans according to its perceptions.
- **Enterprise integration:** In order to support global competitiveness and rapid market responsiveness, an individual or collective manufacturing enterprise will have to be integrated within its related management systems (purchasing, orders, design, production, planning & scheduling, control, transport, resources, personnel, materials, quality, etc.) and with its partners via networks.

- **Distributed organisation:** For effective enterprise integration across distributed organisations, distributed knowledge-based systems will be needed so as to link demand management directly to resource and capacity planning and scheduling.
- **Heterogeneous environments:** Such manufacturing systems will need to accommodate heterogeneous software and hardware in both their manufacturing and information environments.
- **Interoperability:** Heterogeneous information environments may use different programming languages, represent data with different representational languages and models, and operate in different computing platforms. The sub-systems and components in such heterogeneous environments should interoperate in an efficient manner. Translation and other capabilities will be needed to enable such interoperation or interaction.
- **Open and dynamic structure:** It must be possible to dynamically integrate new subsystems (software, hardware, or manufacturing devices) into or remove existing subsystems from the system without stopping and reinitialising the working environment. This will require an open and dynamic system architecture. Terms robustness, adaptability, reconfigurability and openness are often used as synonyms.
- **Cooperation:** Manufacturing enterprises will have to fully cooperate with their suppliers, partners, and customers for material supply, parts fabrication, final product commercialisation, and so on. Such cooperation should be in an efficient and quickly responsive manner.
- **Integration of humans with software and hardware:** People and computers need to be integrated to work collectively at various stages of the product development and even the whole product life cycle, with rapid access to required knowledge and information. Heterogeneous sources of information must be integrated to support these needs and to enhance the decision capabilities of the system. Bi-directional communication environments are required to allow effective and quick communication between human and computers to facilitate their interaction.
- **Agility:** Considerable attention must be given to reducing product cycle time to be able to respond to customer desires more quickly. Agile manufacturing is the ability to adapt quickly in a manufacturing environment of continuous and unanticipated change and thus is a key component in manufacturing strategies for global competition. To achieve agility, manufacturing facilities must be able to rapidly reconfigure and interact with heterogeneous systems and partners. Ideally, partners are contracted with "on the fly" only for the time required to complete specific tasks.

This list provides a test set of characteristics for any system which is designed to meet the challenges of the modern manufacturing environment.

5.4 Overview of manufacturing control architectures and their main characteristics

Manufacturing control systems' architectures have a significant impact on the system performance through its design, implementation, growth, and modernisation. It defines the manner in which the control is executed, the type and flow of information, and the interaction between various control modules of the system. Traditional control architectures in computer controlled manufacturing systems can be viewed through two basic control types: centralised and hierarchical.

5.4.1 Centralised control architectures

The first generation of computer controlled manufacturing systems relied on centralised control systems. Control decisions were made centrally and distributed to the production entities for execution. Centralised systems are characterised by strong master-slave relationships, with a large amount of global data located in a central database, and closely coupled entities. Therefore, a single central computer controlled all major events in the system. To optimise performance of the system a central computer was responsible for scheduling a number of production entities.

Having the features denoted above, it becomes easy for a centralised system (at the cost of significant complexity) to predict the behaviour of entities and to achieve global coherence of the system. In centralised systems, the "master" entity resolves all resource sharing conflicts. It has complete knowledge of the state of the system including the schedules of all entities, all variables, and all relations between these variables. In addition, since one central entity is in charge of directing the activities to all other entities, information for decisions about what to do next can be accumulated quickly. (Faster than in heterarchical control systems – see Section 5.4.3 "Heterarchical control architectures").

The centralised nature of the system brings many disadvantages. Centralised systems have a monolithic and complex structure and therefore they have low flexibility and adaptability. The control software is complex and must anticipate every circumstance that can arise in the system. Any reconfiguration in the system requires modifications in the control software that are difficult to make because of its complexity. Furthermore, the central computer is a bottleneck and can limit the capacity of the production system. Moreover, a single coordinating computer constitutes a single point of failure. If it is disabled then the entire control mechanism is brought down. Not surprisingly, this architecture is suitable only for small systems.

To overcome the drawbacks of a centralised architecture, the design and the control of large manufacturing systems are typically approached by decomposing the system into smaller, manageable modules. Decomposition and decentralisation of the system and distribution of control functions can be performed either in a hierarchical or heterarchical manner. However, note that in literature the term ***distributed*** is usually used to denote heterarchical architecture.

5.4.2 Hierarchical control architectures

The original control concept for computer controlled manufacturing systems lacked features such as distribution and decentralisation. Hence, more decentralised hierarchical manufacturing systems have been proposed as the next evolutionary step from the original centralised architectures [Sousa, et al., 1999].

The term "hierarchy" has generally been used to refer to a complex system in which each of the subsystems is subordinated by an authority relationship to the system it belongs to. Hierarchical architectures contain multiple levels of hierarchy in which most of the responsibility and authority are located in the upper layers, or "masters", and the lower layers, or "slaves", perform "menial" tasks. "Command" information flows downwards, and feedback information flows upwards resulting in strong master-slave relationships. The manufacturing hierarchical control architectures typically have 2-5 levels. In their pure form, a higher level always plays supervisor – master role to the subordinated entities at lower levels. Each entity in the system has a single co-ordinator that solves scheduling and control problems. Decisions at the higher level define the constraints for the next lower level. They are put forward to the lower levels for execution. If the lower control levels cannot satisfy the posted constraints, it is reported as feedback to the

higher level where a new subsequent set of actions is computed, etc. Tight coupling between masters and slaves entities results in short response times between them.

Advantages of the hierarchical architecture are reflected through: the ease in which masters co-ordinate activities of their slaves due to the tight coupling between them; the familiarity of system designers with this architecture; and improved fault tolerance (if a coordinating computer is disabled then only the branches below it, and the entities at the lower levels, are affected).

The hierarchical approach has some important disadvantages. Firstly, dynamic adaptive control is difficult to obtain. Hierarchical architectures are very rigid and give little support for evolution (components are coupled firmly among one other and therefore each part is strongly dependent on the rest of the system) [Dilts et al., 1991]. The organisation and structure of hierarchical systems are fixed in the early stages of their up-front design, so that all further developments rely on the initial architecture and that is why such systems are usually very difficult to modify and maintain (any unforeseen extensions usually require considerable amount of effort). Secondly, taking into account that control always moves from top to bottom, while flow of information goes up and down through hierarchy, if a coordinating computer is overloaded then it is impossible to receive timely feedback from the controlled entities. Finally, besides the fact that up-down flow of information can increase decision-making time, it may also deteriorate the quality of information because information is usually abstracted while going upward in hierarchy. It is interesting to note that large hierarchical organisations tend to have certain common traits: complexity, rigidity, stagnation and, in case of large changes in environment, catastrophic collapse [Prabhu, 2000]. If the lateral flow of information is allowed across hierarchy, performance can be improved but it leads to increased complexity of the system, and possible redundancy and inconsistency between duplicated information [Dilts et al., 1991].

5.4.3 Heterarchical control architectures

Traditional manufacturing control structures, both centralised and hierarchical, were efficient for intensive and repetitive processes. However, with the shift towards the "customised society", these features no longer represent an added value to the manufacturing enterprise [Sousa, et al., 1999]. Conventional systems become prohibitively complex and unreliable with the increasing size of the manufacturing systems. Increasing demands for agility in manufacturing are imposing limits on traditional manufacturing control architectures. Centralised and hierarchical approaches to manufacturing planning, scheduling, and control cannot cope with two major issues:

- The complexity of the scheduling problem (See Section 4.2.1.1. "Why the scheduling problem is difficult?"); and
- The increased dynamics of contemporary manufacturing systems (frequent and unpredictable changes).

It is generally considered that decentralised heterarchical structures offer better flexibility and adaptability than rigid hierarchical ones. Therefore, to overcome the disadvantages of conventional manufacturing control schemes, manufacturing computer control architectures have to be modified from the more rigid centralised and hierarchical structures to more decentralised, heterarchical architectures that are more adaptable to production requirements and the state of the manufacturing processes. The heterarchical approach is a natural way for modelling and controlling manufacturing systems that are comprised of spatially distributed production entities. This has caused growing interest among researchers in heterarchical shop floor control architecture [Baker, 1991, 1995, 1996, 1998, Brennan & William, 2000, Crowe & Stahlman,

1995, Dewan & Joshi, 2001, Duffie, 1982, 1990, Duffie and Piper, 1986a, 1987, Duffie & Prabhu, 1994, Duffie et al. 1988, Lewis, 1981, Lewis et al., 1987, Lin & Solberg, 1992, 1994, Macchiaroli & Riemma, 2002, Maturana et al., 1999, Ottaway & Burns, 2000, Parunak, 1987, 1998a, 1998c, 2000a, 2000b, Parunak et al., 1998, Roy & Anciaux, 2001, Saad et al., 1995, 1997, Shaw, 1985a, 1985b, 1987a, 1987b, 1988, Shen & Norrie, 1998, 1999, Veeramani 1992, 1994, Veeramani et al., 1993, 1997, Wang & Usher, 2002, Wu & Sun, 2002] just to list a few.

Agent technology (discussed in Section 5.5 "Agent technology") fits naturally into the decentralised control structure[1] for manufacturing systems because the autonomous component can easily be represented by an agent that is defined as an autonomous, pro-active element with the capability to communicate with other agents [Weiss, 1999]. Agents can be used to represent physical shop-floor components such as parts, machines, tools, and even human beings. In multi-agent systems, each agent is in charge of information collection, data storage, and decision-making for the corresponding shop floor component. A popular scheme to achieve cooperation among autonomous agents is through the negotiation-based Contract-Net Protocol. The Contract-Net Protocol provides the advantage of real-time information exchange making it suitable for shop floor scheduling and control (see Chapter 6).

In comparison with other architectures, the heterarchical architecture has some attractive features such as: lower development and maintenance cost, reduced complexity, increased flexibility, higher modularity, and higher level of code re-usability [Duffie & Prabhu, 1994]. Under the guidance of such a control architecture, the requirements imposed on modern manufacturing systems, such as: good fault tolerance and ease of modifiability, extendibility, reconfigurability and adaptability (refer to Section 5.3 "Main requirements of the new generation of manufacturing control systems"), can be achieved [Shen & Norrie, 1999, Wang & Usher, 2002]. Another advantage is the ability to "modularise and reuse the heterarchical elements or agents. Though heterarchical control systems would benefit any manufacturing facility, they are viewed as an effective strategy to improve manufacturing responsiveness particularly in small to medium volume, discrete part manufacturing operations. Baker claims that by implementing a heterarchical agent approach, "*the greatest benefit could be expected by manufacturers who need to change configuration of their factories often, who cannot predict the possible manufacturing scenarios in the future, and manufactures whose operations are significantly growing or shrinking*" [Baker, 1998].

There are several disadvantages of heterarchical control systems. Firstly, incomplete information and high local autonomy make it difficult to ensure that local decisions in a heterarchical system are globally optimised. Therefore, the main drawback of such a distributed approach is the lack of global coherence[2]. Also, it can be difficult to make these highly autonomous entities co-operate efficiently. The challenge is to avoid explicit co-operation among entities while making entities implicitly co-operate by modifying their local behaviour based on observations of the effect of their behaviour on lateness, work in progress, queuing time, and other parameters of the manufacturing systems in which they operate.

---

[1] One of the first heterarchical systems [Lewis, 1981] was realised on the bases of multi agent paradigm.

[2] An entity's action is said to be *globally coherent* if the action simultaneously satisfies the entity's and the overall system performance requirements. In centralised systems, the "master" entity resolves all resource sharing conflicts.

5.5 Agent technology

In response to the need for modelling the complexity of interactions in large scale distributed systems, agent technology has emerged from the Distributed Artificial Intelligence (DAI)[1] developments as a paradigm for structuring, designing and building software systems that require complex interactions between autonomous distributed (software) components [Ulieru, 2002]. Like any other technology, agents have certain capabilities, and are best used for problems whose characteristics require those capabilities. Five such characteristics are particularly significant. Agents are best suited for applications that are modular, decentralised, changeable, ill structured and complex [Parunak, 1998c]. As such, agent technology is especially attractive for distributed heterarchical manufacturing environments, in which overall behaviour depends on the interaction and co-ordination of the distributed elements. Agent technology provides the means to implement such distributed systems as a set of agents, which are autonomously acting software entities with capabilities to co-ordinate activities for creating a desired overall system behaviour [Sousa et al., 1999]. Agent manufacturing offers a promising way to model and control not only shop floor level activities but whole extended enterprises as well.

Agent technology has already being used in intelligent manufacturing for more than twenty years. However, the recent developments in Multi-Agent Systems have brought new and interesting possibilities. Thus, in the past ten years, researchers have been applying agent technology to manufacturing enterprise integration and supply chain management, manufacturing planning, scheduling and execution control, materials handling and inventory management, and developing new types of manufacturing systems such as holonic manufacturing systems (discussed in Section 5.6.3 "Holonic Manufacturing Systems (HMS)"). In distributed manufacturing systems, agents can be used to:

- Encapsulate existing software systems so as to integrate manufacturing enterprises' activities such as design, planning, scheduling, simulation, execution, and product distribution, with those of their suppliers, customers and partners into an open, distributed intelligent environment via networks (e.g. [Shen et al., 1998)];
- Represent manufacturing resources such as workers, cells, machines, tools, fixtures, AGVs, as well as products, parts and operations to facilitate manufacturing resource planning, scheduling and execution control (e.g. [Parunak et al., 1998b]);
- Model special services in manufacturing systems, such as: Agent Name Server for providing registration and administration services [Peng et al., 1999], Mediator Agents for facilitating communication, cooperation and coordination among other agents [Maturana & Norrie, 1996], Database Agents [Lin & Solberg, 1992] and Information Agents for providing information management; [McEleney et al., 1998];
- Incorporate a whole scheduler or planner into manufacturing planning and scheduling systems (e.g. [McEleney et al., 1998]); and
- Model holons which are software and hardware entities.

[1] The DAI literature differentiates between two classes of distributed computation problems: ***distributed problem solving*** and ***multi agent systems***. In both cases a computation is performed by distributed nodes which interact in a coordinated manner. In distributed problem solving the problem to be solved is formulated centrally, and then distributed to local computational nodes. In multi-agent systems the problem both originates and is resolved at the local nodes and the resulting overall solution is emergent [Jennings N., 1996].

A good discussion on agent technology for holonic manufacturing systems can be found in [Bussmann, 1998].

Additional information and more detailed discussion on Multi-Agent Systems are provided in Chapter 6.

5.6 Emergence of new paradigms

As indicated previously neither hierarchical nor heterarchical systems cope adequately with the new challenges imposed to manufacturing systems. Hierarchical systems typically have a rigid structure that impedes their ability to react to these disturbances in agile way. Heterarchical systems handle disturbances very well and can continuously adapt themselves to their environment. However, heterarchical control does not guarantee high performance or predictable behaviour. As a consequence, heterarchical systems have hardly been applied in industry. The actual challenge lies in the requirement that real life industrial systems need both performance and reactivity. The answer to this challenge was sought in the theories of complex adaptive systems. These theories are briefly reviewed in the following subsections focusing on their origins and fundamental features.

5.6.1 Bionic Manufacturing Systems (BMS)

The Bionic Manufacturing Systems (BMS) have developed under the ideas and concepts of biology. Based on observations of structures and behaviours in biological systems, the bionic approach assumes that the manufacturing companies can be built upon open, autonomous, co-operative and adaptive entities, which can evolve over time in response to environmental changes.

The basic structural unit in a biological system is a cell. Cells act as building blocks to make up the hierarchical layers in complex organisms (for example, a cell/tissue/organ/body hierarchy). Likewise, there are several levels of regulation in living organisms. *Enzymes* act as catalysts to speed up or inhibit reactions within a cell. *Hormones* exert specific physiological actions in a body. A third level of regulation, which deals with situations requiring rapid reactions to changes in the external environment, is represented by a *central nervous system.* In biological systems, each entity is responsible for its activities and self-division according to the genetic code stored in DNA[1]. Consistency and goal orientation are conceptually supported by the genetic inheritance. All bionic systems, from the simplest through to the most complex living forms, manifest the following key characteristics: hierarchically and heterarchically ordered relations, autonomy, spontaneous behaviour, and social harmony.

The above properties of biological systems have many similarities to the operation of manufacturing systems (see Figure 5-1). Like cells, manufacturing units obtain the needed inputs from the environment and perform operations. Outputs of these operations flow back to the environment. Also, manufacturing units can act as building blocks to derive hierarchical control structures, such as production lines, workshops, and factories. Like enzymes, *production coordinators* in manufacturing systems may act to preserve the overall harmony. Similar to hormones, regulatory schemes may include *policies or strategies* that have a longer-term effect on the environment, for example changes to shop floor practices. Like a central nervous systems, a *centralised control system* may be applicable to urgently react to certain contingencies.

[1] The DNA (deoxyribonucleic acid) stores the genetic information about individual.

BMS uses these parallels between biological and manufacturing systems to translate the structure and organisational behaviour of the living beings into modelling concepts and applications for manufacturing systems [Okino, 1989, 1992, 1993], [Ueda, 1993].

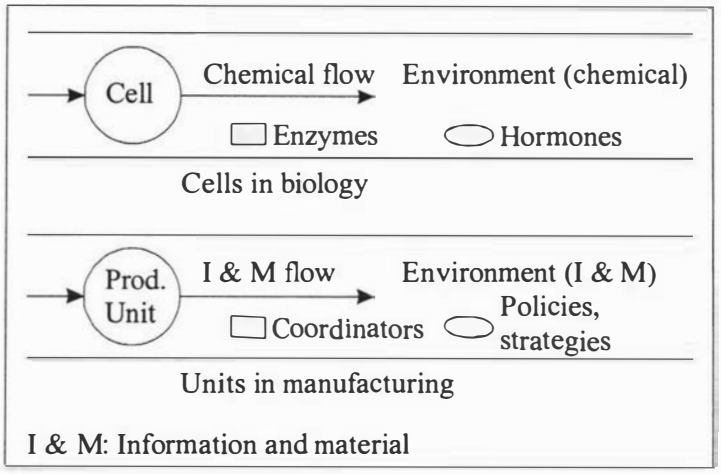


Figure 5-1. Similarity of Biological and Manufacturing Structures

[Tharumarajah et al, 1996]

5.6.2 Fractal Factories (FF)

The Fractal Factory is a paradigm that originated from the theory of mathematical chaos and its associated concept of Fractal Geometry proposed by Benoit Mandelbrot [Mandelbrot, 1982]. The Fractal Geometry provides tools for analysing and describing geometric, objects in multidimensional spaces. The word fractal comes from a Latin word ***"fractus"***, which means *broken* or *fragmented*. Each of these fragments or fractals contains the basic characteristics of the whole structure.

The concept of fractal factories [Warnecke, 1993] proposes that a manufacturing company is composed of small independent, self-similar components, called "fractal objects" or just ***fractals***. The main characteristics of fractals (fractal factories) are: [Sihn, 1997, Tharumarajah et al, 1996, Warnecke, 1993].

- ***Self-similarity***, which is interpreted as similarity of components inside fractals (each fractal contains a set of similar components), and as similarity of goals among the fractals to conform to the objectives in each unit. The explosion of fractal objects into other fractal objects has the particularity of generating objects which possess organisational structure and objectives similar to the original ones. Therefore, one of the implications of self-similarity is that each fractal must itself be a (little) "Fractal Factory".
- ***Self-organisation*** - means that fractals do not need external intervention to reorganise themselves. It implies fractals' have freedom in organising and executing tasks. Fractals may choose their own methods of problem solving.
- ***Self-optimisation*** - means that fractals take care of process improvements, continuously increasing their own performance.
- ***Dynamics*** - means that the fractals can adapt to influences from the environment without any formal hindrance of organisational structure.

- ***Constant evolution***. Fractals have the capacity to react and adapt quickly to the new changes in the environment so that, related to other fractals and the environment, they are in the constant "process of evolution".
- ***Project-oriented organisational structure*** – The fractal factory is based on a project-oriented organisation which is in contrast to the traditional function-oriented organisation. The factory will not have a predefined organisation, but a more or less static set of resources and a very dynamic set of projects/tasks (refer to the text below). Fractals are always structured bottom-up, building fractals of a higher order. Units at a higher level always assume only those responsibilities in the process which cannot be fulfilled in the lower order fractals. This principle guarantees teamwork among the fractals and also forces distribution of power and ability.

To function as a coherent whole and achieve systems goals, the factory fractals cooperate among themselves (See Figure 5-2). Cooperation is based on a negotiation scheme. As a new project is introduced into the system, each resource notices the new event and starts negotiation (with remainder resources) for the execution of each sub-project. Thus, each project initiates a very dynamic process, responsible for resource/sub-project binding, (each resource can belong to multiple projects and yet maintain its core competencies) leading to constant change in the enterprise structure and organisation. This process is very dynamic as the resource can decide its own behaviour and does not rely on a higher entity to do so.

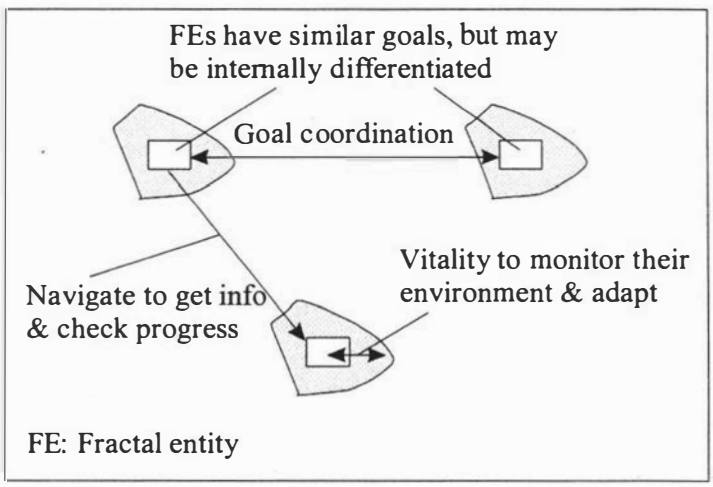


Figure 5-2. Operation of Fractal entities [Tharumarajah et al., 1996].

The operation of the system is characterised by ability of factory fractals to adapt and react to the influences of their respective environments. This ability is called ***vitality***, and it is used to record and evaluate some variables internal to the fractals that are affected by the environment. This information is used to measure the need for change against the characteristics of six specific levels (enterprise dimensions) of the work environment: cultural, strategic, socio-informal, financial, informational and technological, as well as providing an efficient navigation system [Tharumarajah et al., 1996]. Fractals navigate in the sense of constantly checking target areas, reassessing their position and progress, and correcting if necessary. The fractals may contemplate regrouping in response to changes in the vitality measures. The fractal factory must be understood as an open, non-linear system, structurally reactive and adaptive to the dynamic context [Sihn, 1997].

5.6.3 Holonic Manufacturing Systems (HMS)

5.6.3.1 Roots

The Holonic paradigm arises from Herbert Simon and Arthur Koestler studies of complex biological and social organisations. Simon [Simon, 1962] observed that complex systems are hierarchically structured and formed by intermediate stable forms. These forms allow the system to be stable, reliable and evolutionary, while maintaining a goal-oriented functionality due to its hierarchical structure [Wyns, 1999]. Later, analysing Simon's theory of hierarchies and stable intermediate forms and comparing it with his own observations, the Hungarian journalist and philosopher Arthur Koestler [Koestler, 1967] in his book "The Ghost in the Machine" introduced the concept of holonic systems to explain the evolution of complex biological and social systems [Ulieru, 2002].

While analysing living organisms and social organisations Koestler made two key observations: [McFarlane & Bussmann, 2000].

- These systems evolve and grow to satisfy increasingly complex and changing needs by creating stable "intermediate" forms which are self-reliant and more capable than the initial systems.
- In living and organisational systems it is generally difficult to distinguish between "wholes" and "parts": almost every distinguishable element is simultaneously a whole (a self-contained body to its subordinated parts) while concurrently being an individual member (a dependent and integrated part of a larger, more capable body).

Koestler's observation that although it is easy to identify sub-wholes or parts, "wholes" and "parts" in an absolute sense do not exist (everything can be part and whole at once), impelled him to name the basic units of organisation in real-life systems as "***holons***"[1]. The word "holon" is derived from the Greek word *holos*, meaning *whole,* and the Greek suffix *on* meaning *particle* or *part* (as in proton or neutron).

Starting from the empirical observation that, from the Solar system to the atom the Universe is organised into self-replicating structures of nested hierarchies intrinsically embedded in the functionality of natural systems, Koestler has identified structural patterns of self-replicating structures, named "***holarchies***". Holarchies have been envisioned as models for the Universe's self-organising structure in which holons at several levels of resolution in the nested hierarchy behave as ***autonomous*** wholes and yet as ***cooperative*** parts for achieving the goal of the holarchy [Ulieru, 2002]. In such a nested hierarchy (see Figure 5-3) each holon is a sub-system retaining the characteristic attributes of the whole system.

The entire holarchy was seen as a recursive hierarchy[2] or heterarchy of holons with no centralised control. Each holon combines its set of competencies with its partner holons, with whom it co-operates in order to achieve both individual and system goals. This suggests that the holon is an ***autonomous entity***, including operational features and individual goals, and a ***cooperative entity*** that communicates with other holons to solve individual or system problems. System goals and plans are partially defined in higher holons and going down the holarchy, they

[1] According to Koestler, a holon was an identifiable part of a system that had a unique identity, (e.g. a single cell in living organisms or a family unit in social organisations) yet was made up of more basic sub-ordinate parts (e.g. plasma and nucleus, or parents and siblings) while at the same time was a part of a larger whole (e.g. a muscle tissue or a community).

[2] Koestler observed that all complex systems have a form of hierarchical organisation.

are progressively refined. The self-reliant characteristic ensures that holons are stable, able to survive disturbances. The subordination to higher-level holons ensures the effective operation of the larger whole. The result is a highly complex but reactive system with minimal concession to efficiency. According to Koestler, the characteristics of holonic systems made them more stable and flexible than any other system created by humans.

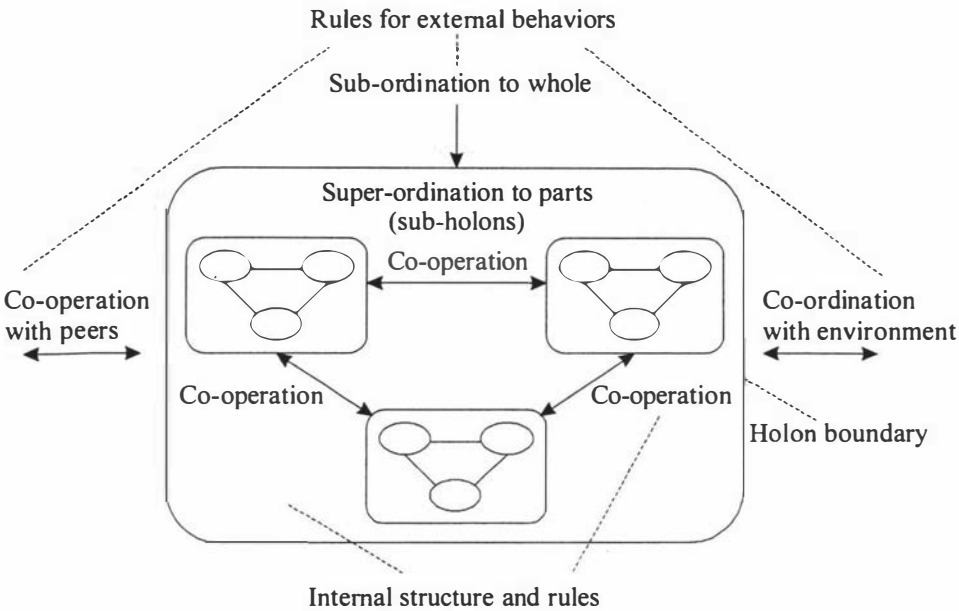


Figure 5-3. Generic model of a holarchy [Ulieru, 2002].

5.6.3.2 First steps in development of Holonic Manufacturing Systems

The field of Holonic Manufacturing Systems (HMS) was initiated in the early 1990's [Suda, 1989, 1990]. Suda suggested that the properties of holonic systems (outlined in the previous section) would be highly desirable for a new generation of manufacturing systems that are subject to increasingly stringent demands and faster changes. He, therefore, proposed a "building block or holon" based model for designing and operating elements comprising manufacturing processes (similar in concept to the one outlined in Figure 5-4).

In 1989 Professor Yoshikawa, then of Tokyo University, recognised that great benefits could arise by sharing the international expertise and the costs of research and development. Based on his proposal an international collaborative research program in manufacturing, called the ***Intelligent Manufacturing Systems (IMS)***, was initiated in 1990. The participating countries were Australia, Canada, the European Free Trade Association (EFTA) countries, the European Union (EU), Japan and the USA.

To find out whether international collaboration in developing new manufacturing systems is feasible and desirable, 31 partners from all regions of the IMS program initially conducted a feasibility study from 1992 until 1994. The IMS feasibility study program consisted of the six test cases - major projects:

- Clean Manufacturing In The Process Industry.
- Global Concurrent Engineering.
- Globeman 21: Enterprise Integration For Global Manufacturing Towards The 21st Century.

- Rapid Product Development.
- Holonic Manufacturing Systems: System Components of Autonomous Modules and Their Distributed Control.
- Knowledge Systematisation: Configuration Systems for Design and Manufacturing (Gnosis).

The success, in addition to the encouraging and promising results, of the HMS tests led to the formation of an international IMS project in 1994. A full-scale program on HMS was begun in late 1994 involving the majority of the participants in the test case with an expectation that it will last for the next 10 years. It was one of ten projects endorsed by the Intelligent Manufacturing Systems (IMS) Steering Committee in May 1995 [Kanchanasevee, et al., 1997]. Today more than 30 academic and industrial partners from the IMS regions participate in this international collaboration.

5.6.3.3 Holon architecture

Holons are seen as units of a manufacturing system that are autonomous: *"self-reliant units that can handle circumstances and problems on their particular level of existence without asking higher-level holons for assistance"*, but simultaneously they are cooperative: *"holons can also receive instruction from and, to a certain extent, be controlled by higher-level holons, that is, they can make and carry out decisions through mutual agreements and co-ordinated actions"* [Brussel at al., 1998]. Therefore, each holon must have the data necessary for deciding its actions, means of communicating with other holons, algorithms and procedures for negotiating and executing mutually agreed actions, and means of carrying out such algorithms, procedures, and actions, whether by manual or automated means [IMS International, web]. The general architecture of a holon is depicted in Figure 5-4.

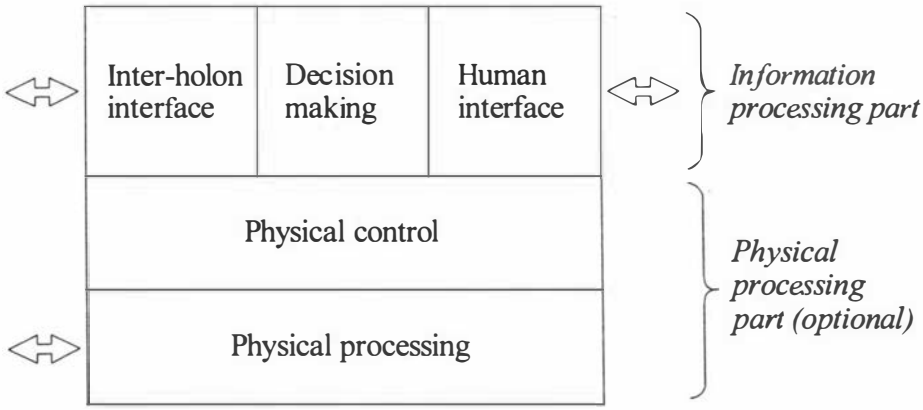


Figure 5-4. General architecture of a holon [McFarlane & Bussmann, 2000]

In the holon architecture the *inter-holon interface* and the *human interface* are responsible for communication with the outer world. In the case where a holon represents a physically existing resource (e.g. a robot) a third interface is needed to control the hardware. In Figure 5-4 this interface is denoted as the physical control. The *physical processing* represents the execution of manufacturing operations, like milling or assembly. The *decision-making* module is the kernel of a holon. It has the control over all interfaces; on the other hand the interfaces supply the decision-making unit with environmental information.

Since a holon represents a combination of an information processing part and a physical processing part, it means that a holon can represent either a physical or logical activity in manufacturing system. Therefore, a suitable combination of a machine tool, an NC controller, and an operator interacting via a suitable interface could form a "resource (machine) holon". (A

"workstation agent" as defined in this thesis can be seen as a "workstation holon"). On the other hand, a holon to handle an order could be an example of the holon that has only information processing part. Other examples of manufacturing holons include: robots, products, customer orders, information processing functions (such as a scheduling holon, or a negotiation holon), a Flexible Manufacturing System, or even a human operator.

5.6.3.4 Key characteristics of Holonic Manufacturing Systems

The key attributes of a holonic manufacturing system are: [IMS International, web]

- A holarchical (recursive) structure
- A flexible and dynamic architecture. Holonic systems should be able to operate in either or both a hierarchical[1] and a heterarchical manner according to the problem which has to be solved.
- Autonomous and cooperative units/elements. If one unit breaks down the system will go on, or will recognise the absence of one (say critical) unit and will stop the activity.
- Reusable and self-configuring elements.
- Holons may engage in multiple hierarchies at the same time.
- Holons should own the ability not only to cooperate with other technical holons, but with humans too. (Recognition of the important role for people in the overall success).
- A holon can be part of another holon. A holon can be broken into several others holons, which in turn can be broken into further holons, which allows the reduction of the problem complexity.

5.6.3.5 Goal of Holonic Manufacturing Systems

The long term goal of HMS is to provide means for development of more flexible, adaptive, agile, reliable, and less expensive manufacturing systems. HMS will be constructed of autonomous, cooperative, intelligent, and reusable manufacturing modules capable of reconfiguration automatically in response to new system demands and/or components. It is envisaged that these modules become equivalent to the "plug and play" information technologies of the personal computer [IMS International, web]. Through integration of such highly flexible modules HMS will be able to cope with change, disturbances (for which no predefined error handling strategies exist) and the uncertainty of manufacturing processes.

5.6.3.6 Some conclusions from the brief review of Holonic Manufacturing Systems

The strength of the holonic organisation, or holarchy, is that enables the construction of very complex systems that are nonetheless efficient in the use of resources, highly resilient to disturbances (both internal and external), and adaptable to changes in the environment in which they exist.

The HMS concept combines the best features of hierarchical (top-down) and heterarchical (bottom-up/cooperative) organisational structures. This concept can preserve the stability of hierarchy while providing the dynamic flexibility of heterarchies.

---

[1] The purpose of hierarchical coordination is to integrate the actions of lower level units, rather than to establish firm command and control relationships.

Holonic manufacturing is an approach to defining and specifying manufacturing production systems. It is a system engineering methodology rather than a solution to a specific control problem. In this sense it can be viewed as an alternative to more hierarchical operations management methods such as those based on Computer Integrated Manufacturing (CIM). Holonic manufacturing concepts try to reduce inabilities of the CIM technology for example to respond rapidly to breakdowns in the system. In fact, holonic manufacturing is referred to as a *bottom-up* approach because the overall plant control is developed through the integration of autonomous, cooperative, flexible, and interchangeable manufacturing modules. This is in direct contrast with conventional *top-down* methodologies for designing and specifying manufacturing control systems (like those found in CIM) in which a computer control systems hierarchy is centrally devised to support the planning, scheduling and shop floor control processes of a factory.

5.6.4 Comparison of concepts

Fractal Factory (FF), Bionic Manufacturing Systems (BMS) and Holonic Manufacturing Systems (HMS) are three paradigms that describe (in quite general terms) the underlying principles and features (as guidelines) for designing a new generation of manufacturing systems. It is evident that these concepts have a lot of similar or even overlapping characteristics. All these concepts can be utilised in proposing distributed and adaptive manufacturing systems with the recursive whole part relationships. Further, all these paradigms propose manufacturing systems that are composed of ***smaller, autonomous, and competitive entities***. All three suggest the idea that manufacturing systems, despite the increased autonomy required by individual entities, will ***continue to need a hierarchical structure*** in order to maintain the overall system coherence and objectivity, and guarantee inter-entities conflict resolution arising from the individual and autonomous attitude of the entities. Since the autonomous units pursue their own actions, to avoid the possibility of chaotic behaviour of the system all approaches indicate the ***use of regulatory mechanisms***[1] for control and coordination of their components. Other common points are: 1) co-operating and grouping entities; 2) dynamic nature of structures; 3) goal oriented grouping [Silva & Rocha, 2000]. These paradigms were proposed in order to provide the enterprise with standard and effective organisational and technological approaches, which does not mean that hybrid approaches could not be developed or combined, forming heterogeneous structures, exploiting the maximum benefit from each particular application.

However, the three paradigms have different origins, which imply different technical approaches in designing systems with the above identified common features. HMS and BMS tend to develop technology that will provide more flexible and intelligent forms of automation for the devices and equipment, focusing attention primarily on self-management[2] features of the units. On the other hand, the technology addressed by the Fractal Factory is more oriented towards applying principles of flexibility in layout, focusing more attention on the self-governing features of the units.

For more in depth comparison of the above concepts, the reader is referred to studies reported by [Tharumarajah et al., 1996] and [Sousa et al., 1999].

---

[1] A regulatory mechanism is released by modifying various activities of the component parts.

[2] Levels of autonomy of units range (in increasing order) from manager-led units, self-managing, self-designing, and finally, to self-governing units. In manager-led units the level of autonomy is restricted to the execution of tasks. At the other extreme, a self-governing unit assumes total responsibility including setting the overall direction. As the unit moves towards this end, its sphere of influence extends beyond the immediate control of the processes to other aspects (i.e. the human and business of the organisation).

5.6.5 Some remarks related to the review on new paradigms

The reviewed technologies provided the fundamental approaches which formed part of the ultimate theoretical basis for the approach used in this thesis. The thesis has been concerned with the practical implementation of an approach based on agents and low cost computing elements and it was not intended to test any of the theoretical concepts given above. However, the review was provided as an introduction to the thinking that led to the approach used in this thesis. Holonic Manufacturing Systems and Multi-Agent Systems are the most significant approaches for this project. It was believed that the following two concepts ***the event driven control strategy***, typical for holonic systems, and ***the distributed information processing,*** resulting in the Multi-Agent Systems, will soon make an important breakthrough in the field of intelligent manufacturing and control [Marik & Pechoucek, 2000].

5.7 Comparison between Multi-Agent Systems and Holonic Manufacturing Systems

The research communities working in Holonic Manufacturing Systems and Multi-Agent Systems fields approach the problem of "intelligent manufacturing" from different viewpoints and nearly independently (agent technology was firstly implemented to realise heterarchically controlled systems, and later on, the concept of holonic systems was introduced to address more specific requirements for manufacturing systems). They use their specific terminology and techniques. Both the paradigms share some ideas and they differ in the other issues.

The debate on clarifying the difference between holons and agents is an ongoing issue in the research communities using these paradigms. Given the essentially different path on which each concept was developed the question itself is inappropriate [Ulieru, 2002]. In the following we briefly present the main similarities and differences between the holonic and multi-agent paradigms.

5.7.1 Similarities between Multi-Agent Systems and Holonic Manufacturing Systems

The vision of a holonic factory draws a number of its concepts from the world of Multi-Agent Systems. That is why many similarities can be identified between these two areas.

Both the research communities do respect the same, very fundamental principles of holons' and agents' activities such as their ***autonomy***, ***cooperativeness*** and ***openness***. [Marik & Pechoucek, 2000].

- ***Autonomy*** –An entity (agent/holon) has the ability to operate independently of the rest of the system and possesses some kind of control over its actions and internal state [Ulieru, 2002].
- ***Cooperation*** – Through the cooperation process entities (agents/holons) develop mutually acceptable plans and execute them. [Ulieru, 2002]
- ***Openness/reconfigurability*** - integration of new systems or remission of existing systems can be achieved without stopping the process.

Both approaches provide most of the characteristics listed in Section 5.3 "Main requirements of the new generation of manufacturing control systems" (modifiability, extendibility, adaptation, fault-tolerance, etc.) and both, holons or agents, have ***multi-layered architectures***. [Marik & Pechoucek, 2000]

There are similar trends in standardisation which are quite evident with the IEC (International Electrotechnical Commission) 61499 standard in the case of holonic systems and the FIPA (Foundation for Intelligent Physical Agents) standard in the area of Multi-Agent Systems.

5.7.2 Differences between Multi-Agent Systems and Holonic Manufacturing Systems

The holonic system community is rooted in the concept of holons as presented by Koestler and is strongly ***driven by the requirements of industrial control***. The community is well organised around the international HMS (Holonic Manufacturing Systems) consortium. On the other hand, the comparatively much larger and more diverse community of researchers working in the Multi-Agent System (MAS) area is influenced by the ideas of highly distributed computing in computer networks as well as by the ideas of distributed artificial intelligence. As the community is much more heterogeneous, there are different organisational frameworks where the researchers are grouped. The European MAS researchers are organised in the AgentLink consortium, worldwide in IFMAS (International Foundation for MAS), Agent Society, and FIPA with an emphasis on industrial standards [Marik & Pechoucek, 2000]. Holonic and multi agent approaches differ in the following:

- **Differences in a concept origin and emphasis:** Holonics is an *organisational paradigm* inspired by the self-organising properties of natural systems. The emphasis is being on the *structure* of components rather then on the interaction between them. On the other side, agents have been envisioned as a *software paradigm* aiming to expand the limitations of the static object model with proactive capabilities of autonomy and environmental awareness. MAS aims to represent dynamical systems in software by focusing on the *interactions* between their parts - software components modelled as software agents - rather then on their structure [Ulieru, 2002].
- **Nature of systems elements:** In the manufacturing domain, holons have been defined as entities consisting of an information processing part and an optional physical processing part. On the other hand, a software agent is exclusively a software entity.
- **Interest:** The primary interest of HMS is in building a physical shop floor architecture, which is composed of co-ordinated and co-operating, interoperable and reusable hardware/software field components. MAS community has concentrated (until recently) on information agents only.
- **Differences in a nature of interactions among elements:** In a holonic system *cooperation* is a precondition for the existence of the holarchy per se. Cooperative interactions among holons bind the holarchy together driving it towards the achieving of common goals with maximum efficiency. On the other side, in a MAS there is no pre-assigned condition that the interactions among agents should be driven by cooperative forces. In a MAS agents may interact based on *competitive* rather than cooperative rules (such as electronic markets - competitive/conflicting environment) [Ulieru, 2002].

In addition to above differences, considering motivation, subject of research, usage of holarchy principle, and implementation of human interface, Marik and Pechoucek reported the following quite evident and distinguishable differences [Marik & Pechoucek, 2000].

- ***Motivation:*** The holonic research is motivated by pragmatic manufacturing control requirements, on the opposite side, the agent research is motivated by implementation of distributed computational systems and decentralised decision-making.

- ***Subject of research:*** The holonic system researchers are preferably oriented towards the low-level end of the manufacturing process, low-level communication and behavioural standards, integration, etc. Multi-Agent System researchers aim at implementing social behaviour of intelligent entities, cooperation and coordination strategies, intelligent brokerage, learning from ones own experience, teamwork and coalition formation, etc. From a very simple viewpoint, we can see the holonic system research stream providing platforms/frameworks for implementation of knowledge driven higher level coordination and communication strategies based on the MAS research results.
- ***Usage of holarchy principle:*** The holarchy principle, which allows the creation of a holon as an integrated set of lower level holons, is used in HMS. This is not considered in the MAS field where autonomy and functional differences of individual agents are preferred. However, agents very often group themselves into hierarchically organised teams.
- ***Human interface:*** Each holon is usually equipped with a human interface. Human interfaces in MAS are very often implemented as separate agents providing services to the community as a whole.

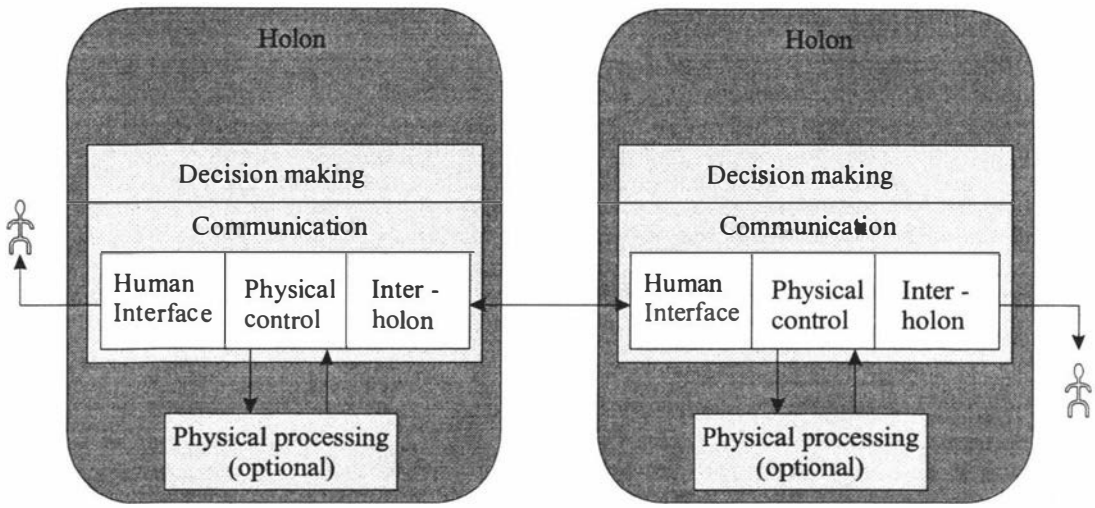
5.7.3 Holonic Manufacturing Systems and Multi-Agent Systems integration

There are no conventional methods for realising the holonic characteristics. To realise a holonic manufacturing system, comprising numerous interacting holons, special techniques for task executing and problem solving have to be applied. From a software engineering perspective a holon, as a unit of composition retaining the characteristic attributes of the whole system (holarchy), can be viewed as a class. Thus the object-oriented paradigm seems suitable for modelling holarchies as software systems. However, because of its characteristics (of autonomy, cooperation, and pro-activeness at the first place), an agent is nearly predestined for implementing holons. Thus, a Multi-Agent System appears an even more suitable tool for emulating holarchies as software systems than an object-oriented model. Through the concept of 'partial cloning' characteristics from a real physical entity (a holon) which are needed for pursuing collaborative actions in holarchy are abstracted and encapsulated in a software entity (an agent). Thus these software entities, which emulate the physical entities, enable the coordination of production through intelligent control procedures [Brennan, 2000]. A MAS which emulates a holonic system will consist of specialised autonomous agents (which have a particular structure and holonic properties), driven by a coordination mechanism designed according to the rules for cooperation of the respective holarchy. With this in mind it is easy to point out that software holarchies are specialised MAS's that define the interaction between their agents based on the underlying cooperative holonic units [Ulieru, 2002].

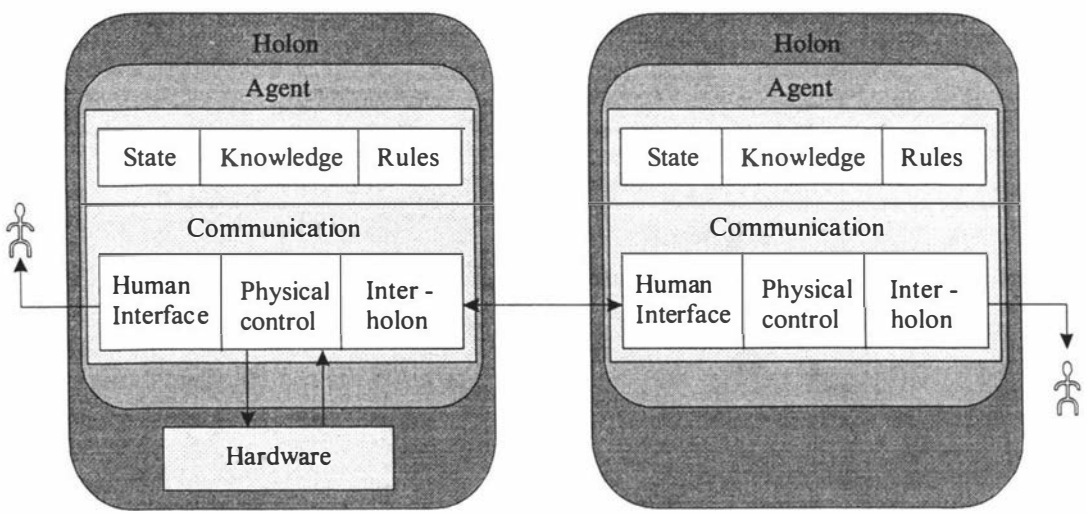
The issue is not "convergence" of the two paradigms but rather the implementation of an organisational holarchy (real life system) into software using the MAS paradigm. It should be noted that holonic manufacturing is not an *alternative* nor an *identical* approach to multi-agent control but rather it is *complementary* in that it represents a systems engineering approach to the development of manufacturing control systems infrastructure, rather than a solution mechanism for solving individual manufacturing control problems [Ulieru, 2002].

Figure 5-5 describes a potential pattern of two holons, where the "intelligent" control module is implemented by agents [HMS, web]. The decision-making unit of the holon and the communication with other holons and humans, and optionally the physical entities are all adopted by agents. To take decisions an agent needs some information such as knowledge about

the outer world or other agents. The *state* makes the agent react in a different way to similar inputs. It can be changed corresponding to some *rules*, which are based on *knowledge*. Obviously the MAS techniques are convenient for realising holonic systems. In fact, most developments of holonic systems to date have deployed agent-like solvers as a means of resolving planning, scheduling and shop floor control issues [Ulieru, 2002].



a) General architecture of a holon



b) Agent-oriented architecture for a holon

Figure 5-5. A holon architecture

5.8 Design principles for highly distributed heterarchical Shop Floor Control systems

For designing highly distributed heterarchical systems Prabhu reported the following design principles: [Prabhu, 2000].

- ***There should be no master scheduler.*** Complexity due to inhomogeneity of levels[1] would be reduced to give much simpler and more manageable problems. The benefits of

[1] The control equipment that needs to be interfaced usually comes from many diverse sources - different brands.

reduced complexity would include reduction in development cost, and improvement in the chance of success.

- ***Scheduling and control must be contained within the same logical entity.*** Such an approach would ensure the highest level of local autonomy. The most important information required for scheduling would be available within the entity, avoiding the need for using time critical messages. Also modularity of the system would ensure that the system is easy to modify and integrate.
- ***Entities must follow the principle of least commitment.*** The establishment of relationships must be delayed as long as possible and terminated as soon as possible. Uncertainty problems caused, for example, by changes during production would be, therefore, be partly solved.
- ***Entities must co-operate with other entities whenever possible.*** This is a rational requirement of a heterarchical system. Such an interaction is necessary to solve resource sharing conflicts in the best possible way.
- ***Entities should abide by the schedule whenever possible.*** In the system as a whole, unnecessary surprises, which might occur if any entity pursued its own agenda, would be eliminated. This is in accordance with the spirit of co-operation in the system.
- ***Entities should not assume that other entities will abide by their schedules.*** Loose coupling between entities should be ensured. In return, this would increase implicit fault-tolerance.
- ***Entities should not need to know the schedules of other entities.*** Global data should be avoided. Dependency on other entities would be minimised and implicit fault-tolerance would increase.
- ***Entities must not make any assumptions about the system characteristics.*** (For example, infinite buffers, machine failure characteristics, processing times, etc.) Global information would be minimised and implicit fault-tolerance would be further increased. Also, this would increase the system flexibility.
- ***Entities must be under minimum design constraints.*** There should be no predetermined routes or sequences between entities built into the entities themselves. Therefore, system flexibility could be used to cope with shop floor uncertainties.
- ***Entities must autonomously decide trade-offs between local and global performance.*** In this case the highest level of autonomy would be ensured and communication requirements would be minimised. For this process to occur, global information is required as discussed below.
- ***Entities must form the local schedules with a global perception.*** To ensure system survival, instead of local, entity goals, local schedules must pursue overall system goals. Global perception of the system is the essence of co-operation in heterarchically controlled systems.

These principles are important considerations in the design of the system presented in this thesis (see Chapter 10). The next chapter, therefore, is dedicated to a brief overview of Multi-Agent Systems which have been seen as a promising approach for developing distributed manufacturing control system.

Chapter 6. A review of research on Multi-Agent Systems

Agent systems represent a relatively a new research area that has been developed under the broad scope of Artificial Intelligence (AI). As this is a *highly heterogeneous field*, providing only a very brief and modest review of Multi-Agent Systems is possible in this thesis and this is the primary goal of this chapter. (Multi-Agent Systems are applied in different domains of human activates, hence each multi-agent implementation requires the application of different specific tools, methodologies, and techniques that are appropriate for these domains.) In addition to providing basic background information on Multi-Agent Systems, this chapter also describes in more detail some of the key "features" that influenced the design of the distributed heterarchical shop floor control system advocated in this research project (described in Chapter 10). The negotiation mechanism and the Contract-Net Protocol (CNP) are examples of such topics.

The chapter is organised around the following main sections:

Section 6.1 gives a few agent definitions, summarises common agent properties, and provides some agent classification schemas.

In Section 6.2 different kinds of coordination mechanisms and protocols that are commonly used among autonomous agents in Multi-Agent Systems are reviewed. The Contract-Net Protocol which is the most commonly used agent protocol for resolving task coordination and resource allocation problems among agents in manufacturing environment, is also covered in more detail in this section.

Section 6.3 is dedicated to the agent systems applied in manufacturing systems. Topics such as the reasons for using agents in manufacturing systems, what entities in manufacturing systems can be mapped as agents, inter-agent communication, multi-agent architectures in manufacturing control systems, scheduling in Multi-Agent Systems, and advantages and disadvantages of applying Multi-Agent Systems, are covered in this section.

Section 6.4 briefly reviews some tools and standards used for developing multi agent applications, and finally, Section 6.5 provides some concluding comments on Multi-Agent Systems.

Contemporary agent-based technologies can be classified as being either single-agent or Multi-Agent Systems. In ***single-agent systems***, an agent performs a task on behalf of the user or some process. While performing its task, the agent may communicate with the user as well as with local or remote system resources, but it will never communicate with other agents. In contrast, the agents in a ***Multi-Agent System*** (MAS) not only communicate with the user and system resources, but they also extensively communicate and work with each other, solving problems that are beyond their individual capabilities. This review is about Multi-Agent Systems and all the material is related to them, except where otherwise stated. Before discussing agent-based systems, the concept of an "agent" will be clarified first.

6.1 What is an agent?

First and foremost there is no straightforward answer to the question of what is an agent. There are two reasons for that. Firstly, the same concept of building agents is used by a vast number of researchers in various research domains to create systems with different capabilities. As a result, each definition of the term is based on the inherent capabilities of the agent. Secondly, agent-based systems are a relatively new research area. In fact, the literature is flooded with a variety of definitions, but these definitions just confirm that there is no general agreement as to what constitutes an agent. This point is made well by Krogh [Krogh, 1996]:

"... What's an agent anyway? Agents may be many things. Attempts to find one central common denominator of operative or theoretical conceptions of agents in recent publications on the topic ... will probably fail".

Therefore, the few definitions that follow do not represent a research consensus, and it should be no surprise even if they are in direct conflict with each other. However, they are useful in communicating some of the main features of agent research.

6.1.1 Agent definitions

As indicated above, many definitions remain specific to a given set of examples of agents that the definers had in mind, for example, *"Agents are software entities that have enough autonomy and "intelligence" to carry out various tasks with little or no human intervention. They are software delegates of individuals and organisations, and can act on behalf of their delegators"* [Wong & Sycara, 1999]. Others are too broad in their meaning: *"An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors"* [Russell & Norvig, 1995, p33] or, for example, the other definition commonly found in the literature *"An agent is an entity which acts on behalf of its user"*. A brief review of a few agent definitions is given in [Franklin & Graesser, 1997]. Investigating them, the authors extracted some common attributes and gave their own definition of an agent: *"An autonomous agent is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future".*

In the software engineering domain, the above definition allows us to draw a distinct line between ordinary programs and agents. A program that senses the world via its input and acts on it via its outputs is not considered an agent unless its output affects what is sensed later. Usually a program also fails the "over time" test of temporal continuity. A program runs once and then waits to be called again. For instance, a spell checker adjunct to a word processor is typically not an agent, but a spell checker that watches as one typing and corrects "on the fly" might well be an agent. But according the above definition, an agent need not to be a program at all; it may be a machine, a robot, or an operator.

In defining software agents Parunak sees agents as the next natural incremental step in software evolution [Parunak, 2000a]. Namely, a software agent can be thought of as a natural extension to object-oriented programming. This is an interesting point since agent-based solutions are more likely to be considered by business and industrial people (often with conservative attitude) if they can be seen as an incremental step that builds on proven earlier technology. As Parunak stated: *"Once agents have been seen as that, it is a relatively simple matter to explain to users why one or another accessory is needed for the agents that will address their particular application".*

Originally, the basic unit of software was the complete program. Arbitrary jumps of control made it difficult to manipulate any unit of code and the entire program. In ***monolithic programs***

data often intermingled with program code and a programmer was responsible for determining the behaviour of the complete program before it begins execution.

The ***structured programming*** approach designed programs from smaller packages of code, such as structured loops and subroutines, with a high degree of local integrity. Though a subroutine's code was encapsulated, its state had to be supplied externally through arguments. The code of a subroutine gains control only when the subroutine is externally invoked by a call.

The next generation in programming evaluation was ***object-oriented programming***. Objects not only provide data and code encapsulation, but they also provide a modular way of modelling a system in terms of its entities and the interactions among those entities. However, objects are passive entities, which gain control and change state only in response to method invocations[1] from external entities so that they have no control over when those invocations will occur.

Software agents take the next logical step of giving each object its own thread of control and its own internal goals, thus localising not only code and data, but also invocation. From this perspective, an agent is *"an active object with initiative"*, or more clearly, *"a pro-active object."* Unlike other objects, it does not need to be invoked, but constantly monitors its local environment and acts autonomously based on its individual programming [Parunak, 2000c]. Though each agent runs independently of the others, agents do interact among themselves. Interaction is performed by sending and receiving messages, but not through the form of function calls. It is possible to broadcast a message to a group of agents, and each recipient agent is free to ignore a message or deal with it at a later time.

Table 6-1 depicts the relation of agents to other common software technologies. Values in the table fields show places where the key questions (column header) are answered. Agents are portrayed as the natural next step in a software development of increasing the software localisation and encapsulation.

Table 6-1. Agents from perspective of software evolution [Parunak, 2000a].

	Monolithic Program	***Structured Programming***	***Object-Oriented Programming***	***Agent-Oriented Programming***
How does a unit behave? (Code)	External	Local	Local	Local
What does a unit do when it runs? (State)	External	External	Local	Local
When does a unit run?	External	External (called)	External (message)	Local (rules; goals)

Yet another definition of an agent that is often quoted in literature related to manufacturing systems is: *"An agent is a computer system situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives"* [Jennings & Wooldridge, 1998]. An autonomous agent should be able to act without the direct intervention of humans or other agents, and should have control over its own actions and internal state and an "agent based system" means one in which the key abstraction used is that of an agent.

---

[1] Each ***software object*** (a code-based abstraction of real-word item or relationship) has its own ***properties*** (a set of data that describe object's state), ***methods*** (procedures or functions that manipulate object's data) and ***events*** (predefined responses (event procedures) to certain external events (such as a mouse click for example)). The only way to change the object's state (data) is through the invocation (calling) of the object's methods.

6.1.2 Common agent properties

Focusing even on software agents only, it is likely that the above definitions omit many characteristics that some researchers insist are essential for an agent (for example, the agent's ability to move over a network, or carry out the function of representing a human). To overcome the problem of a universal agent definition usable for all kinds of problems, a list of common agent properties might be helpful to further classify agents. From the discipline of agent-based software engineering the following properties of agents can be distinguished [Luck & d'Inverno, 2001]:

- ***Autonomous.*** An agent operates without the direct intervention of other systems or humans (acts without being invoked), and is able to take an initiative and exercise a non-trivial degree of control over its own actions.
- ***Social ability.*** An agent has a capability to interact with other agents and possibly humans (by mutual message-based communication).
- ***Reactive.*** This is a characteristic of an agent that perceives and responds to changes that occur in the environment in a timely fashion.
- ***Pro-active (self-starting).*** Unlike standard programs that are directly invoked by the user, an agent can take the initiative and decide when to act. Sensing changes to its environment, agents act by exhibiting goal-directed behaviour.
- ***Goal-Oriented.*** An agent accepts high-level requests indicating what a user of an agent wants, and the agent is responsible for deciding how and where to satisfy that request.
- ***Collaborative.*** An agent does not blindly obey commands, but has the ability to modify requests, ask clarification questions, or even refuse to satisfy certain requests.
- ***Flexible.*** The agent's actions are not scripted. An agent is able to dynamically choose which actions to invoke, and in what sequence, in response to the state of its external environment.
- ***Temporal continuity.*** Agents are continuously running processes, not "one-shot" computations that map a single input to a single output and then terminate.
- ***Character.*** An agent might have a well defined believable "personality and emotional state".
- ***Communicative.*** An agent is able to engage in complex communication with other agents, including people, to obtain information or enlist their help in accomplishing its goals.
- ***Adaptive (learning).*** An agent automatically customises itself to the preferences of its user, based on previous experience. The agent also automatically adapts to changes in its environment.
- ***Mobile.*** An agent is able to transport itself from one computer to another and across different system architectures and platforms.

Most agents possess some of these properties but not all as will be indicated in the following.

6.1.3 A few agent classification schemes

To help with understanding what else an agent can be, a few classification schemes are given below. Accordingly to Keilmann [Keilmann, 1995] the following agents can be distinguished:

- ***Primitive agent.*** A sensor-actor system that reacts on information received by its sensors in a fixed manner.
- ***Technical agents.*** Robots and flexible transport-systems, such as those that are available currently. They often include some type of programmable control.
- ***Technical-intelligent agents.*** Autonomous systems that are capable of solving a class of defined problems on their own, under circumstances that do not have to be known in advance.
- ***Cognitive agents.*** Agents those are able to learn.
- ***Social agents.*** Agents that have social behaviour from an observer point of view. For example, they are able to recognise and identify other agents (artificial and humans) and to establish and maintain inter-species relationships with them. They might have characteristics such as making commitments to one another, planning tasks, and learning from experience. More sophisticated agents may have recognisable personality, emotional states, and some form of embodiment (like cartoon creatures, or virtual pets - robots). Some of them even might have an ability to learn from experience, and to be - capable of telling a story (dynamically reconstructing its individual "history" during its life time). An interesting study on social agents that includes many references from this field is given in [Dautenhahn, 1998].

An agent's sophistication can theoretically range from ***simple sensing*** agents (with no memory and no model of other agents) that react to their environment, all the way up to agents with ***full human capabilities***. However, in practice, actual implementations add at least *memory* to ***sensing agents***, so that the agents maintain information on local states. The next level of sophistication is ***self-consciousness***, in which each agent knows of the existence of other agents as distinct from itself, and thus can carry on rudimentary communication. A ***social agent*** goes a step further and has the ability to model other agents' states, goals, and plans [Shoham, 1993]. Even higher agent capabilities include such functions as those presented in the classifications above.

Another classification is possible accordingly to the nature of interacting mechanisms among agents. Namely, in Multi-Agent Systems, agents either try to compete against each other or try to collaborate with each other. This gives rise to two different types of agents participating in a Multi-Agent System:

- ***Competitive Agents.*** Each agent's goal is to maximise its own interests, while attempting to reach agreement with other agents. Examples are buying/selling agents where negotiation to resolve conflicts is at a competitive level, namely each is trying to obtain the lowest/highest price possible for its own good and not for the good of the market community as a whole. Hence these agents are working on behalf of an individual user and not as part of a unified community.
- ***Collaborative Agents.*** In contrast to the above, collaborative agents share their knowledge and beliefs to try to maximise the benefit of the community as a whole. For example, in an air traffic control scenario, the agents (representing the pilots and computerised air traffic control systems) engage in a collaborative planning process to construct the best plan for re-routing flights in and out of an airport. Here negotiation is at a collaborative level, where agents try to resolve conflicts via collaborative discourses, instead of competitive bidding. Indeed, some researchers characterise collaborative agents as being rational and attribute them with mentalistic notions, such as beliefs, desires, and intentions.

Many other classifications are possible. For example, by control architecture, by the range and sensitivity of agent's senses, according the tasks they perform, (regulation, planning, and adaptive agents), according the environment in which the agent find itself, (human agents, software agents, and artificial – hardware – agents), and so forth.

6.1.4 Individual agent architecture

Agents usually perform diverse functions as agents and they might differ from one another in their internal structure. Their architecture can be:

- ***Dissimilar.*** This architecture is specific for agents that do not communicate directly with one another, but simply interact through their effects on the world.
- ***Body-head.*** This architecture is characteristic for agents that communicate with one another. They must speak the same language [Jennings et al., 1992]. Agents share identical modular "heads" (including at least communication interfaces to permit communication among agents), but the code in their "bodies" is architecturally distinct from the head and may differ radically from one agent to the next. The head contains the knowledge of the individual agent (specific to each class of agents) and the knowledge of the rest of the community (common for all agents)
- ***Almost identical.*** All agents have the same architecture so that agents all run the same code and differ only in state parameters [Morley & Schelberg, 1993].

6.2 Coordination among agents

Achieving coordination among autonomous agents in Multi-Agent Systems (MASs) is a major problem. To interact among themselves, agents must be able to share some commonalties. Most often agents use a ***common artificial language,*** but in the absence of a common language, agents can interact by changing a ***shared physical environment*** and sensing those changes, see the text below [Parunak, 1998a].

Coordination between agents may take a form of:

- ***Inter-agent communication***, which is based on sending digital messages over a communication network using appropriate protocols. The content of the messages is determined by the kind of information to be communicated while protocols determine the *meta-level* rules (structured templates) that govern the communication between agents. More specifically, to be able to communicate with each other, an agent language, a transport protocol, and an interaction protocol have to be defined for every agent. ***The agent communication language (ACL)*** defines the set of communication primitives (both syntax and semantics) that the agents can either generate or receive during communication, and specifies a declarative representation of the content of the message. To understand the content of a message agents have to share common ontology[1] [Gruber, 1993]. A vocabulary and semantics of the messages must be determined for the agent communication language. The ***transport protocol*** is the actual transport mechanism used for the communication. The best known is TCP/IP (**T**ransmission **C**ontrol **P**rotocol/**I**nternet **P**rotocol) suite of communications protocols used to connect hosts on the Internet. The ***interaction (coordination) protocol*** is a description of standard patterns

[1] *Ontology* defines a set of definitions (a set of agreed-upon symbols - words) of content-specific knowledge. An ontology is a separate, publicly available information model constructed for the given problem domain and serves as a common dictionary for the agents.

of interaction between two or more agents. It constrains the possible sequences of messages that can be sent amongst a set of agents to form a conversation of a particular type. An agent initiating a conversation with others can indicate the interaction protocol it wishes to follow, and the recipient (if it knows the protocol) then knows how the conversation is expected to progress. The most common in manufacturing environment is the Contract-Net Protocol (see Section 6.2.3.1 "The Contract-Net Protocol"). Communication may exist at a number of levels of sophistication. Interaction between agents in agent-based systems is predominantly done through the process of ***negotiation*** (see Section 6.2.2 "Different kinds of negotiation").

- ***Solely coordination***, which is based on the process of making changes in a shared environment. An agent sense changes in environment and accordingly to these changes an agent can change the environment and/or to alter its behaviour in response to those changes. In that case, a head of a "body-head architecture" consists of common sensor-effector modalities. An intuitive approach and the theory of games are two characteristic approaches for this type of coordination between agents [Keilmann, 1995].

There is no single best coordination mechanism. Instead, different mechanisms should be applied in different classes of problems, as they are most appropriate.

6.2.1 Communication strategies

To provide inter-agent communications there are several different communication strategies depending on whether a message is addressed to specific agent or broadcast to the entire population, and whether messages persist after being sent. Table 6-2 reviews a few of these.

Table 6-2. Communication strategies

Are messages addressed to specific agents?	***Do messages persist after being sent?***	***Communication Strategies***
Yes	Yes	Interactive Transition Net [Lee et al., 1993]
Yes	No	Directed
No	Yes	Blackboards [Nii, 1986]
No	No	Broadcast

In designing an inter-agent system the choice of communication strategy is an important issue. The adoption of the "right strategy" depends on the system application domain (the nature of a problem which needs to be resolved).

6.2.2 Different kinds of negotiation

A common, very flexible way for designing the interaction among agents is to enable them to carry out negotiations, like humans usually do when interacting between themselves. Negotiation can be thought of as a discussion process in which interested parties (a group of agents) exchange information to reach a mutually accepted agreement on some matter. During negotiation, agents do not need to exchange information about their current state, or how they solve a problem or carry out a computation task but they need to transmit some kind of meta-level information.

With regard to how the goals of negotiation are achieved, negotiation mechanisms can be either ***cooperative*** or ***competitive*** [Parunak, 1998a]. An example of cooperative negotiations is the

negotiations about product design that are often structured to maximise the amount of common knowledge about the corporate goal being pursued. To the contrary, in competitive negotiations opponents often seek to conceal their individual objectives from one another.

Negotiation mechanisms can also differ according to the various classes of threats against which they offer robustness. Successful negotiation mechanisms should be able to cope with the issues such as: communication failures, misunderstanding, insincerity, ambiguous information, and dynamic change of context [Parunak, 1998a]. ***Communication failures*** during the course of a negotiation have to be detected and recovered from. Nowadays, this is not perceived as an issue since simple check-sum and acknowledgement mechanisms, which are embedded in the underlying communication system (TCP/IP suite), are quite sufficient for that purpose. In many cases these mechanisms operate invisibly to the application implementer. Sometimes it can be difficult to detect ***misunderstanding***, when agents successfully exchange messages but honestly interpret them in different ways. Hence agents need to use the same semantics for messages. However, some agents do use the same semantics, but occasionally (social agents) deliberately misrepresent the world. In some cases the negotiation mechanism has to provide the means to cope with the ***problem of insincerity*** (misrepresentation of commitment). Sandholm suggests a commitment protocol that uses penalties for defaulting [Sandholm, 1996]. Other mechanisms have been devised to eliminate misrepresentation of facts caused by ***the ambiguity of information*** [Rosenschein & Zlotkin, 1994]. Also, if the context of a negotiation can ***dynamically change*** over time then renegotiation or a meta-level negotiation mechanism is required.

6.2.3 Coordination mechanisms

In the field of distributed artificial intelligence, for achieving coordination among distributed entities different techniques have been proposed. Examples include the Contract-Net Protocol for task allocation [Smith, 1980], and the Multistage Negotiation Protocol for resource allocation under global constraints [Conry et al., 1991]. These protocols have been further customised to suit various application domains and have a lot of minor variations (a trend for customised protocols will increase as the software agents proliferate).

A communications mechanism determines how conversations among agents are structured. We could distinguish the following protocols:

- ***Reactive (react principle).*** Agents react without question or discussion to messages or commands from one another.
- ***Command or Directive (central moderator principle).*** Agents either give orders to one another (as in classical hierarchical architectures) or they express their opinions to a central moderator, who counts their opinions or gives their opinions weights in order to select a course of action, [Fordyce et al., 1992].
- ***Voting (constraint propagation principle).*** Agents express themselves by a scalar quantity, and their behaviour is determined by some measure of aggregation (such as a sum, product, or average) over the scalars of different agents. Decisions are made on the basis of multidirectional discussions among agents, without a rigid structure implicit in the negotiation.
- ***Fixed protocol (negotiation principle).*** As in voting, negotiating agents exchange a series of messages before a decision is made on a course of action. The canonical example is the Contract-Net Protocol. Negotiation protocols determine a fixed (static) set of options that a conversation can follow, and this structure is fixed when the agents are implemented. Voting and negotiation differ in three ways. First, while both a bid and

a vote are intended to influence the behaviour of the recipient, the bid has the additional purpose of acquiring further assignments for the sender. Second, the exchanges in voting tend to be shorter than those in negotiation, since the protocol must include additional steps for awarding the contract and managing the execution of the task. Third, while the simplest bids (like votes) are just scalars, bids can become much more complicated, including extensive symbolic information representing the timing and specifications of the task to be performed.

- ***Conversation (Speech act principle).*** More recently, speech act theory has been used to build general grammars out of which arbitrary protocols can be constructed, and agents that understand these grammars can evolve new protocols for their conversations as they operate. Agents are able to reason about messages they have received, and decide dynamically what sorts of utterances (for example, assertions, directives, declarations, or commitments) are appropriate. Communication is of a much higher degree of complexity than in Contract-Net Protocol or constraint propagation. Section 6.4.3 "Agent languages" briefly overviews the most popular languages used in inter-agent communication.

Table 6-3 gives examples of several classes of coordination protocols arranged from less complex to more complex.

Table 6-3. Coordination protocols

Class of Protocol	***Description***	***Minimal Language***
Reactive	Sense, then act	Environment
Command	Master agent sends unilateral instructions to servants	Symbolic
Voting	One-shot quantitative statement of interest	Currency
Fixed Protocol	Back-and forth; symbolic and quantitative	Messages (CNP)
Conversation	Arbitrary interchange	Speech Acts

Many coordination protocols employed in Multi-Agent System (MAS) applications are based on Smith and Davis' Contract-Net Protocol [Smith & Davis, 1981].

6.2.3.1 The Contract-Net Protocol

Direct interaction of agents is based on their ability to communicate and exchange messages. The Contract-Net Protocol (including its modifications) is the most commonly used method in the manufacturing environment. There are a number of research studies in which the Contract-Net Protocol has been used and some examples include the following: [Baker, 1991, Dewan & Joshi, 2000, 2001, Kanchanasevee et al. 1997, Lin & Solberg, 1992, Ottaway & Burns, 2000, Ouelhadj et al., 1999, Parunak, 1987b, Saad et al.; 1997, Shaw, 1988, Wang & Usher, 2002].

The fundamental idea of contracting that is implemented in the process of negotiation in distributed multi-agent controlled manufacturing systems is not new and dates back almost 30 years. In the 1970s a rudimentary bidding scheme was used for resource allocation in distributed computing systems [Farber et al., 1973]. Later, the Contract-Net Protocol principles were applied in ***distributed problem solving*** in the domain of Distributed Artificial Intelligence (DAI).

The Contract-Net Protocol is a bidding based, high level, negotiation protocol, which originally was designed by Reid Smith and Randall Davis [Smith, 1980, Smith & Davis, 1981] to facilitate

problem solving in distributed environments. The protocol assumes an architecture of independent nodes that are interconnected in a computer network. The physical architecture is irrelevant. Each node owns individual resources and can communicate with every other node by low-level communication protocols that support reliable and efficient communication of bit streams between nodes. Protocols underneath the Contract-Net Protocol enable reliable, error free transmission of data between nodes, but do not consider the semantics of the information being passed. Contrary to this, the Contract-Net Protocol specifies communication and control in a distributed problem solver at the highest level. The Contract-Net Protocol assigns interpretations to the data. It offers a structure that assists the system designer in deciding *what the nodes should say to each other*, rather than *how* to say it. In the text that follows basic notions regarding distributed problem solving and the Contract-Net Protocol will be reviewed.

6.2.3.1.1 Distributed problem solving - basic definitions

Distributed problem solving offers great potential for the solution of single problems that appear complex and difficult because of their size. These "large" problems might be able to be broken into modular sub-problems (if that is possible) and then a collection of processors (multiple problem solvers), each of which handles some fraction of the total problem (sub-problem), can be applied together to get a solution of the single "large" problems. Such problem solvers offer the promise of speed, reliability, extensibility, and the potential for increased tolerance to uncertain data and knowledge, as well as the ability to handle applications with a natural spatial distribution.

The concept of distributed problem solving is described as "*the **cooperative solution** of problems by a **decentralised** and **loosely coupled** collection of **knowledge sources (KSs)** located in a number of distinct processor nodes.*" [Smith, 1980].

The key issue to be resolved in distributed problem solving is how tasks are to be distributed among the processor nodes. The dynamic decomposition and distribution of sub-problems - subtasks - is called ***task sharing***. The key issue to be resolved in task sharing is how tasks are to be *distributed* among the processor nodes. Nodes (*managers*) with tasks to be executed need to find the most appropriate idle nodes (*contractors*) to execute those tasks. This is called the ***connection problem.*** Connection problem can also be viewed from the perspective of an idle node. An idle node must find another node with an appropriate task that is available for execution.

Solving the connection problem is crucial to high performance in a distributed problem solver. It has two aspects: 1) resource allocation and 2) focus. Effective ***resource allocation*** is achieved by balancing the computational load among the nodes. ***Focus*** is achieved by the effective selection of tasks for allocation to nodes and by effective selection of KSs for the execution of tasks. Note that the most appropriate KS cannot be identified a priori. The combination of many tasks and many applicable KSs can lead to a combinatorial explosion. Therefore, focus needs to be maintained to achieve high performance in practical applications.

A connection problem can be solved through the process of ***negotiation***. This process represents a fundamental mechanism of interaction between nodes. Negotiation is characterised by the four important points:

- It is a process that does not involve centralised control;
- There is a two-way exchange of information;
- Each party to the negotiation evaluates the information from its own perspective; and
- Final agreement is achieved by mutual selection.

6.2.3.1.2 The Contract-Net Protocol – details

In the Contract-Net Protocol the collection of nodes is referred to as a ***net*** and the execution of a task is dealt with as a ***contract*** between two nodes. Each node in the net takes on one of two roles related to the execution of the responsibilities of an individual task: manager or contractor. A ***manager*** is responsible for monitoring the execution of a task and processing the results of its execution. A ***contractor*** is responsible for the actual execution of the task. Any node can take on either role dynamically during the course of problem solving. Typically, a node (agent) will take on both roles, often simultaneously for different contracts. This way of task execution by contracts between nodes in a system is defined as a ***contract net***.

In the Contract-Net Protocol a process of negotiation is based on the principles of auctioning. An ***auction*** can be defined as a set of negotiation activities accomplished among distributed entities (nodes). Each entity applies certain criteria to select another entity with the minimum or maximum value of a given measure from the set of available entities. An auction can also be considered as a cycle in the negotiation process that consists of certain types of messages interchanged between workstation agents.

The main mechanism on which the Contract-Net Protocol is based is described as follows. When a node (called a manager) within a contract net needs to find another node to deal with a problem that the former cannot solve, it ***broadcasts a task announcement*** (contract) for that problem. Other nodes in community (called contractors) evaluate the task announcement (together with any other task announcements that can be made by several managers) and ***submit their "bids"*** for solving the problem back to the manager node. The nodes submit bids on those tasks for which they are suited. ***Bids are evaluated*** and then the ***contracts are awarded*** to the nodes determined to be the most appropriate. The negotiation process may then reoccur. A contractor may further partition a task and award contracts (or sub-contracts) to other nodes. It is then a manager for those contracts.

The Contract-Net Protocol allows distributed problem solvers (managers) to contract other problem solvers (contractors) for the solution of sub problems via a fixed interchange of messages. When a manager needs a contractor to execute a task, it initiates an exchange of messages to find the most suitable contractor (see Figure 6-1). Each message in the Contract-Net Protocol has a set of slots for the task specific information in the message. The information that fills the slots is encoded in a simple language common to all nodes (refer to Section 6.2.3.1.3 "The Common Internode Language").

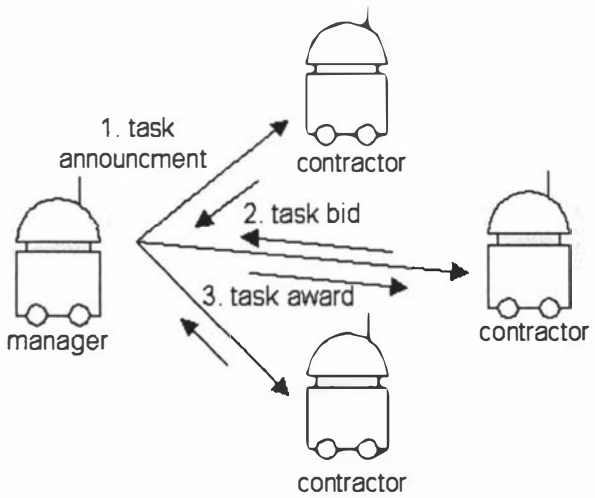


Figure 6-1. Messages exchanged by the Contract Net Protocol

A core part of the protocol is made up of task announcements (requests), bids, and awards messages. Note that in addition to these core messages, the original specification of the Contract-Net Protocol contains the following set of messages: acknowledgment, report, termination, node-available-message, request-message, and information-message.

The structure of the core messages is as follows:

a) Task announcement: A node that generates a task normally initiates contract negotiation by advertising the existence of that task to the other nodes with a task announcement message. It then acts as the manager of the task. The manager broadcasts or sends (if the contractor or contractors are identified) a message demanding a service.

A task announcement can be addressed to: 1) all nodes in the net (general broadcast), 2) a subset of nodes (limited broadcast), or to 3) a single node (point-to-point). The latter two modes of addressing, reduce message processing overhead by allowing non-addressed nodes to ignore task announcements after examining only the **addressee** slot. The saving is small, but is useful because it allows a node's communication processor alone to decide whether the rest of the message should be examined and further processed.

The message includes the following parts:

- ***Destination address:*** A broadcast or the contractor's address.
- ***Origin address:*** The manager's address.
- ***Type:*** TASK ANNOUNCEMENT.
- ***Contract number:*** A number to identify the demand (simultaneous demands can share the environment).

The set of information slots (parts) specific to the task announcement message are:

- ***Task abstraction:*** The task description. Provides information about a task type and geographic position of the manager.
- ***Eligibility specification:*** The contractor requirements. This is a list of criteria that a node must meet to be eligible to submit a bid. For example, it indicates that the only nodes that should bid on this task are those that are located in the same area as the manager that announced the task.
- ***Bid specification:*** The characteristics of the service demanded. It indicates the information that a manager needs to be able to select a suitable node.
- ***Expiration time:*** The time limit to answer the demand. This indicates a deadline for receiving bids.

b) Task bid: Those contractors that can accomplish the requirements of the task announcement answer the manager and submit a bid that details their offer. The message includes the following parts:

- ***Destination address:*** The manager's address.
- ***Origin address:*** The contractor's address.
- ***Type:*** BID.
- ***Contract number:*** The number to identifying the demand

The set of slots specific to the bid message are:

- ***Node Abstraction:*** The contractor's offer. This slot is filled with a brief specification of the capabilities of the bidding node that are relevant to the announced task.

c) Task award: The manager analyses the submitted bid. If any of the bids accomplish its expectations, the manager awards the task to the contractor and establishes a contract. If several contractors can accomplish the task a priority selection process such as the shortest time to complete or the lowest cost will decide which contract is accepted. If there is no suitable bid, the manager can start another demand. The award message advises the successful contractor that they have been awarded the task and it includes the following parts:

- ***Destination address:*** The contractor's address.
- ***Origin address:*** The manager's address.
- ***Type:*** AWARD.
- ***Contract number:*** A number to identify the demand

The set of slots specific to the award message are:

- ***Task specification:*** Further data or specifications required to execute the task.

6.2.3.1.3 The Common Inter-node Language

The Contract-Net Protocol provides a framework for problem solving. The protocol specifies the *type* of information that is to fill a message slot but still it leaves to the user the difficult task of specifying the actual *content* of that framework (slots) for any particular problem domain. However, the Contract-Net Protocol offers the users some additional assistance. It provides a very simple high-level language based on an *object, attribute, and value* representation. This language is called a *common internode language* and it is used for encoding slot information in such a way that is understandable to all nodes. The language includes a simple grammar, predefined for each slot, and a number of predefined domain-independent terms (e.g., "task", "type", "procedure", and "name"). They are offered to the users to help them to organise and specify the slot information but the users must augment the language with domain-specific terms (e.g. PartID, ToolID, etc.) as needed for the application at hand. Such a language, along with a high-level programming language (for transfer of procedures between nodes), forms a common basis for communicating slot information among the nodes.

A message that does not have to be understood by many nodes (e.g., messages exchanged by a manager and contractor during the execution of a contract) can be usefully encoded in a private language. This can reduce both the lengths of the messages and the overhead required for their processing. In the Contract-Net Protocol an "escape" character precedes such "private" information. This allows private information to be inserted in any message, even one that includes some public information encoded in the normal manner.

6.3 Agents in manufacturing systems

Interest in distributed control of manufacturing systems originated in the late of 1970s with the pioneering work of Lewis [Lewis, 1981] in negotiation-based shop floor control systems. Nowadays, in manufacturing, most Multi-Agent Systems (MASs) are applied to two functions:

- ***Production.*** The most common application of MASs to manufacturing is in production, mostly in scheduling and control, and to a lesser extent in monitoring and diagnosis.
- ***Design.*** Here, multiple agents help an interaction between design and production engineers by taking care that design conflicts are avoided and manufacturability is ensured.

A few systems deal with other manufacturing functions such as power distribution, information integration, and logistics.

In recent years, because of its commercial potential, researchers in agent-based systems find *"shop floor scheduling and control increasingly attractive as an application problem"* [Baker, 1998]. Multi-Agent Systems (MASs) offer production control systems that are decentralised, evolving, and concurrent in contrast to conventional systems that are characterised as centralised, planned and sequential. The autonomous agent approach replaces a centralised database and control with a network of agents. Each agent has a local view of its environment and the ability and authority to respond locally to that environment. The overall performances are not globally planned, but emerge through the dynamic interaction of the agents in real-time. A schedule emerges from the concurrent independent decisions of the local agents.

6.3.1 Why use agents in manufacturing systems?

There are a number of compelling reasons to use agents in manufacturing systems. A brief list of a few of them is given below, while more details are provided later in Section 6.3.6.1 "Benefits of distributed, multi-agent, heterarchical architectures".

Why use agents in a factory?

- Distributed entities on the shop floor can be mapped with ease and in natural way by agents.
- Both people and machines can be treated as agents. Thus, they can work together as colleagues instead of being opposed to each other, (humans operate as peer elements in the system rather than being run by the system) [Hatvany, 1985].
- With the multi-agent paradigm, entities on the shop floor (machines, tools, parts, etc.,) are endowed with some level of local intelligence and start to be "partners" to humans. People are still in overall charge of processes (through evolving control) but equipment is more sophisticated and assumes much more responsibility than before.
- By introducing agents for low-level capabilities, the agent paradigm can be used to empower the factory's information sources that can most directly effect changes in the system [Warnecke, 1993].
- Agents can represent real physical devices or imaginary devices. The agent community can be used to simulate what might happen if the imaginary devices were actually installed in the factory.
- Agent systems are robust to change and easy reconfigurable. Since agents are able to discover the effect of the change on their own initiative and adapt to it, an agent system can organise and readjusts itself automatically in response to frequent change (instead of being explicitly reengineered to accommodate changes). Likewise, because an agent does not depend on external invocation to run, the system can be modified without the need to modify or even notify other components. Agents can be added and subtracted from the system while it is running without requiring external intervention.
- A system with autonomously functioning components will not collapse when one or more of the components fail or malfunction [Hatvany, 1985].
- A multi-agent heterarchy makes the vision of creating the virtual enterprises closer to the reality. It matches the common concept in the Internet or e-commerce communities where intelligent entities work together by communicating and coordinating with each other over wide area networks.

Why use agents to schedule?

- The use of the agent paradigm allows a large number of computers (processors) to be applied to the NP-complete scheduling problems (See Section 4.2.2.1. "Why the scheduling problem is difficult") in a balanced manner.
- Agent-based systems make it possible for scheduling to be performed in real time. There is no need for a system to wait for scheduling – a system schedules itself as it runs.
- Agents can develop schedules using the same mechanisms that are used by businesses in the manufacturing supply chain[1] [Baker, 1995].
- Multi-Agent Systems can exhibit evolving behaviour where the results of the whole system are more sophisticated than any of the pieces. This provides exciting new possibilities in factory scheduling.
- It is substantially less expensive and much more reliable from a hardware perspective to use a large number of inexpensive processors than a single processor having equivalent total processing capabilities.

Why use agents when developing software?

- Agent software can be less expensive to develop than centralised or hierarchically structured software [Duffie et al., 1986b].
- Agents are consistent with the object-oriented paradigm. The efficiencies of programming and all the advantages of the object-oriented paradigm are inherited.
- The software for each agent is much shorter, simpler, and as a result, is easier to write, debug, and maintain.
- Multi-Agent Systems can be designed to be self-configuring. In addition to decentralisation this feature makes these systems more robust and fault-tolerant.
- Multi-agent architectures are inherently scaleable and modular.

6.3.2 Types of agents and their application

In previous research on agent-based control and scheduling systems, agents took the form of different entities. For example, agents represented levels in a hierarchical decomposition of the factory [Butler & Ohtsubo, 1992, Parunak, 1987b, Tilley & Williams, 1992], resources [Baker, 1996, Shaw & Whinston, 1985c], or parts [Duffie et al., 1988]. In other research work [Parunak et al., 1997a, Parunak et al., 1998b] agent-based systems have been represented by communities of several classes of agents. In a case study of AARIA (Autonomous Agents for Rock Island Arsenal)[2] two groups of agents are distinguished:

- ***Persistent agents*** are agents whose behaviour does not change over the time scale involved in daily shop operation. This group is comprised of parts, unit processes, resources, managers, customers, and suppliers' agents.

---

[1] Agent-based systems can generate ***"system's schedules"*** for each work-part to be fabricated in the manufacturing system, in a similar manner to classical scheduling systems based on MRPII techniques. For example, the same ***backward scheduling principle*** can be used for generating work-parts schedules before their production processes start (off-line and in a static manner). Starting from work-part's due date, it is possible to design a multi-agent system in which agents (work-part and workstation agents or only workstation agents) determine a schedule for each work-part in the system through the process of negotiation.

[2] AARIA was a project that yielded an industrial-strength agent-based shop-floor control and scheduling architecture being developed for an Army manufacturing facility.

- ***Transient agents*** are agents that represent interactions among persistent agents and which go through a well-defined life cycle of behaviours as the shop operates. Engagement agents, material agents, product agents, and operation agents belong to this group.

One of the fundamental decisions in developing an agent-based system is the mapping the agents to the domain. In manufacturing scheduling and shop floor management, agents are typically mapped either to clusters of problems, functions, nodes in a manufacturing hierarchy, or shop-floor entities [Parunak et al., 1998b].

Clusters of problems. An agent architecture can be design in a way that each agent runs on a separate processor (solving a set of sub-problems that were obtained by decomposition of one "large" problem). Such a distribution increases the amount of computing power that can be brought to bear on the combinatorial nature of scheduling problems. For example, such mapping of agents is performed in the Cortes system [Sycara et al., 1991b]. The system treats the individual order coming into the shop as the basic unit, and distributes the orders across a set of identical agents. Each agent has two functions. The main function is to generate a schedule for its set of orders, and the second function is to serve as a monitor for one or more of the resources, and mediate requests for that resource from other agents.

Functions. Since agents can be seen as representatives and assistants for humans, in some architectures agents are identified with traditional functions in manufacturing (for example, agents for order acquisition, logistics, transportation, scheduling etc.). LMS (Logistics Management System) uses functional agents as critics to evaluate the priority of the next order to load into a tool in a semiconductor fabrication [Fordyce & Sullivan, 1994]. Separate agents assess the importance of the order from the perspective of the serviceability of goods made to order, daily planned output of goods feeding inventory, downstream pull, and current tool setup.

Nodes in a hierarchy. Several attempts have been made to assign agents on the basis of their time horizons (e.g. strategic, tactical, and real-time) and thus establish a hierarchy of agents. A couple of examples are YAMS (Yet Another Manufacturing System) [Parunak, 1987b] and BOSS (Bunch of OPIS-like Scheduling Systems) [Hynynen, 1989] which assign agents to the nodes of a hierarchy, representing the company as a whole, each plant, each cell within the plant, and each machine in a cell.

Shop floor entities. A large number of systems map agents primarily onto manufacturing entities such as parts, machines, and operations [Baker, 1996, Burke & Prosser, 1994, Butler & Ohtsubo, 1992, Duffie, 1990, Lin & Solberg, 1992, Shaw, 1989, Sikora & Shaw, 1997, Sousa & Ramos, 1998, Upton et al., 1991, Vaario & Ueda, 1998]. Smith and Becker offer comprehensive analyses of the kinds of entities that need to be considered in such a mapping [Smith & Becker, 1997].

6.3.3 How agents are connected?

Many systems provide a broadcast capability as a way of inter-agent communication. However, since communication paths among agents are determined by the configuration of an agent community, often a more restricted pattern that arises naturally from a system's internal logic can improve communication efficiency. In the review that follows, in addition to broadcast, three such communication mechanisms (configurations) are considered: hierarchy, mediator, and process flow [Parunak et al., 1998b].

Broadcast. In broadcast (non-directed) communication each agent propagates messages through the communication medium to all other agents in community. Such systems are described in [Vaario & Ueda, 1998] and [Fordyce & Sullivan, 1994].

Hierarchy. Hierarchical configuration is associated with the hierarchical agent mapping. In such a system, higher-level agents impose constraints on lower-level ones, and lower-level agents pass status information back to higher-level ones. We have already denoted two such systems YAMS [Parunak, 1987b] and BOSS [Hynynen, 1989].

Process Flow. In process flow configurations, the communication paths among agents follow the usual paths of information (business process paths) and material flow (shop-floor process paths) in the manufacturing enterprise. This configuration is followed in part or in whole by the systems that map agents onto entities in the manufacturing domain. In the Cortes system [Sycara et al., 1991b], the *cluster agents* follow a process configuration to the extent that their communication paths are determined by which resources they share. Sikora and Shaw [Sikora & Shaw, 1997] combine a process flow structure among agents with aggregation of agents representing manufacturing entities into higher-level entities. These higher-order entities communicate hierarchically, but the communications are generated bottom-up, not in a classical interchange of downward commands and upward status flow.

Mediator. An increasingly popular hybrid of hierarchical and process configurations is the mediator configuration, exemplified in [Maturana & Norrie, 1995, 1996] and [Interrante & Goldsmith, 1998]. These systems define two classes of agents: the "fundamental" agents that represent entities such as resources or parts, and "mediator" agents whose task is to oversee groups of entity agents and resolve conflicts among them. The mediators thus represent places of centralized conflict resolution or constraint optimisation reasoning. The hierarchical structure they impose makes the system more rigid and thus inhibiting of frequent change, modality emergence, and uniformity requirements.

6.3.4 Multi-agent architectures in manufacturing control systems

The backbone of a multi-agent scheduling system is its agent structure and the communication and control mechanism among agents. According to the agent structures, Multi-Agent Systems can be generally classified into the two major categories as discussed previously, namely, systems with a hierarchical structure and systems with a heterarchical structure.

6.3.4.1 Hierarchical structure

In a hierarchical structure, there is only one agent at the highest level whose responsibility is to supervise the agents at the lower levels so that the overall objective is achieved. Below this agent there are one or several levels, in which various kinds of agents are responsible for their own share of the work. Each agent in the intermediate levels reports its work only to its immediate supervisor, meanwhile monitoring its immediate lower-level agents. At the lowest level the agents usually perform the basic execution, calculation, or information collection tasks. Communication among agents at the same level is possible, but it is regulated by their supervisors. The hierarchical structure is presented, for example, in papers by [Gomes et al., 1994, Lathon et al., 1994, Maturana & Norrie 1996, Parunak, 1987b, Sycara et al., 1991b].

Maturana and Norrie used "mediators", a special kind of agent, to manage manufacturing and scheduling operations in an agent-based manufacturing system. [Maturana & Norrie 1996]. These mediators have a three-level architecture. The highest level is called template mediator who decomposes and integrates the tasks. The lower level includes a data-agent manager and an active mediator. Decomposed tasks are scheduled and dispatched to various resources through the coordination of these two types of agents. The lowest level relates to resource agents (controllers).

"MetaMorph I" system provides a framework for enterprise integration [Shen & Norrie, 1998]. The mediator agents assume the role of system co-ordinators by promoting co-operation among intelligent agents and learning from the agents' behaviour. Mediator agents are able to expand their capabilities to include mediation behaviours, which may be focused on high-level policies to break the decision deadlocks (similarly to the "staff holon/agent" in [Brussel et al., 1998]). Mediator agents can use brokering and recruiting communication mechanisms to find related agents for establishing collaborative sub-systems ('co-ordination clusters' or 'virtual clusters'; similarly to "co-operation domains" in HMS [Brussel et al., 1998]). In "MetaMorph II" [Shen et al., 1998] the framework of "MetaMorph I" has been applied at the enterprise function level, using a hybrid agent-based mediator-centric architecture to integrate partners, suppliers and customers dynamically with the main enterprise through their respective mediators within a supply chain network via the Internet and Intranets. In MetaMorph II, agents can be used to represent manufacturing resources (machines, tools etc) and parts, to encapsulate existing software systems, to function as system/subsystem coordinators (mediators), and to perform one or more supply chain functions. A prototype implementation has been reported with four mediators: enterprise (enterprise administration centre), design (integrates a functional design system), resource (co-ordinates an agent-based manufacturing scheduling sub-system), and marketing (integrates customer services).

Lathon et al. adopted a two-level structure [Lathon et al., 1994]. In the higher level is the coordination (global) agent (or so-called "globe agent") whose responsibility is to regulate the low level agents ("machine agents", or subsystem agents). Low level agents execute local scheduling.

Gomes et al. used a tree structure to represent the relationships between agents [Gomes et al., 1994]. At the top is the strategic agent who checks the global correctness of the schedules. Below it are two types of tactical agents: job tactical agents (JTA) and resource tactical agents (RTA). Each RTA's work is further facilitated by its sub-level operational agents who are in charge of local optimisation. In general, the main bottleneck in this system occurs at the coordinating agent.

Sycara et al. presented a model, which is another paradigm of a hierarchical structure [Sycara et al., 1991b]. In their model all agents make decisions without supervision and intervention, but they all share a common information base, called the coordination agent. The contribution of their research is that they selected what they believe is the most valuable information (so called "texture measures") as the basis for individual decisions.

Parunak presented Description of one of the earliest agent-based manufacturing systems - YAMS (Yet Another Manufacturing System) [Parunak; 1987b]. In this model each factory and factory component is represented as an agent. Each agent has a collection of plans, representing its capabilities. The Contact Net is used for inter-agent negotiation.

6.3.4.2 Heterarchical structures

In heterarchical structures [Baker, 1991, Duffie, 1990, Duffie & Prabhu, 1994, Lin & Solberg, 1992, Saad et al.; 1997, Shaw, 1987a, 1987b, Parunak et al., 1998b, Wang & Usher, 2002] various types of agents have no control over one another. They simply exchange information and negotiate when conflicts between scheduling decisions occur. There is no "information, computing, and control centre." In this scheme the information and workload are more balanced because of the distributed computation, but the coordination is more difficult to regulate efficiently. Most market-like agent systems have such structure. For example, [Lin & Solberg, 1994] modelled the manufacturing floor shop exactly like an economic market place. Each task agent enters the market carrying certain "currency." It bargains with each resource agent for the

machines on which it can be processed. Similarly, each resource agent competes with other agents to get more "valuable" jobs. Many decentralised models use Smith's Contract-Net Protocol for communication and scheduling, as outlined in Section 6.2.3.1 "The Contract-Net Protocol".

6.3.4.3 Hybrid structures

Between a hierarchical and a heterarchical structure, some authors adopted a mixed architecture. [Ramos, 1994] proposed a structure in which task agents and resource agents are coordinated by a task manager and a resource manager respectively, but the two managers are independent of each other. [Brennan et al., 1997] proposed a Partial Dynamic Control Hierarchy by combining both hierarchical and heterarchical architectures. The use of partial dynamic hierarchies assists reconfigurability and can provide a better system performance than either of the single control approaches. [Choi et al., 2000] described a hybrid control system with a shop floor activity methodology called Multi-Layered Task Initiation Diagram. The architecture of the control model identifies three levels; i.e. the shop floor controller, the intelligent agent controller and the equipment controller. In addition, holonic control was proposed to combine the best of hierarchical and heterarchical control, to provide a high performance under disturbances [Bongaerts et al., 1999].

6.3.5 Scheduling in distributed multi-agent heterarchical control systems

In both the centralised and hierarchical approaches (discussed in Section 5.4 "Overview of manufacturing control architectures and their main characteristics") scheduling and control processes are treated separately de-emphasising the relationship and dynamics among them. A predefined schedule is usually valid for a short planning period. This schedule can be invalidated for any reason (machine breakdowns, operator's errors, late supplies, etc.), and at this point it will require a rescheduling process. Due to frequent changes in the system and the fact that schedulers deal with models of factories that cannot completely reflect the actual current situation in the factory, scheduling or rescheduling actions often yield unrealistic schedules [Parunak, 1991].

To produce more realistic schedules, in a heterarchical control system scheduling and control processes are more integrated, autonomous entities are loosely coupled so that relaying of global information is minimised, and ***the principle of last commitment*** (the establishment of relationships between distributed entities must be delayed to as late as possible and terminated as soon as possible) is followed.

The main idea behind the scheduling in heterarchical control systems could be generally described as follows. *"Make distributed manufacturing units autonomous and "intelligent" enough to perform planning, scheduling and control decisions locally and allow the overall system behaviour to emerge as a result of co-operation among these entities".*

Generally speaking, in heterarchical control systems scheduling can be seen as a process consisting of two components: global and local scheduling.

Global scheduling refers to scheduling jobs among autonomous entities. The goal of global scheduling is to optimise the performance of an entire manufacturing facility. Two approaches can be distinguished in global scheduling of heterarchical control systems:

- ***Reactive scheduling*** – in which autonomous entities act on a purely reactive basis, without forecasting. In such a scheduling approach, each entity generates its local schedule based on local information (which is incomplete) and without considering the

global effects of the local schedule. For that reason, in reactive scheduling there is a danger of bringing a manufacturing system into a state of "anarchy".

- ***Co-operative scheduling*** – in which the autonomous entities are able to modify their local decisions and plans so that the global performance of the system is improved. To optimise the system performance in co-operative scheduling, entities must be able to plan and evaluate their local plans with a global perspective [Prabhu, 2000]. Entities must modify their plans based on such evaluations and they must be willing to trade-off local performance to improve the global performance. Co-operative scheduling, hence, requires the *existence of an entity capable of evaluating the global merits of local schedules.* Entities generate tentative schedules and send them on for evaluation. If global performances are not improved the local entity should generate another schedule in an attempt to improve the global performance of the system as a whole. These new local schedules would be less "locally optimal" but would be likely to be more "globally optimal".

Local scheduling refers to the scheduling of jobs inside autonomous entities. The goal of local scheduling is to optimise the performance of individual autonomous entities. Quality and speed in providing good schedules is significantly increased in this case because the scheduling problem is enormously reduced, the necessary information is available locally, and the flow of data is very fast. There are basically two types of scheduling methods for each individual agent:

- ***Deterministic methods***, in which each agent chooses jobs using local information (present or predicted data) in a deterministic manner.
- ***Stochastic methods*** use stochastic learning to improve the quality of schedules [Daouas et al., 1995]. Stochastic methods include, for example, simulated annealing that was discussed in Section 4.3.2.5.2 "The simulated annealing optimisation technique".

How the principles of global and local scheduling are applied in this project is discussed in Section 10.3. "Scheduling in the model of a distributed heterarchical shop floor control system".

6.3.5.1 Agent-based dynamic scheduling

In an agent-based dynamic scheduling problem, agents are typically used to represent each resource and job. The job agent associated with a job will announce its requirements for the next operation to those resource agents that have the potential to perform that operation. The resource agents who receive the announcement message will respond with a bid message to the job agent. All the bids submitted for the job's next operation will be evaluated by the job agent based on a set of heuristics. Once bid evaluation is finished, one resource will be selected and awarded a contract to perform the operation. The above bidding procedure is the core of the Contract-Net Protocol. Bidding schemes based on the Contract-Net Protocol may differ in such aspects as the timing of message exchanges involving announcements and bid collection, information reported within the bid, and the rules used in bid evaluation. Some examples of the application of agent-based dynamic scheduling are as follows.

Shaw employed the contract-net method for dynamic scheduling in cellular manufacturing systems [Shaw, 1988]. In his approach, when an operation of a job at a cell is finished, the cell's control unit will make the decision regarding which cell the job should visit next. To do that, the cell's control unit broadcasts the task announcements to the other cell control units. The cell control unit which received a task announcement checks if the required operation is within its capability and submits its estimation on the earliest finishing time (EFT) or shortest processing time (SPT). There is no job agent in this case. Each job's route is determined through the negotiation between the cells. Shaw's experimental results indicated that the bidding scheme

with EFT (earliest finishing time) outperformed the bidding scheme with SPT (shortest processing time).

Lin and Solberg presented the integrated flow control framework which employs a market-like system model, where a generic bid construction mechanism based on a combination of price and objective mechanism is used [Lin & Solberg, 1992]. The integrated flow control framework is part-centred, and tries to find critical resources based on a dynamic resource unification scheme. The framework follows the data flow model of Lewis [Lewis, 1981, Lewis et al., 1987]. Under the data flow model, machines select jobs according to a simple dispatching rule (first-in-first-out); jobs are routed to the first machine available to complete the next task to be performed on them. No supervisory control is applied. In Lin and Solberg's agent-based shop floor scheduling system, each job agent with its unique set of weighted objectives enters the system with some currency and alternative process plans. To achieve the objectives, job agents try to fulfill the processing requirements by bargaining with resource agents. Each resource agent sets its charging price based on its status. The part agent tries to minimise the price paid, but the resource agent's goal is to maximise the price charged. Each deal is completed once the part agent and resource agent are mutually committed. One important feature of this market-like mechanism is that the negotiation among agents is invisibly guided by an adjustable price to improve the system performance. Lin and Solbergs' results essentially showed that their system was able to handle unexpected resource failures and part objective changes. Lin and Solberg later presented a manufacturing simulation system based on the dynamic price mechanism for agent negotiation [Lin & Solberg, 1994]. The proposed agent-based framework simplifies implementation of different negotiation strategies in manufacturing systems.

Saad et al. proposed a contract-net-based heterarchical scheduling approach for flexible manufacturing systems [Saad et al., 1997]. In their study, two scheduling mechanisms were tested. The first is the ***Production Reservation (PR)*** method where all the operations of a job are scheduled completely at the one time when it arrives at the system. The other method, referred to as ***Single Step Production Reservation (SSPR),*** schedules one operation at a time with the job agent delaying negotiation of its next operation until the current operation is finished. In the Contract-Net Protocol, a job agent selects the machine that can finish processing the required operation first. If at least two alternatives are tied for this criterion, the job agent will choose the machine with fewer jobs in its reservation list. They compared the PR and SSPR approaches with some traditional dispatching rules. Their results showed that PR outperformed the traditional dispatching rules, while SSPR only outperformed PR on average tardiness. However, unexpected events such as machine breakdowns or emergent jobs were not considered in their experiments. Otherwise, SSPR will take the advantage in the face of these uncertainties.

In [Kanchanasevee et al., 1997] authors suggested the development of Augmented Bidding Production Reservation Scheduling (BPRS) scheme for use in a realistic holonic manufacturing test-bed system. The system had heterarchical architecture and was comprised of 5 holons types: product machine, scheduler, computing, and negotiation holon. Augmented BPRS is combination of the BPRS scheme (which uses a Production Reservation (PR) scheme with a contract-net negotiation mechanism), with the local dispatching rules, Early Due Date (EDD) and Shortest Processing Time (SPT). The difference between Augmented BPRS and BPRS is in terms of when products are released to the input buffer of the machine. In the case of BPRS, parts are held back till they are ready for processing and the allocated sequences for jobs are not changed. In the Augmented BPRS, order of jobs in a queue may be changed based on the EDD or SPT dispatching rules.

Ouelhadj et al. [Ouelhadj et al., 1998] presented a negotiation strategy similar to the approach of Shaw [Shaw, 1988]. The resource agent is responsible for establishing the negotiation with other

resource agents in order to select the most appropriate resources to allocate to the specific task operations. The PR method was employed in their study. Ouelhadj et al. [Ouelhadj et al., 1999] extended the Contract-Net Protocol to a Multi Contract-Net Protocol. It provided the function of scheduling several tasks simultaneously. Their experimental results showed that the time required to schedule operations with this approach and the run time including scheduling and execution both are linear rather than exponential with the increase of the number of scheduled tasks.

Ottaway and Burns proposed an agent-based negotiation involving a currency scheme [Ottaway & Burns, 2000]. In their model, the amount of currency that a job agent carries is based on the job's objective function that is a weighted linear combination of time, cost, and quality. The resources determine the amount of currency to be charged for their production services based on their capabilities and the demand for their services. It is noted that there is an incentive factor for preventing a job from being stuck in the system due to a lack of currency. This factor is used to increase the budgeted funds for the jobs that kept failing in the bidding process. Ottaway and Burns also addressed the importance of using supervisor agents to balance the production load and maximize overall throughput. The supervisor agents essentially played a key role for dynamically switching the system structure between a hierarchy and a heterarchy.

Dewan and Joshi developed an auction-based scheduling mechanism for a job shop environment [Dewan & Joshi, 2001]. They also used currency as a means for agent negotiation. Their market-like approach differed from the others [Lin & Solberg, 1992] in using Lagrangian relaxation to decompose the problem formulation. Whenever a machine agent is available, it announces an auction for time slots from the current time to the end of the time horizon. Each job agent will bid for the time slots with the cost that they are willing to pay. The job agent's goal is to minimize cost, while the machine agent uses the submitted bids for price adjustment. If more than one job demands the same time slot, the price for that slot will increase. The price adjustment and bid calculation continue iteratively until the price converges. The machine agent determines the best bid for the earliest time slot as the next operation. After processing is finished for that operation, the above auction procedure is executed again. Dewan and Joshi used the above mechanism to schedule the jobs with different objectives [Dewan & Joshi, 2001].

Wang and Usher proposed a factor called a resource collaborative factor that essentially provided some degree of cooperation among resource agents when used within the context of the Contract-Net Protocol [Wang & Usher, 2002]. This additional factor enhanced a job agent's ability to make decisions on routing selections taking into account a more global perspective. Also, the factor gave an indication of the importance of a resource to the scheduling function providing a glimpse as to its potential of becoming a future bottleneck. These advantages are demonstrated through results of a job shop simulation that indicated enhanced system's overall performance, in particular when the system was heavily loaded.

In summary there have been a number of applications of agent-based dynamic scheduling in theoretical evaluations. These examples provided the theoretical basis for the system developed in this thesis. Also, it is acknowledged that a limited number of research projects have evaluated to the level of prototype applications or full production installations. A brief review of a few agent based projects in domain of manufacturing scheduling, control, design collaboration, and agent simulation is provided in [Parunak, 2000c]. Examples include the CEC (Centre for Electronic Commerce) agent-based systems; Daimler Chrysler projects: Fakos, KoWest, Holomobiles, MASCADA (Manufacturing control Systems capable of managing production Changes And Disturbances); Deneb Robotics project: ANTS (Agent Network for Task Scheduling); ObjectSpace and Advanced Micro Devices project: AEMSI (Agent-Enhanced Manufacturing System Initiative); ERIM CEC and Ward Synthesis project RAPPID (Responsible Agents for Product-Process Integrated Design); ERIM CEC project DASCh

(Dynamical Analysis of Supply Chains); and so forth. The system developed in this thesis is distinguished from these by its simplicity, practicality and very low development cost, resulting in a cost effective solution to job shop scheduling problems. Most of the other systems have been developed as multi million dollar projects and the cost is commonly very high and too expensive to be afforded by smaller manufacturers.

6.3.6 Some advantages and disadvantages of applying Multi-Agent Systems in manufacturing

This section discusses some of the advantages and disadvantages of implementing multi-agent heterarchical systems in manufacturing. Firstly, Table 6-4 [Parunak, 1994] briefly contrasts the two philosophies: autonomous agent systems and traditional approaches while more details are provided in the following sub-sections.

6.3.6.1 Benefits of distributed, multi-agent, heterarchical architectures

Because of the underlying heterarchical architecture, distributed scheduling in Multi-Agent Systems has several important benefits such as reduced complexity, ability to schedule in real-time, high flexibility, and high fault-tolerance. These benefits and a few issues that can be successfully resolved by employing distributed heterarchical scheduling and control systems are discussed in the remaining text of this section.

Table 6-4. Agent-based vs. conventional technologies

Issue	***Autonomous Agents***	***Conventional***
Model	Economics, Biology	Military
Issues favouring conventional systems:		
Theoretical optima?	No	Yes
Level of prediction	Aggregate	Individual
Computational Stability	Low	High
Issues favouring autonomous agents:		
Match to reality	High	Low
Requires central data?	No	Yes
Response to change	Robust	Fragile
System reconfigurability	Easy	Hard
Nature of software	Short, simple	Lengthy, complex
Time required to schedule	Real time	Slow

6.3.6.1.1 Reduction of the complexity of scheduling problems

As noted in Chapter 4, in traditional approaches to job shop scheduling systems the complexity of scheduling problems increases exponentially with the size of the system and often the solution of such a complex problem cannot be contemplated all at once. Following the principles of heterarchical systems, the scheduling task in distributed scheduling is decomposed among the system entities (agents) and each entity generates its own schedule. Therefore, the system level

scheduling problem is decomposed into smaller and simpler tasks. Also, since there are no strong relationships among entities, adding an entity does not increase the combinatorial complexity of other entities.

6.3.6.1.2 Ability to achieve Real-time scheduling

Ideally, a schedule for a complete shop floor is generated once and is executed perfectly, but this is rarely true. In this static approach to scheduling, an optimal schedule (generated after considerable effort) may rapidly become unacceptable because of the uncertainties on the shop floor and a new schedule has to be generated to restore performance. In the worst case, if the input information changes before the schedule is completely generated, the schedule will no longer be effective or useful even though it is not yet fully generated. Low effectiveness of optimised scheduling has been a major concern.

The best way to deal with uncertainties is to prevent them, but this may not be always possible. Alternatively, it may be possible to generate an acceptable, i.e. "good", schedule with significantly less effort, and such that, this schedule can be continuously improved as it is executed. This approach can be beneficial because it reduces the complexity in scheduling. Moreover, a "good" schedule can be generated more frequently because it takes less effort to generate it. This increased frequency will result in increased effectiveness because the actual events would tend to conform to the expected events in the schedule.

If "good" schedules are generated at a rate much greater than that at which uncertainties occur in the system, then we have a case of ***real-time scheduling***. Such schedules can be expected to be highly effective and will lead to an acceptable system performance. In addition to coping with uncertainty, real-time scheduling will result in strategic benefits due to reduced scheduling lead-time.

The real-time scheduling capability of a distributed heterarchical control system strongly depends on the storage and flow of information in the system. Because most of information that is needed for local scheduling is locally available in heterarchical systems, the flow of information is very fast (often through the sharing of common memory space). Also, because of the high level of autonomy of the distributed components, the complexity of the scheduling problem is significantly reduced. These two characteristics make generating *local schedules* easily achievable in real time. On the other hand, in *global scheduling*, because of the need to exchange information among the distributed entities via a network, the time to find a good schedule may be several orders of magnitude longer. However, the available time for finding a good global schedule is usually much longer (in comparison with the time required to perform an operation at an individual workstation) so real-time scheduling as defined in the paragraph above may be possible.

6.3.6.1.3 Increased flexibility

The ability of a manufacturing system to demonstrate its flexibility depends to a large degree on the manufacturing control system. This flexibility is created in two phases. Firstly, the system is designed to be flexible and then it is managed so as to achieve that flexibility. Installing the appropriate production equipment makes it possible to achieve the first phase. Achieving the second phase, which includes achieving process flexibility, operational flexibility, routing flexibility, and volume flexibility (refer to Section 2.1.1 "A brief review of development of manufacturing systems with emphasis on job shop production" for more details on different types of flexibility perceived in manufacturing systems), will depend primarily on the characteristics of the manufacturing control system.

6.3.6.1.4 Increased fault-tolerance

Local autonomy and reduction of global information enhances the attainment of implicit fault-tolerance and reduces the need for additional, explicitly programmed fault-tolerance. When a fault occurs in a heterarchical system it tends to be confined to the entity level and local intelligence of the entity is used to recover from it (breaking a cutting tool is the classical example). If the system prevents faults from propagating, it also confines the schedule disruption to the affected part of the system. This translates into a reduced rescheduling effort and a simpler scheduling system. However this assumes, for example, that additional capacity is available to cope with machine breakdown or non-availability of resources, or it is possible to adjust the due dates, or "catch up" at a later stage, and so forth. In *a hierarchically controlled system, the fault is propagated down the stream in the control hierarchy.* The outcome of this malfunction could be much more severe, since a greater amount of work would be affected. For such work, finding "spare capacity" in the system would be harder than in the case when only one workstation is out of order. Investing in redundant (parallel) control units is considered to be an effective but costly solution. Decisions about what level of redundancy should be incorporated in a system design are an important system design consideration.

6.3.6.1.5 Reduction of software related costs - reduction of overall size and complexity of the production planning and control system

More than half of the cost associated with computer controlled manufacturing equipment is in the preparation and maintenance of its software. A high degree of system decomposition and autonomy of production units ultimately influences software development. The control software is less complex and "more repetitive" (most of modules developed in one agent are reused in others) so that high costs related to development and maintenance of the software are significantly reduced.

From the production planning and control point of view the ultimate expectations are that heterarchical control systems will contribute to reduction of the overall size and complexity of the system required for detailed material and capacity planning, controlling and monitoring of production activities in job shop manufacturing. Ultimately, costs are expected to be significantly reduced in domain of production planning and control software. Since early MRP software products were too limited and too brittle in the face of the dynamic environment of the factory floor, software vendors have tried to keep adding functionality to meet the problems of managing a complex operation. That led to Enterprise Resource Planning (ERP) software that becomes so complex that successful implementations require a multi-million dollar multi-year effort. Scaling these complex software systems up to handle the greater complexity of an entire supply chain (that often includes small manufactures with little or no software staff) is extremely difficult if not impossible. On the other hand, small grained, agent-based systems offer promising alternatives to monolithic software modules being developed today [Sauter & Parunak, 1999]. Agents that respond to their environment using simple rules and interacting directly with other agents through predetermined protocols offer management software that can handle complex dynamic systems while being much simpler to construct and manage.

6.3.6.1.6 Increased system responsiveness

Prompt response to changes in manufacturing processes is an important factor in the competitiveness of modern manufacturing. One of the main characteristics of Multi Agent Systems is their ability to respond rapidly to unanticipated change. By moving decisions points closer to manufacturing processes, such as is the case in Multi-Agent Systems, more dynamic responses to changes in demand, disturbances in material flow, and changes in the real-time availability of materials, machines, and tools can be achieved.

6.3.6.1.7 Increased utilisation of system resources

Due to the high capital cost involved in installing an FMS, a high rate of efficient utilisation of resources is needed to ensure an early return on investment. Balancing the workload of machines can be effectively achieved by developing sound planning, scheduling, and monitoring strategies in heterarchical distributed systems and through the negotiation process conducted among agents i.e. corresponding autonomous production units.

6.3.6.1.8 The ability to control large manufacturing systems

Traditional control architectures, both centralised and hierarchical, are not feasible ways to control large manufacturing facilities because beyond a certain level of complexity they are virtually impossible to implement. Designing and constructing large centralised and hierarchical control systems requires a considerable amount of effort, these systems are not adaptable to new production requirements and unexpected dynamic changes in manufacturing processes, and finally, they become prohibitively complex and unreliable with the increasing size and complexity of the manufacturing system. On the other hand distributed heterarchical control systems offer an approach that is capable of controlling manufacturing systems that are comprised of, theoretically, an infinite number of machine tools. The control architecture is reduced to a network of small, modular reusable control elements.

6.3.6.1.9 The ability to deal with partial information

The scheduling process in distributed heterarchical systems is capable of coping with situations when: 1) data loads, in terms of type, quality, and availability of requested information, are not complete and 2) when there is not enough time to tune or adjust the complete schedule. In some situations the scheduling process needs to deal with urgent issues by updating only part of the schedule and delaying the updating of the rest of the schedule.

6.3.6.2 The main drawbacks of distributed heterarchical control systems

Section 6.3.6.1 "Benefits of distributed, multi agent, heterarchical architectures" addressed several benefits of distributed heterarchical shop floor control systems. However, there are a number of disadvantages that have to be confronted with the introduction of heterarchical systems. This section points out the main disadvantages.

6.3.6.2.1 Prediction of how the system will behave can be made only at the aggregate not at the individual level

One of the justifications for agents is that the whole system can be more than the sum of its parts. A collection of relatively simple agents can yield surprisingly rich and complex interactions. However, this potential represents a two-edged sword. While such distribution of control reduces the complexity of the manufacturing control system and ensures loose coupling of entities, it becomes difficult to predict the behaviour of the overall system which is often not obvious from the outset, and can challenge the viability of the application if it is not managed appropriately. The security and trust in MASs that the agents will handle the tasks the way the users expect the agents to perform, are the main concerns among business managers considering applying heterarchical, agent based systems [Parunak, 1996a].

6.3.6.2.2 Theoretical optima cannot be guaranteed

In traditional centralised systems, in theory the "master" entity resolves all resource sharing conflicts. It has complete knowledge of the state of the system including the schedules of all entities, all variables, and all relations between these variables. It generates schedules for its

"slaves" by using complete information about the system and considering the interactions between its "slaves". This way of generating schedules makes the "master" very complex but the schedules are globally coherent. An entity's action is said to be *globally coherent* if the action simultaneously satisfies both the entity's and the overall system's performance requirements [Prabhu, 2000]. Hence, in a centralised system global coherence can be achieved because of the large amount of global data and strong master-slave relationships, but at the cost of significant complexity and a high reliance on the accuracy and completeness of the information.

In a distributed heterarchical control system entities know neither the plans nor the intentions of other entities. This minimises the global data and ensures loose coupling between entities resulting in high autonomy and fault-tolerance. These are key principles of heterarchical systems. Incomplete information combined with high local autonomy makes it difficult for heterarchical system entities to ensure that their local decisions are globally coherent. Explicit coordination among entities can be expected to be complex and will compromise the primary objective of reducing complexity in the system. However, while an entity is not explicitly aware of the existence of the other entities in the system, it becomes implicitly aware of them when it shares resources and information with them. Also, it should be remembered that the optimum schedule, even if it could be computed by conventional systems may not be realisable in practice – since it is often invalidated by the actual conditions in manufacturing operation at the time of production. It is important to note that the agent based system developed for this project can provide monitoring data that will enable shop supervisors to become aware of problems and take the appropriate action reducing the risk of instability. This is discussed further in Section 14.4.4 "Achieving global coherence".

6.3.6.2.3 System of autonomous agents can become computationally unstable

Some researchers believe that a manufacturing system can evolve to such a stage, that a heterarchical control system may not produce a satisfactory solution. This may occur, for example, when many workstations in the system malfunction almost simultaneously and when the demand for shared resources increases dramatically during that period. In such circumstances, when communication among agents is significantly increased, it is believed that a heterarchically-controlled system may become computationally unstable (in an attempt to find a solution, the system can enter infinite "negotiation loops"). Worse still, the entire system could collapse, that is, the network could become quickly overloaded. This project partially addresses this issue and it is discussed in Section 12.2 "Contribution of the study".

6.4 Overview of standards and tools for developing Multi-Agent Systems

Wide use of agent technology in industry depends on the availability of development tools and platforms that protect developers from the need for developing basic functionality with each system. Such tools and platforms, in turn, presume the existence of standards that reflect the agreement of researchers and developers on what that basic functionality should be and how it should be presented. Some of these tools, standards and platforms that are most commonly used for developing agent systems, are addressed in this section.

6.4.1 Agent standards

Applications of agent technology will develop and flourish only if it is possible for agents developed by different companies and/or using different software platforms to communicate

reliably and effectively. Single-supplier solutions will not be acceptable. Standards will therefore be required that ensure "open" interaction between heterogeneous communities of agents. While several groups have identified the need for such standards, their production is currently a research activity. For example, both DARPA and FIPA have identified specific types of agent standards they feel are needed. These include: *the agent reference model* (agent management and control, privacy, security, access control, and veracity), *agent-to-agent communication* (communication languages, coordination protocols, security protocols) and *agent-software communication* (communication between agents and non-agent software). Although research groups in various companies have produced working Multi-Agent Systems, many issues are at present too poorly understood for reliable, well-specified standards to be generated [Steventon, A., 1998].

However, two consortia have emerged focusing on formalising standards specifically in support of agents: FIPA and MASIF.

6.4.1.1 FIPA

The Foundation for Intelligent Physical Agents (FIPA) was formed in 1996 as a non-profit organisation registered in Geneva, Switzerland [FIPA, web] with the purpose of promoting the development of specifications of generic agent technologies that maximise interoperability within and across agent based applications that run on heterogeneous agent-based systems. To support this, FIPA is working (on the basis of ideas adopted from KQML [Finin et al., 1997] and *belief, desire and intention - BDI* principles [Bratman, 1988]) on specifications that range from agent architectures, communications and content languages for expressing messages, to the interaction protocols which expand the scope from single messages to complete transactions. In addition, standards for specific applications, such as a 'wrapper agent', are provided.

The core mission of FIPA is to facilitate the interworking of agents and agent systems across multiple vendors' platforms. This is expressed more formally in FIPA's official mission statement: *The promotion of technologies and interoperability specifications that facilitate the end-to-end interworking of intelligent agent systems in modern commercial and industrial settings.* The core message is that through a combination of speech acts, predicate logic and public ontologies, FIPA can offer standard ways of interpreting communication between agents in a way that respects the intended meaning of the communication. This is much more ambitious than, for example, XML, which only aims to standardise the syntactic structure of documents.

FIPA has produced three sets of standards so far: FIPA 97, FIPA 98 and FIPA 2000

The FIPA97 specifications define:

- ***Normative specifications*** for agent management (agent platform services), agent communication language (based on speech acts and formal semantics; it includes several predefined protocols such as: contract-net negotiation and auction protocols) and agent-software integration; and
- ***Several reference applications*** (such as: personal travel assistance, personal assistant, network provisioning and management, and audio/video entertainment and broadcasting)

FIPA98 and FIPA 2000 are extending these specifications, including work on: agent management support for mobility, an ontology service, and additional applications (for example, product design and manufacturing agents). More information is available on http://www.fipa.org/repository/fipa2000.html.

The FIPA standard provides mandatory and informative specifications about a wide variety of topics and issues that go further than the pure aspects of inter-agent communication. In this

respect FIPA offers a set of application independent specifications that are extremely useful for the conceptual design of the agents and their interactions and entire Multi-Agent Systems.

Furthermore, the FIPA choice to adopt formal languages of a high level to encode the contents of messages allows the transmission of complex data structures in a way which is independent of the specific programming languages used to implement the agents. FIPA does not currently constrain the low-level implementation of agents to any great extent, nor does it constrain the infrastructure (defining lower-level components as agents), except for defining agent platform services.

6.4.1.2 MASIF

The OMG's (Object Management Group) MASIF (Mobile Agent System Interoperability Facility) focuses on inter-agent communication between mobile agents developed on CORBA (Common Object Request Broker Architecture)[1] platforms and does not support or standardise high-level interoperability among non-mobile agents on different agent platforms.

MASIF regards the mobility of the agent to move from one location to another as the key characteristic of an agent. MASIF defines a general agent reference model and interfaces to support agent management, agent transfer and agent tracking functions. The MASIF specification discusses its relationship with some of CORBA services but does not represent a thorough integration of agent technology into CORBA. The MASIF interfaces are defined at the agent system level, rather than at the agent level. Agent systems and agents may be, but are not required to be, CORBA objects. However, when agents *are* defined as CORBA objects, they potentially have access to all CORBA services, any legacy software wrapped by CORBA objects, etc. MASIF discusses agent management functions and general agent reference models (as FIPA does) and address agent-software communication in terms of Remote Procedure Calls (RPC) or Remote Method Invocation (RMI). The FIPA and MASIF work are somewhat complementary, in that FIPA has so far mainly been concerned with high-level agent to agent communication (ACL, negotiation protocols, ontologies), and has not said much about mobility, while MASIF has primarily considered mobility and does not deal with agent-to-agent communication at all (at either high or low level). Both FIPA and MASIF steer away from trying to overly constrain agent implementation technology at this early stage in its development. There is a great deal of potential synergy between the FIPA and OMG activities. More information on MASIF is available at http://www.omg.org.

6.4.2 Agent development platforms

For wider use of agent technology in the development of agent based manufacturing systems, powerful agent development tools are strongly needed. Recently, a number of tools have been reported, some of which are already commercially available. Examples include:

ABS (Agent Building Shell) was developed in Enterprise Integration Laboratory of the University of Toronto especially for developing cooperative enterprise agents [Barbuceanu & Fox, 1995, 1996]. It is being used to develop multi-agent applications in the area of manufacturing enterprise supply chain integration.

---

[1] CORBA is an architecture that enables objects, to communicate with one another regardless of what programming language they were written in or what operating system they're running on. CORBA was developed by an industry consortium known as the Object Management Group (OMG), http://www.omg.org.

ObjectSpace's ***VoyagerTM*** product provides a Java based Object Request Broker (ORB) designed for mobile agents. [ObjectSpace, 1997].

Gensym's ***ADE (Agent Development Environment)*** builds on its intelligent manufacturing software development environment G2 [Gensym, 1997]. ADE has been applied to the development of agent-based systems for supply chain management [Mehra & Nissen, 1998].

Other general agent development tools include IBM's ***Aglets SDK*** [IBM, 1998], General Magic's ***OdysseyTM*** [General Magic, 1997], and ***OAA (Open Agent Architecture)*** [Martin et al., 1998], Stanford's ***JATLite*** [Stanford, 1997], AARIA team's ***Cybele*** [Baker et al., 1997], and ***DESIRE*** [Brazier et al., 1998].

Some of the major publicly available implementations of agent platforms which conform to the FIPA specifications are presented in subsequent sections. More information on these and other platforms that are complaint with the FIPA specifications such as: Agent Development Kit (ADK), April Agent Platform, Comtec Agent Platform, Grasshopper, JAS (Java Agent Services), LEAP (Lightweight Extensible Agent Platform), and JACK Intelligent Agents, can be found at [FIPA, web].

6.4.2.1 FIPA-OS

FIPA-OS was the first Open Source implementation of the FIPA. Dedicated developers from around the world have contributed to numerous bug fixes and upgrades, leading to over 10 formal new releases. FIPA-OS now supports most of the FIPA experimental specifications currently under development. With the new in depth developers guides, it is an ideal starting point for any agent developer wishing to benefit from FIPA technology. FIPA-OS 2 is a component-based toolkit implemented in 100% pure Java. FIPA-OS is design to run on the computer systems that have Java virtual machine [FIPA, web].

6.4.2.2 JADE

JADE simplifies the development of multi-agent applications, which comply with the latest FIPA 2000 specifications. While appearing as a single entity to the outside world, a JADE agent platform can be distributed over several hosts. Agents can also migrate or clone themselves to other hosts of the platform, regardless of the OS. The life cycle of agents can be remotely controlled via a GUI, which also allows debugging tools to be started. The communication architecture tries to offer (agent transparent) flexible and efficient messaging by choosing, on an as needed basis, the best of the FIPA-compliant Message Transport Protocols (MTP) that are activated at platform run time. JADE is implemented in version 1.2 of JAVA and has no further dependency on third-party software. JADE is design to run on the computer systems that have a Java virtual machine.

6.4.2.3 ZEUS

ZEUS is an Open Source agent system entirely implemented in Java, developed by BT Labs and can be considered a toolkit for constructing collaborative multi-agent applications. Zeus provides support for generic agent functionality and has sophisticated support for the planning and scheduling of an agent's actins. Moreover, Zeus provides facilities for supporting agent communications using FIPA ACL as the message transport and TCP/IP sockets as the delivery mechanism. Zeus also provides facilities for building agents in a visual environment and support for redirecting agent behaviour. The Zeus approach to planning and scheduling involves representing goals and actions using descriptions that include the resources they require and the pre-conditions they need to be met in order to function. This allows goals to be represented using a chain of actions that have to be fulfilled before the goal can be met. This action chain is built

up using a process of backwards chaining. ZEUS can operate on the computer systems that have Java virtual machine installed but also can operate on Windows (95/98/NT4 2000, XP) and Solaris platforms.

6.4.3 Agent languages

Agent Communication Languages (ACLs) have been developed to provide a way for agents to communicate with each other supporting cooperation in Multi-Agent Systems. Most of the current proposals for ACL adopt protocols that use the speech act theory [Searle, 1969] as a basic model of communication.

To implement speech acts, a set of message types, so-called "performatives[1]" is provided. Performatives can be considered as atomic message operations, are general and have a clear semantics. Performatives can be ***actions acts*** (as they are intended to perform some action by virtue of being sent) or ***communicative acts*** (verbs that tell a receiving agent in which context to interpret the contents of the enclosed message). The message that is supplied with a performative is itself wrapped in a well-specified envelope, called an ***agent communication language (ACL)***. An ACL provides mechanisms for adding context to the performative and the message content, such as identifying the sender and receiver, the ontology, and interaction protocol of the message. The ontology (also referred as a content language) is used to express the actual content of a message. The set of ***interaction protocols*** describe entire conversations between agents for the purpose of achieving some interaction or effect, such as, auctioning, issuing a call for proposals, negotiating brokering services and the registration and deregistration of subscriptions.

The most common ACLs are [Labrou, 2001]:

- KQML (Knowledge Query and Manipulation Language) conceived in the early 90's [Finin et al., 1997], and
- FIPA-ACL [FIPA, web].

The goal of these languages is to support high-level, human like communication between intelligent agents, exploiting knowledge-level features rather than symbol-level ones.

The KQML (Knowledge Query and Manipulation Language) [Finin et al., 1997] is an evolving standard agent communication language for exchanging information and knowledge among agents. KQML was aimed to develop techniques and methodology for building large-scale distributed knowledge bases which are sharable and reusable [Ferber, 1999]. KQML was one of the first initiatives to specify how to support the social interaction characteristic of agents and is now one of the most pervasive ACLs. KQML defines the allowed operations (message types) that agents may attempt to perform during the knowledge sharing process and is indifferent to the actual content of the message and the format of the information itself. KQML implementations have used standard communication and messaging protocols such as TCP/IP, email, and HTTP. The most common content language used in KQML for specifying the content of the message is KIF (see Section 6.4.4 "Content (representation) languages - ontologies"). KQML is not a true de facto standard in the sense that there is no consensus in the research community on a single specification (or set of specifications). As a result, variations of KQML exist and different agent systems which speak different dialects may not be able to

[1] Speech act theory is used to define the semantics of messages. Although there are several hundred verbs in English, which correspond to performatives, the ACL defines what is considered to be the minimal set for agent communication. FIPA ACL specifies a number of communicative acts (more than 20), such as, *request*, *inform* and *refuse*, in a well-defined manner that is independent from the overall content of the message.

interoperate fully. (Additional information on KQML, including papers, language specification, etc. can be obtained at http://www.cs.umbc.edu/qkml).

The FIPA-ACL (foundation for intelligent physical agents – agent communication language) consists of a set of message types and the description of their pragmatics [FIPA, web]. In contrast to the traditional RPC (remote procedure calls) based paradigm, the FIPA ACL provides an attempt at a universal message-oriented communication language. The FIPA ACL describes a standard way to package messages, in such a way that it is clear to other compliant agents what the purpose of the communication is. The FIPA-ACL specification describes every communicative act with both a narrative form and a formal semantics based on modal logic. It also provides the normative description of a set of high-level interaction protocols, including requesting an action, contract-net, and several kinds of auctions. The FIPA ACL (like KQML) does not make any commitment to a particular content language. This claim holds true for most primitives. However, to understand and process some FIPA ACL primitives, receiving agents must have some understanding of the Semantic Language (SL). Additional information on FIPA agent communication language specifications can be obtained at [FIPA, web] (http://www.fipa.org/specifications/index).

Other commonly used ACLs are the ***Agent Interaction Protocol (AIP)*** and ***STEP*** [Bjork & Wix, 1991] for providing semantics of messages in manufacturing applications.

6.4.4 Content (representation) languages - ontologies

Since communication between agents can only take place if there is a common understanding between agents about the concepts communicated in messages' content, standard content languages, also called representation languages or ontologies[1], have to be defined to create shared understanding between co-operative agents.

There are many defined ontologies but none is universally accepted to represent the content of an agent communication language. However, during many years of research efforts in developing agent communication languages (ACL), it is possible to distinguish the most commonly used for expressing information in messages. They are: ***KIF (Knowledge Interchange Formalism)*** [Genesereth & Fikes, 1992], ***FIPA SL (Semantic Language)***, and ***RDF (Resource Description Framework)***. Other content languages such as: extended SQL, LOOM, a constraint choice language (CSL), etc. used in earlier implementations of ACL, and recently XML (eXtensible Markup Language)[2], also have their supporters. Currently, there is a strong position that it is too early to standardise on any representation language. As a result it is necessary to say that two agents can communicate with each other if they have a common representation language or use languages that are inter-translatable.

Information on KIF and associated tools is available from http://www.cs.umbc.edu-/kse/kif. A detailed description of FIPA SL, including its own semantics, can be found in the FIPA ACL specification (http://www.fipa.org/specifications/index).

---

[1] The purpose of ***ontologies*** is to define the vocabulary (i.e. the terms) that will be used in the communication (knowledge exchange) between agents [Gruber, 1993].

[2] Another important emerging standard is XML (extensible markup language) from W3C group. XML is an application-independent language for describing data. A number of researchers have suggested that ACL messages ought to be encoded in XML in their entirety, i.e., both the message layer and the content layer should be in XML. In contrast to HTML which describes document structure and visual presentation, XML describes data in a human-readable format with no indication of how the data is to be displayed. Since XML will have a serious impact on the next generation of Web technologies it is of crucial importance to relate this technology with FIPA's developments. FIPA is considering the use of XML and RDF as possible encoding schemes of FIPA-ACL messages.

6.5 Some concluding comments on Multi-Agent Systems

Multi-Agent Systems (MASs) are still at a stage of development particularly in the domain of manufacturing systems. Continued advances are especially noticeable in the field of agent communication standards and improving agent development platforms. The most successful deployment of agent technology is in information industry. In this relatively new field (in comparison with manufacturing) difficulties accompanied with the introduction of new technologies seem to be less prevalent. Firstly, because information systems represent a homogenous environment (applications are entirely computational in nature like software agents are), and secondly, prohibitive traditional thinking and inertia are less manifested. To the contrary, however, the deployment of agent technology in traditional manufacturing systems has made much slower progress. Regardless of some early industrial implementations presented in research publications, in which agents have been moved out of the laboratory into the industrial workplace, it appears that a heterogeneous and more complex manufacturing environment, in which digital manipulations must be coupled with material transformation through elaborate control schemes, imposes many difficult problems for the deployment of agent technology. For example, most of the agent's scheduling architectures and protocols have only been developed to the theoretical stage. They are yet to be implemented and their viability and efficiency is yet to be tested. In addition, solutions to some of the difficulties that are fundamental to agent scheduling have not been found. For instance, although most researchers have put a lot of effort into achieving good communication among the agents, it is still a confusing and difficult problem. The first question to be answered is, "What information should each agent transfer in order to obtain high efficiency? Should we have a common information centre or should agents just provide information on their own status or should they make inquires about others in a point-to-point fashion? Which is better, and in what situations? We may be able to find some answers in the general DAI (Distributed Artificial Intelligence) problems, but for manufacturing scheduling problems more specific answers are required. The second major question is how to prevent or solve the conflicts between the individual local solutions. As we learnt from studies and implementations of multi-agent scheduling systems, conflicts can be prevented by simply queuing up tasks (jobs) on critical resources and not redoing the overall scheduling at all. Routing flexibility has to be utilised as much as possible, and yet queuing cannot always be avoided. So far, in the cases where we do want to consider conflicts, there is no agreement on the proper way to do so. By "texture measures" [Sycara et al., 1991a] agents can foresee some of the possible conflicts, but again conflicts cannot be totally eliminated. "Negotiation" is, therefore, suggested by almost all authors. But these negotiations are far from mature in the sense that often they are just "doing negotiation for negotiation's sake", that is, they only concentrate on solving conflicts, not on improving solutions.

In summary at the time when this project commenced, most of the above mentioned systems were expensive. However the main reason why the development of the test bed application had not been conducted using the above tools was the lack of support in the case should something go wrong. In addition, it was considered that these tools were not yet sufficiently developed for practical application. Also many of the systems have proprietary features which do not make them suitable for universal application. On the other hand, it was expected that the Visual Basic community of over 4 million users and Microsoft Corporation could provide prompt and adequate support when needed. Also one of the project's objectives was to use a general-purpose programming language to avoid any proprietary issues that might be encountered during the development.

Practical investigations as well as comparison with traditional techniques should be the focus of future work in this area. Only on the basis of such work it would be possible to find appropriate agent architectures and communication schemes for different real-world situations. It is for these reasons that the research in this project has concentrated on the development and implementation of an operating core of a heterarchical, Multi-Agent System, which could be used for comparing performance of such a multi-agent system with the other more conventional types of planning, control, and scheduling systems described earlier in Chapter 3.

A review of manufacturing systems with particular problems in relation to manufacturing control systems indicated that modern small manufacturing organisations responding to rapid market changes and variable demand for their products posed particularly difficult scheduling problems. Many New Zealand manufacturing companies fall into this category making the solution of the scheduling and control systems particularly relevant.

In reviewing the control systems for manufacturing organisations such as those identified in this and the previous chapter, it appeared that heterarchical systems using agents should be able to provide a cost effective solution. At the same time it was clear that the newer technologies of the so called personal computers and networking had changed the cost effectiveness of this type of hardware and software, relative to the high cost of centralised computing systems, suggesting that distributed networks of the smaller computers could be utilised to provide the hardware and software basis for heterarchical, agent based distributed manufacturing control systems. From these deductions the research problem of interest for this study was developed to be as described in the following chapter.

Chapter 7. The problem statement, the goals, and the research objectives

After introducing the background material, from Chapter 2 to Chapter 6, this chapter defines the problem statement, the goal and the objectives of the research project.

The chapter is organised around the following topics. Section 7.1 provides some introductory information related to the project itself. A retrospective discussion on relevant background material is given in Section 7.2. Section 7.3 discusses a few points that influenced design of the proposed system of heterarchical shop floor control. The problem statement is defined in Section 7.4. Section 7.5 defines the goals of the research in its broader and narrower sense. Section 7.6 presents a list of the research objectives. The chapter finishes with a few concluding comments given in Section 7.7.

7.1 Introduction

The New Zealand manufacturing environment consists of mainly small manufacturers who are finding it difficult to afford the flexible manufacturing systems which would enable them to compete in international markets by providing niche market products with a rapid response to changing market requirements. They are also unable to afford the complex and costly enterprise planning systems designed for larger scaled manufacturing operations. In addition they are operating in one of the freest international market places with very low or non-existent tariffs for imported products. This leads to particular requirements for their systems and this thesis is aimed at researching and developing some form of manufacturing control systems which would assist them in remaining internationally competitive. It is also seen that such systems have applications in many parts of the international manufacturing community where not all (or in fact perhaps relatively few) manufacturers these days can rely on economies of scale to be competitive.

This project was intended to be the first in a series of projects to investigate manufacturing control systems suitable for New Zealand manufacturers with emphasis on rapid response to small production run operations using automated manufacturing systems for niche market areas. Unfortunately because of changes in the University structure, staff departures from the Faculty of Technology and Engineering, and management reorganisations at Massey University this project became an orphan as the general programme area could no longer be supported either with staff or funding. However, since the project was well underway at the time of the changes it was appropriate to finish the project to the point where a model of control system could be developed and demonstrated. Also, for the above reasons, the follow on projects are, unfortunately, unlikely to be developed.

7.2 A retrospective discussion on relevant background material

In **Chapter 1** we noted that the effect of ***globalisation*** has been an increase of competition worldwide for manufactured goods. This makes it more important for manufacturers to be efficient and effective in meeting ***customer requirements***. Customers are demanding that products have a wide range of options available to meet the needs of the particular customer. Products need to be reliable and of high quality and need to incorporate the most recent technological developments. (Many of the features which sell products in today's international markets are fashionable rather than utilitarian as can be seen in items such as VCR's and DVD players but it is the particular features which sell a given product even if they are not used by the consumer in practice. These fashions change rapidly as competitors attempt to increase their market share). Customer demands put manufacturers under increased pressure to be first to market to capture the early market share and then maintain it by introducing product refinements ahead of the competition. Customer requirements (refer to Section 1.1 "Preliminary remarks") and market trends (refer to Section 5.1 "Introduction") lead to a wide range of difficulties particularly in the area of planning and control of manufacturing systems. From the literature review it was evident that traditional centralised and hierarchical approaches to manufacturing control were not able to provide adequate solution for the above developments and that there was a need for advances in manufacturing control systems and accompanying software. Development and implementation of distributed heterarchical control structures is seen as a feasible approach (as discussed in Chapter 5) to cope with the new challenges, particularly with the increased complexity, dynamic and uncertainty in manufacturing environment. Distributed systems offer a whole set of favourable characteristics such as: adaptability, fault-tolerance, etc. (see Section 5.3." Main requirements of the new generation of manufacturing control systems") and became of particular importance for low-volume, high variety job shop manufacturing systems (which are systems of our interest; discussed in more details in Chapter 2), in which order rates and sizes are not known in advance. On the basis of the literature review it was determined that there was a lack of research tools for investigating heterarchical distributed structures (a few such tools were part of ongoing research programs and were not publicly available). The lack of the research tools was one of the main reasons for proposing this research project to develop a software system model of a heterarchical, agent based shop floor control system. (Reasons and motives for conducting the research project are discussed in Section 1.2. "Motivation for conducting this research project").

In **Chapter 2** the importance of batch production in manufacturing industry (in particular low volume, random job shop manufacturing systems) is discussed and a need for a manufacturing control architecture which would be able to provide an efficient control of a manufacturing facility that might be comprised of a large number of machine tools is emphasised. The project advocated in this thesis is oriented towards developing a practical, low-cost manufacturing control architecture (discussed in Chapter 8) for use in a rapidly changing small production run environment. However, one of the main characteristics of the control system is its scalability. The system can be easily extended and is able to control (it is believed with equal efficiency) a manufacturing system consisting of a large number of machine tools, so that the existence of large flexible manufacturing facilities could become a reality.

In **Chapter 3** we concluded that in practice the traditional approaches in planning, scheduling and control of manufacturing systems (based on MRP/MRPII concepts), are too inflexible, costly, and too slow to satisfy the needs of contemporary manufacturing systems. Because of the huge amount of information needed, which is changing all the time, traditional control systems have difficulties in coping with the increasingly dynamic and unpredictable environment that

became evident in everyday operation on the shop floor (for example, the rapid changes in customer demands, product developments and changing factory environments such as machine breakdowns, delays in material delivery, etc. covered in the previous chapters). There are two main reasons why these systems became less efficient. Firstly, traditional methods are *essentially centralised* in the sense that all the information concerning every job and every resource still has to go through one central computing and logic unit (either in centralised or hierarchical control structures) to do the calculations. Secondly, the traditional approaches are based on a *metaphor of prediction.* However, the actual shop floor always deviates from whatever model one uses to predict behaviour. To cope with unpredictable deviation of system's operating parameters from those assumed in computing a schedule, 1) "deferred commitment scheduling" should be applied and 2) the majority of short term planning and control decisions need to be moved from the higher levels of manufacturing control to the shop floor. That is, the manufacturing control systems should be designed in such a way as to rely on forecast information as little as possible. For the above reasons, more and more researchers have begun to try solving the complex control and scheduling problems in a *distributed heterarchical way*. The shop floor control system presented in this thesis is based on the reactive scheduling (uses a principle of last commitment) and scheduling decisions are made on the shop floor ("on fly" and in real time) among production workstations during work-parts fabrication.

In **Chapter 4,** after a brief review of a classical scheduling research, we deduced that in spite of some breakthroughs that have been recently achieved, the scheduling problem in job shop manufacturing systems still remained unresolved in satisfactory manner. Namely, all the traditional approaches encounter considerable difficulties when they are applied to real situations. This is due to the following two main reasons. Either the traditional scheduling methods use *simplified theoretical models*, (to cope with the complexity of the problem) which seldom match the real-world scheduling environment, or they use methods that consider a larger set of parameters while looking for "optimal" or close to optimal solutions which are based on unacceptable *time consuming and computational demanding processes* for practical applications. The proposed model of a distributed heterarchical shop floor control system is believed to be capable of using real-life manufacturing settings and resolves resource allocation and scheduling problems in real time with negligible computational requirements (in comparison with the methods that seek optimal solutions).

Further, in **Chapter 5,** it is pointed that manufacturing systems with traditional centralised and hierarchical architectures do not perform well in environments characterised with unpredictable and frequent changes. This ability requires systems with architectures that are adaptable, scalable, fault-tolerant etc. (refer to Section 5.3. "Main requirements of the new generation of manufacturing control systems"). Subsequently, the chapter provides a review on some of new approaches (such as bionic, fractal, holonic and virtual manufacturing) that emerged recently in attempt to provide adequate solutions to these new challenges. As stated in Section 5.6.5 "Some remarks related to the review on new paradigms" the intention of this thesis is not to test any of the theoretical concepts given above. (The thesis has been concerned with the practical implementation of an approach based on agents and low cost computing elements). Rather, the review was provided as a theoretical basis and an introduction to the thinking that led to the approach used in this thesis. From the review it was apparent that distributed control systems have significant potential for controlling complex manufacturing systems and that the agent technology can offer many characteristics that are required for the realisation of this type of control. This is why was decided to concentrate the research on multi agent systems which appeared to be the right approach for developing distributed control architectures. The chapter finishes with the list of several specific requirements (design principles) that an agent-based shop floor scheduling and control system needs to fulfil (see Section 5.8. "Design principles for highly

distributed heterarchical shop floor control systems"). The research project undertaken in this thesis aimed to address most of these requirements.

It should be noted that at the time when the major literature review was conducted in 1997, it was difficult (in some cases impossible) to have access to data of importance for the research. In the published reports on distributed manufacturing research, the most important elements of know-how (application details, such as decomposition and structure of distributed data and the particular details related to the software realisation of implemented negotiation mechanisms/protocols) were omitted. A good example to demonstrate this was the research on holonic manufacturing systems. Due to the involvement of partners from industry and the enforcement of legal acts related to protection of their intellectual property rights, key findings from this research were treated as classified information and, therefore, were strictly limited to members of the Holonic group. Researchers were restricted from freely publishing their results. This was very well illustrated in [Tharumarajah et al, 1996] where authors acknowledged that: *"Due to the restricted circulation of these studies, references to the studies undertaken have been omitted."* (For these reasons an in depth review on holonic manufacturing systems was not included). The aim to obtain the above mentioned know-how knowledge, to get insights of distributed manufacturing systems and to clear a path to further research in this domain, led to the decision to attempt to develop a simplified prototype application of a multi agent heterarchical shop floor control system.

Chapter 6 provides a brief review of agent technology that has been seen as a promising approach for developing distributed manufacturing system architectures (see Section 6.3.1. "Why use agents in manufacturing systems?"). Although there has been a considerable amount of research into the requirements of modern manufacturing control systems it would appear that relatively little of the developments have been progressed into commercial development. It still appears that such systems that are available are expensive and the details of their operation are commercially confidential and also because Massey University was not able to be a participant in the international projects, so that from a research perspective it was difficult at the time the project commenced to assess their usefulness in the environment under consideration.

Previous research has shown that acceptable overall performance of the agent-based systems in manufacturing can be achieved from local interactions even with simple agents. To avoid many of the design pitfalls (focusing attention on resolving issues that are not of crucial importance to get system to work) and to achieve the ultimate goal of the project (to develop a fully operational model of a multi agent shop floor control system), a minimalistic approach in design and development of multi agent systems was deliberately adopted (the system consists of only one type of agents –workstation agents). The focus was on simplicity (to avoid issues that could potentially arise in agent communication and to provide enough time for resolving technical programming "problems") but, at the same time, the intention was to implement as many "beneficial features" as possible that had been reported to be beneficial in the previous research work.

An incremental approach to the system development was applied. Namely, at the very first stage of the project the aim was to develop a functional prototype of a core part of the system, to demonstrate system feasibility, and later on, through incremental development and continuous improvement process, to make a system more sophisticated by mapping other entities of a manufacturing system as agents (jobs, individual machines, AGVs, tools, fixtures, etc.). Also, at the later stage, once the system became operational and refined (properly tested and cleaned of bugs), another more sophisticated negotiation scheme and other scheduling algorithms could be applied for experimenting and testing purposes. Examples of this sophistication include the introduction of time slots for machines as in [Dewan & Joshi, 2001] or the inclusion of a currency scheme in the negotiation process as suggested in [Ottaway & Burns, 2000].

7.3 Some points that influenced design of the system

Recent developments in a domain of information technology industry had a significant influence in the selection of a developing platform for conducting the project. The development of the modern so called personal computer has led to low cost but extremely powerful computers being available with capabilities far surpassing those available a decade or so ago at a very affordable cost for even a relatively small manufacturer. In addition there has been a development of de-facto "standardisation" through the dominance of systems such as the "IBM compatible" PC with Microsoft Windows® and Apple Macintosh® systems (refer to Section 1.2 "Motivation for conducting this research project"). Because of the ready availability of these systems the hardware and software for the system were to be chosen from those that have the most universal use in industrial and commercial applications with the widest range of software development support applications. The overall cost of the system was a major consideration in this project and the so-called "Personal Computers" connected in Local Area Network (LAN) was a logical choice for a hardware base to be used in this project (refer to Chapter 9 which discusses the selection of a development platform).

The Internet and its associated networking technology has revolutionised communications for all sectors and has made practicable cooperative ventures such as virtual factories, global clustering without geographical boundaries and so forth. At the same time the new technology is affordable as is the case with the modern personal computers discussed above. Both the computer and network developments have in effect sidelined a lot of the previous standardisation of manufacturing systems, particularly for smaller manufacturers. As with the hardware the choice of the communication network was to be based on the most universally available network system and this led naturally to the use of the Internet TCP/IP network as discussed in more detail in Chapter 9.

At the time when the project commenced agent technology was in a process of rapid evolution. Because of universe applicability of basic principles, ideas from multi agent systems (MAS) have been used in many domains of human activities. As a consequence, the definitions of an agent and what constitutes an agent were not firmly determined. Over time the agent technology has been recognised as successor to object oriented programming [Parunak, 1998c]. However, in the absence of sound agent development platforms (with standardised agent communication languages and software tools specifically tailored to ease the design and creation of distributed multi agent applications), agents were built as an "ordinary" software units based on the principles of the object-oriented concept. Looking from outside though, the agent was a "special" software unit that has autonomous, proactive, and cooperative characteristics (an agent can make its own decision, can run without external invocation, and it can establish relationships with other agents to achieve its own or system goals). In the absence of suitable platforms (they were not available at the time when the project commenced), a similar approach to the selection of the agent software development tools to that used for the hardware and software systems was adopted in this project. Namely, a decision to use one of the prevalent programming languages for rapid application development in building the multi agent application was made.

7.4 Problem statement

There are a number of issues that need to be addressed when developing a model of a distributed agent based manufacturing system. A few are listed below.

- How to decompose a system and how far in the system decomposition should we go? In other words: What manufacturing elements should be modelled as agents? Too many elements mapped as agents can result in unnecessarily complicated system design. Also,

it will impose more difficult requirements on the communication infrastructure and may result in degraded system efficiency. On the other hand, too simplistic an approach may lead to less flexible and less efficient systems.

- How to establish relationships among these entities in distributed heterarchical manner?
- How to distribute control, responsibilities and data among system entities?
- What should be communicated among agents?
- How to reach global optima with selfish agents pursuing their own goals?
- What is the optimal ratio of hierarchy and heterarchy in a given situation?
- Which control strategy is better? Using the same manufacturing reference model and production data it would be possible to implement different negotiation and control strategies and to compare them by performing a variety of performance analyses.
- How to deal with disparate viewpoints and conflicting intentions among agents?

To provide answers to some of these questions, the development of a sound model of an agent-based manufacturing system was a requirement of the Massey research programme when this project was commenced.

7.5 The research goals

The goal of the dissertation, in its broadest sense, is *to contribute towards the development of efficient, low-cost, heterarchical shop floor scheduling and control systems for application in small-scale, job shop manufacturing systems with low resources which produce products in low-volumes but with a large number of varieties, and in which order rates and sizes cannot be predicted in advance*. This type of manufacturing systems is typical for New Zealand industry, as when compared with the much larger manufacturing facilities encountered overseas.

More specifically, the goal of this dissertation was *to develop an appropriate agent-based software model (a test-bed network application), as a core part of a distributed heterarchical shop floor scheduling and control system. The model was to be used as a research tool for experimenting with different control strategies and conducting a number of experiments on the performance of such heterarchical agent-based shop floor control systems.*

One of important goals of this research is *to demonstrate that it is possible to develop a heterarchical shop floor control application by using commonly available and inexpensive technology (PCs, operating systems, network hardware and communication protocols, relational databases, and programming tools) that was ubiquitous at the time when the project commenced*[1]. Since such an application would include most of the crucial control mechanisms that would be found in the real systems, the application could be considered as a partially developed real heterarchical distributed shop floor control system using an agent based approach.

Finally, once tangible results are obtained and feasibility of approach is demonstrated, it was expected that the model to be developed (as a leading project) *would facilitate the creation of an environment in which the subsequent phases of the project would be initiated and established (improving program design, implementing adequate programming tools, using emergent agent standards, and so on)*. After investigating the performance of such a small demonstration

[1] The simulation model of the distributed heterarchical shop floor control system is implemented by using Visual Basic 5 and Microsoft Access Version 7. The software model runs on PC Windows 95/NT workstations that are connected into a LAN by using the TCP/IP protocol suite.

system, the test-bed application could be then scaled up to a full, real world practical industrial application of a distributed heterarchical shop floor scheduling and control systems.

These goals led to the identification of problems to be addressed in this research. They are divided into following domains:

- Identification and definition of production units in a heterarchical shop floor control (SFC) system;
- Design and formulation of both: a job shop manufacturing system model and a heterarchical shop floor control system model;
- Identification of the most important information for facilitating production processes and defining how this information will be distributed among autonomous entities in heterarchically controlled manufacturing system.
- Integration of process planning and scheduling functions;
- Selection of hardware components (determining the minimum of adequate resources in terms of processors, memory, hard disc capacities, network cards, etc.) and network protocols (Ethernet, Token Ring, TCP/IP, etc.) for establishing an experimental platform for developing a software model of a heterarchical SFC system;
- Selection of operating systems, software programming tools, relational database systems, and other technologies (ActiveX, ADO, etc.) for developing the software model.
- Development and implementation of a software control model.
- Demonstration of the feasibility of proposed solutions.

The project focus is on distributed control at the shop floor level. This sets the research between two abstract boundaries. The scope of the interest towards higher layers of the overall production planning and control (PPC) system extends to the point where production orders enter the shop floor system. On the opposite side, a lower boundary separates agent based shop floor control systems (workstation agents) from the immediate physical control of production resources on a shop floor.

7.6 The research objectives

The specific objectives of the research project are as follows:

- To provide decomposition of the job shop manufacturing system and to identify basic constitutive elements of a manufacturing system that is to be heterarchically controlled (mapped as agents);
- To define a structure of a job shop manufacturing system model that will be used as a reference model for the development of a distributed, heterarchical shop floor control (SFC) system;
- To define an operational model (architecture) of an agent-based heterarchical SFC system that facilitates modifiable, extensible, reconfigurable, adaptable, reliable, and fault tolerant control. This model is to be used as a blueprint for developing a test-bed application of a heterarchical SFC system. An important objective is to provide scalability from a basic small-scale demonstration system to a full scale, real world practical system as well as a system that can be used for simulation in a university environment where investment is severely limited.

- To derive a methodology for resolving real-time scheduling and resource allocation problems in a heterarchically controlled manufacturing environment;
- To define the information that is necessary for workstations to make rapid and effective scheduling decisions while routing work-parts through the manufacturing system;
- To envisage a mechanism that will provide integration of process planning and scheduling functions;
- To analyse manufacturing data and decide how it will be distributed among workstation agents. Some data are to be kept locally on the workstations (inside workstation agent's database) while the other is transferred among workstations. In the latter case the information flow is supposed to be identical with the physical flow of work-parts (both should have the "same routes" through the system);
- To define a set of messages that would enable the conducting of efficient negotiation processes between workstation agents (using the original Contract Net protocol as a model). The application should demonstrate the execution of an auction-bidding scheme;
- To envisage the decision-making processes (flow of information) inside workstation agents;
- To select a developing platform (computer hardware and software technology) that will facilitate the development and implementation of a heterarchically controlled system with a high degree of fidelity;
- To master and apply one of the programming languages for rapid application development (Visual Basic) and network technologies (covering predominantly basic hardware aspects and communication protocols used in Local Area Networks - LANs);
- To create an experimental LAN for developing and testing a software model of a heterarchical shop floor control system;
- To design and develop a small communication application for creating, sending, receiving, and interpreting appropriate messages for the system over a LAN;
- To design and develop procedures for real time simulation of manufacturing activities inside workstations;
- To design and develop a relational database for storing and retrieving information that is kept locally in the workstation agents (design the underlying tables, relationships; set the primary and foreign keys and so on);
- To design and develop a graphical user interface (forms) to manipulate (entering, editing, and deleting) data in the local workstation agent's database;
- To support human supervisory control of both simulated manufacturing and control activities performed inside each workstation. A user should be able to take control of the simulated manufacturing activities (machine processing and robot material handling operations) and the execution of the negotiation processes between workstation agents (sending and receiving messages) at any time;
- To design, develop, and implement an overall functional software package - a core part - of the agent-based SFC system that can be used for modelling and simulation of a job shop manufacturing system controlled in the heterarchical manner;
- To validate the methodology and the developed control system by demonstrating a working test-bed network application on an experimental LAN.

7.7 Concluding comments

The existence of a reliable means for exchanging information between agents is a fundamental precursor in all the research work described in this thesis. Also, the kinds of communication and information exchange (considering interfacing issues) between the agent and physical workstation's control devices that are part of the workstation and in charge for controlling physical equipment are of fundamental importance for conducting workstation's control functions. However, the project did not intend to pursue research on inter-agent communication (considering ontology issues for example) nor communication among physical control devices. These matters were not the subjects of the research at this first stage of the project development and thus are not further discussed (this task is left to appropriate authorities that have been established with such goals). Rather the intention was to implement results from such research in the subsequent developments of the project. The emergence and development of FIPA-ACL and IEC61499 standards are the best examples of such work.

In summary the project was to develop a demonstration low cost heterarchical distributed shop floor control system suitable for use in a rapidly changing small production run environment using universally available hardware and software. The system should be capable of modelling and simulating a job shop factory operation so that future projects can investigate its performance in relation to more traditional control systems. Therefore, the thesis is a demonstration of a practical concept for real world manufacturing operations not a theoretical and academic study on, for example, scheduling or communication issues.

Having selected the overall goals and detailed objectives, the project proceeded in the following stages:

- The description of the target manufacturing system as described in Chapter 8.
- The selection of operating system and the software development tools to be used are discussed in Chapter 9.
- The assembly of the hardware (creation of the experimental local area network) and the design of the distributed heterarchical shop floor control system, including the agent structure and agent-to-agent communication, are described in Chapter 10.
- Finally, the demonstration of the prototype system as constructed is covered in Chapter 11.

Before progressing with the description of the developed system, it should be noted that besides envisaging and designing of the overall heterarchical shop floor control system (in the way which is discussed in Chapter 8 and Chapter 10), the following tasks have been done solely by the author:

- Definition of a manufacturing system model (decomposition of the systems and adoption of a workstation model as a basic building block around which the workstation agents were designed and developed).
- Design of a workstation agent and its constitutive modules. On a basis of the literature review it became quickly clear, at the very beginning of the project, that an agent has to have: 1) a communication module (for exchanging messages), 2) a local database module (for storing workstation's local data), 3) a processing or decision making module (for coordination of agent's activities) and 4) a control module which has to interact with workstation (modelled or real) components such as, for example, a machine and a robot.
- The entire software development. No other software or tools have been used than the software development system and the author undertook all the software development.

- Assembling, configuring and testing IBM compatible personal computers (from partitioning hard disks to installing operating systems and resolving conflicting - IRQs and memory allocation - problems), which are used in the experimental local area network.
- Setting up a small peer-to-peer local area network (from installing network cards to wiring and configuring network protocols).

As indicated in this section, the next chapter describes the manufacturing system reference model adopted.

Chapter 8. Description of the manufacturing system model

This chapter describes the basic structure and operational principles of the adopted reference model of a job shop manufacturing system, which was used as a basis for the development of a proposed distributed heterarchical shop floor control system.

8.1 A modelling framework

To develop a test-bed application of a heterarchical shop floor control (SFC) system more detailed descriptions and the level of decomposition of the components of the manufacturing system are required. For example, decomposition may be done to the work cell level, or further to the workstation or to the equipment level.

An important task to be addressed at this stage is to consider which elements in the system should be mapped as an agent. On the basis of the literature review, we found that agents are typically mapped either to clusters of problems, functions, nodes in a manufacturing hierarchy, or shop-floor entities (refer to Section 6.3.2. "Types of agents and their application"). In this project the latter approach is adopted (to map shop floor entities as agents) but still the question about what entities should be mapped as agents remains open. In principle there are two approaches. One is to have a number of diverse agents in the system (agents that map basic manufacturing elements such as machines, AGVs, work-parts, tools, fixtures, etc.) or to have a system with homogenous, almost identical agents that map more abstract production units such as workstations or work cells. As discussed in Section 7.3. "Some points that influenced design of the system", we adopted the latter, simplistic approach. To design such an SFC system (based on homogenous workstation agents) a description of the manufacturing structure and the definition of the accompanying terms are necessary precursors.

As denoted in Section 2.1.3 "A desirable structure for a model of manufacturing system", the selection of an appropriate structure for the manufacturing system is an important task that occurs at the phase of the design and development of manufacturing systems. Each structure will determine a pattern of workflow through the system. If the same operation for a given part type is assigned to more than one machine or worker, then it will be necessary to decide which machine or worker should be allocated with work-parts. Also, if different part types require processing by the same resource, then it will be necessary to choose which part type to process next. In both cases it means that there will be a need to control job flow through the system. Therefore, the model of a manufacturing system was developed considering: (as it would be the case with the real system)

- Firstly, the structure of the system that was adopted keeping on mind what orders (part types) will be produced, and
- Secondly, the control of the flow of jobs through a system of the given structure.

8.1.1 Components of the manufacturing system model

A basic constitutive element of a manufacturing facility is an ***equipment entity***. The equipment entity represents a piece of physical equipment, which nowadays is usually an automated or semi-automated facility with a built-in equipment controller. Equipment entities in a manufacturing system can be divided in two main groups:

- ***Processing entities***, which refer to machines which can change the physical (shape, dimensions) or other attributes (such as chemical characteristic) of work-parts, and
- ***Supportive entities***, which refer to other equipment, which is used for transport, material handling, inspection (quality control), and storage of work-parts.

Processing and supportive entities can be organised in several ways to form ***production units***.

Depending on the complexity of a production system, a production unit can be a single machine, or a set of machines with associated personnel. For example, Bertrand, et al. considered a production unit as a selected part of the production process of a company that produces a specific set of products from a given set of materials or components, with the use of a particular set of capacity resources [Bertrand et al., 1990]. Depending on how their equipment entities are organised, the following production units can be distinguished:

- ***Workstations*** represent small groups of equipment with direct interaction with one another. Workstations have a basic configuration that is made up of a machine (conventional or CNC type), an operator, incoming or input buffer, outgoing or output buffer, and a local magazine for storage of the accompanying cutting tools, jigs, fixtures, and other accessories (Figure 8-1). In addition, fully automated systems (Figure 8-2) may have a material handling device (robot or manipulator) instead of a permanently employed operator, input/output transport ports, and an inspection station. Input (delivery) transport port and output (pick-up) transport port, together with the indoor material transport system (for instance AGVs) provide the interface to the other workstations and entities of the system. In addition, these units can have a role as temporary buffer stations. In the case of fully automated systems the workstation has a ***workstation controller*** that is responsible for co-ordinating the interaction of the equipment within the workstation.
- ***Work cells*** could be considered as extended configurations of workstations. They can have several machines with input/output buffers and several material handling devices, which facilitate all the material handling operations inside the work cell. A work cell is an organised entity that can perform several manufacturing tasks (Figure 8-3).

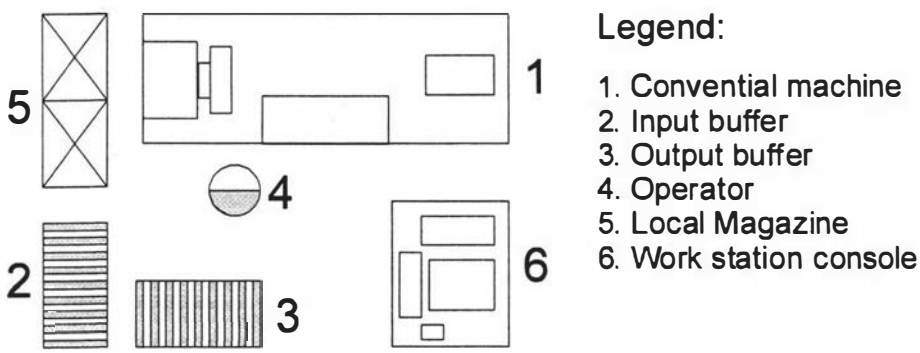


Figure 8-1. A layout of manually operated workstation

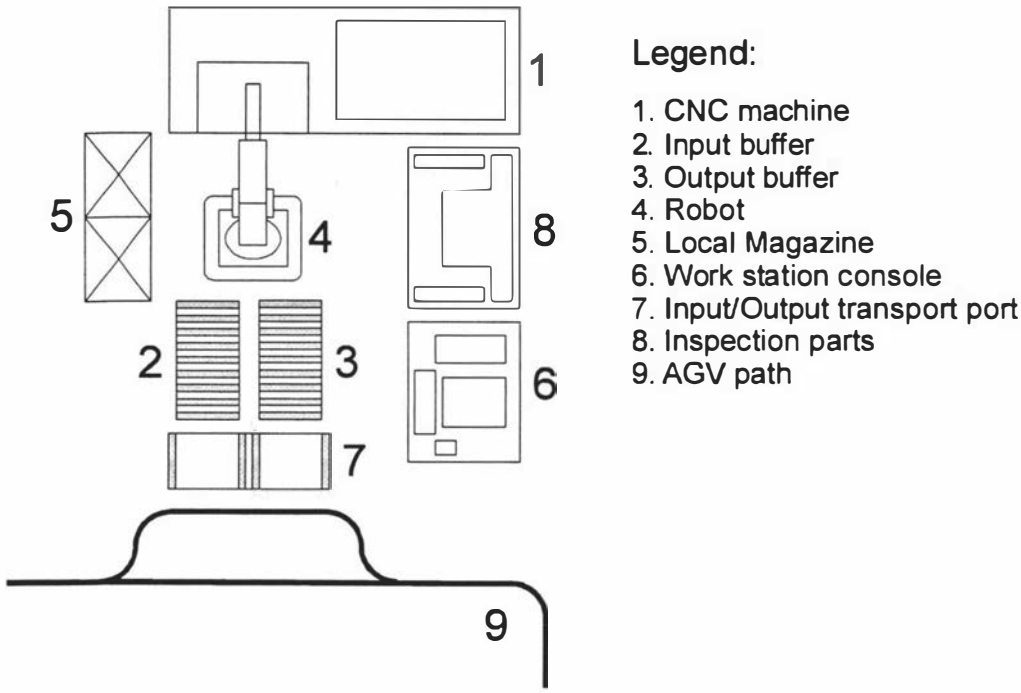


Figure 8-2. A layout of fully automated workstation

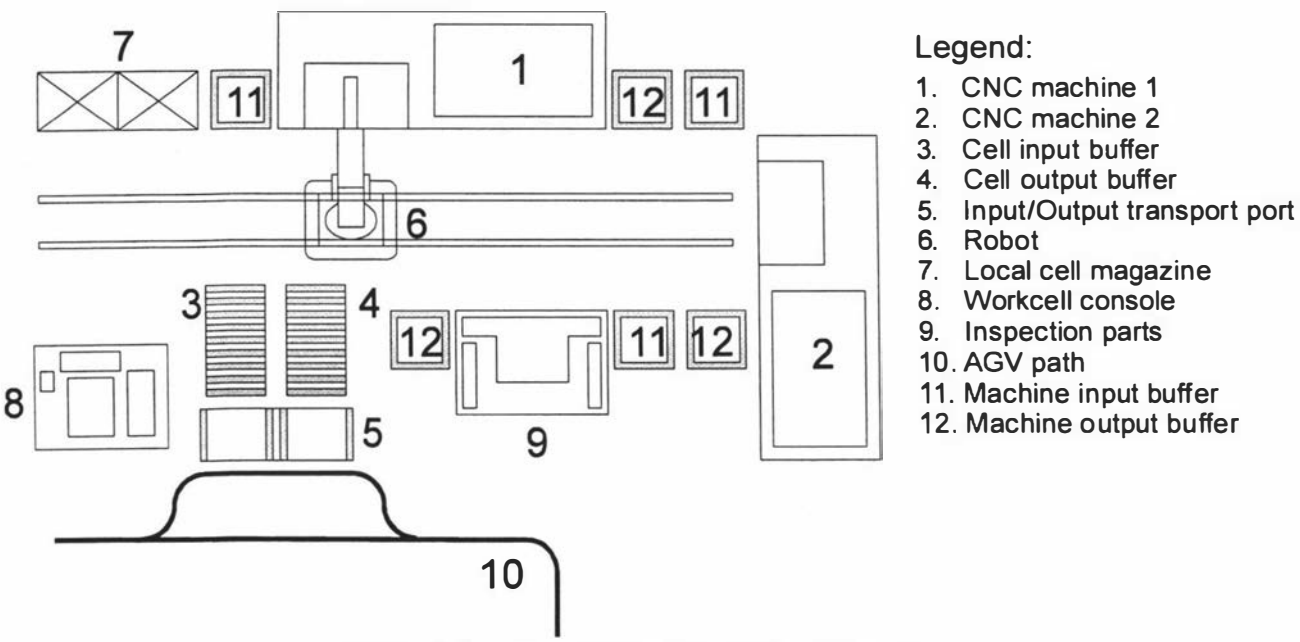


Figure 8-3. An example of a work cell layout

- Workstations and work cells can be arranged in larger production units. **A *production cell*** may consist of several work cells and/or workstations. Production cells commonly use a product type layout, (layout based on the principles of group technology as discussed in Section 2.1.1 "A brief review of development of manufacturing systems with emphasis on job shop production") though they can be organised as a process type layout. In a production type layout the production cells are organised around a "family of parts" that are usually produced completely inside one cell.
- ***A Shop*** can consist of one or more production cells.

As it will be discussed later in Section 10.1 "The proposed control architecture of a heterarchical shop floor control system" ***the workstation is adopted as the basic autonomous production unit of the manufacturing system around which the heterarchical shop floor control system is designed and developed.*** This choice has been adopted because a workstation represents an elementary functional and organised production unit that is capable of performing versatile manufacturing processing tasks autonomously for a wide range of different work-parts. With regard to the decomposition of a manufacturing system, this choice represents a compromise between two contradictory objectives. Firstly, there is a tendency to decompose the system as much as possible (and to map elements as agents) so that the maximum flexibility and reliability of the heterarchical shop floor control system can be achieved. On the other hand, going too far with the decomposition of the manufacturing system leads to a more complex design of the heterarchical shop floor control system at a later stage (this concern is related in particular to the design of agent communication modules that have to handle a number of simultaneous negotiation processes and how this approach would effect overall load on communication network). The approach adopted in this study imposes low requirements on a communication network (since communication is conducted among workstation agents only), and makes possible the design of heterarchical shop floor control systems that are equally successful in controlling manufacturing facilities of all sizes, that is, manufacturing facilities that are theoretically comprised of only one or of an infinite number of workstations. However, in practice, the number of interrelated agents and the number of open (by each agent) negotiation processes[1] affect the performance of the computer network.

8.1.2 Other preliminaries

This section gives descriptions of several frequently used terms, defining their meaning in the light of how they have been used in this project.

An ***order*** is a request to perform a defined function and it is intended to produce a certain output within specified constraints while meeting established objectives. A ***product order*** is a request at the factory level to make a specific kind of product. A product order is decomposed into ***job orders*** that are assigned to individual job shops.

A ***part*** or a ***work-part (work piece)*** is an individual item that is to be produced by the manufacturing system. Each part is fabricated by visiting several workstations in the system. In the production system, each part has several ***administrative attributes*** such as part type, part number, due date, customer number, etc., and a set of ***technical attributes*** which describe the part and define the required manufacturing operations. These technical attributes are used to create a process plan for the part.

Pallets are used to transport work-parts through the systems. They are interfaces between workstations and a transport system.

Jigs are used for positioning and orientating work-parts on pallets while ***fixtures*** are used for fastening work-parts on pallets. Therefore, jigs and fixtures hold the work-parts on the pallets and are bolted on pallets and travel through the system together with them. In addition, some machines might request different settings and, consequently, additional jigs and fixtures for setting up work-parts in different positions and orientations during processing on different work-part's surfaces may be required.

---

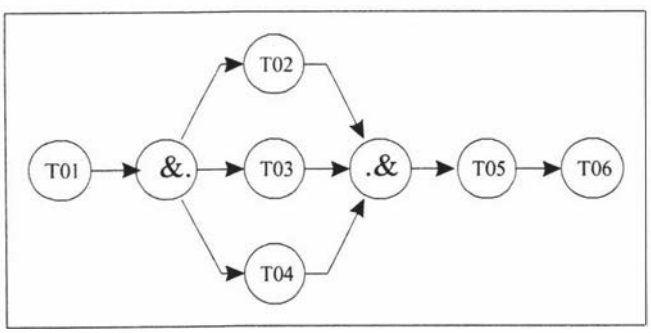
[1] In the proposed system, workstation agents currently negotiate only about work-parts but the system can be extended to include negotiation processes for example for AGVs, tools, jigs, and fixtures.

A route or ***a processing route*** is a part's flow through the system, and at the end of the route the part is removed from the system.

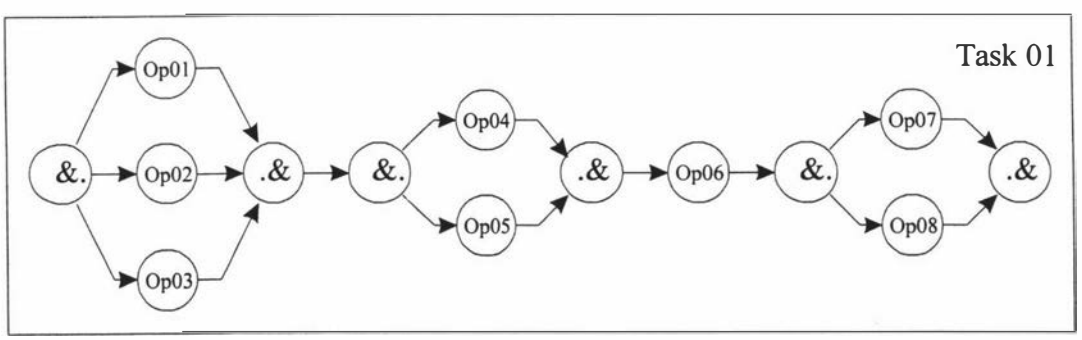
All the work to be done on a work-part through a single route is called a ***job***. Thus, one job is needed to process the work-part of a given part type. Because the corresponding work is never separated from a part, ***the terms part and job are used almost interchangeably*** so that the term job usually refers to a work-part to be produced.

The work to be accomplished on a work-part when it visits a workstation is called a ***task***. The work performed by a machine on a work-part using one tool is called an ***operation***. Therefore, to complete a job for a specific part, several tasks are required to be performed on different workstations. Moreover, a task may require one or more operations to complete.

A ***process plan*** provides the processing instructions necessary to produce a work-part. In discrete manufacturing, the production process of any work-part consists of a set of operations that are subsequently executed on a variety of production equipment. The entire work-part production process is decomposed into smaller tasks and further into operations. A sequence of those technological processes that a work-part needs to follow on its way through transformation from raw material or semi-product to final product represents a ***process plan*** for a given part. The process plan can be represented as a directed AND/OR precedence digraph [Cho et al., 1994, Cho & Wysk, 1995] that shows the precedence constraints and alternative routings for the work-part as paths through the graph (Figure 8-4). The graph can also be hierarchically constructed to show various levels of detail, for example, shop, workstation or machine level.



a)



b)

Figure 8-4. Process plan representation:

a) Shop level process plan graph, b) Workstation level process plan graph

In the ***shop level process plan*** each node in the graph represents a specific task (operation or set of operations) to be performed by a machine (workstation) or group of machines (a work cell that is considered as one undivided production unit). The processing instructions, which are

associated with each task, are assumed to exist (for example CNC files). Each arc in the digraph represents the movement of the part from the workstation represented by the tail node to the workstation represented by the head node of the arc. Any path through the graph represents a feasible processing route for the part. The shop level process plan that is used in this study to produce a ***Processing Route Table (PRT)*** file is discussed in Section 11.2.1 "Pre-process activities – the system input workstation".

Routing flexibility provides more options for fabrication of work-parts during the process of creating process plans. Routing flexibility is the capability to produce a work-part by using alternative processing routes. These alternatives are taken into account in the process plan and arise due to an availability of multiple machine types for processing a specific operation, as well as the ability to offer more than one operation sequence and still adhere to the prescribed precedence relationships.

8.1.3 Integration of the process planning and shop floor scheduling and control functions

This section briefly describes an approach that closely links process planning and the scheduling functions developed in this research study.

In recent years, the process plan has been identified as a critical element in scheduling. Information provided by a process planning function, is the input data for a production planning and control (PPC) system, which is, in turn at a later stage, responsible for the timing and execution of the operations at the shop floor.

Inspired by research work in the area of integration between process planning and PPC functions, as well as by many studies that have shown enhanced system performance when routing flexibility was considered, (for example [Arzi & Roll, 1993, Stecke & Solberg, 1981, Tsubone & Horikawa, 1999, Yao & Pei, 1990,][1]), the approach that closely links process planning and scheduling functions has been incorporated into this research study – a test-bed application. Two features distinguish the approach. Firstly, it increases process flexibility. It makes it possible for different process plans to be undertaken depending on the ongoing circumstances in the system at the time when the work-parts are manufactured. Secondly, the scheduling flexibility is increased by having a greater number of possibilities to choose from for executing a processing task (a few workstations per processing operation).

By employing the concept of Processing Route Tables, the process planning and shop floor control functions are partially integrated. This approach is based on the Non-linear Process Planning (NLPP) concept. Although, the process plans are not generated in the real time (that is, on-line while the processes are being executed, they are not fixed inputs for the scheduling function either (as is the case in the conventional, linear approach to process planning). In particular, in the conventional approach to process planning, depending of characteristics of both 1) machines installed in the system and 2) the work-part, which needs to be manufactured, it is possible to generate a few process plans for the work-part in advance, "off-line". However, by selecting only one of them in advance, prior to manufacturing, we have a sequential process plan, which cannot offer any flexibility to react to shop floor disturbances (refer to Figure 8-5). Even alternatives that are feasible and frequently used on the shop floor would be represented as unforeseen deviations from conventional plans. For that reason, an approach based on

[1] Authors agree that manufacturing performance can be increased by increasing routing flexibility. Tsubone and Horikawa stated that routing flexibility is superior to machine flexibility when machine breakdown is not negligible, especially when the duration of machine breakdown is long [Tsubone and Horikawa, 1999].

generating Non-Linear Process Plans (NLPP), used as an input data to a heterarchical shop floor control system has been adopted in this research. By using directed AND/OR digraphs, described in Section 8.1.2 "Other preliminaries", previously mentioned linear process plans can be presented as one large Non-Linear Process Plan (NLPP) with a number of alternatives (refer to Figure 8-5). Therefore, a process plan presented in this way has a "net" form, instead of the form of an inflexible linear sequence of processing operations, provides additional scheduling possibilities and can take into consideration actual shop floor conditions during the process of manufacture.

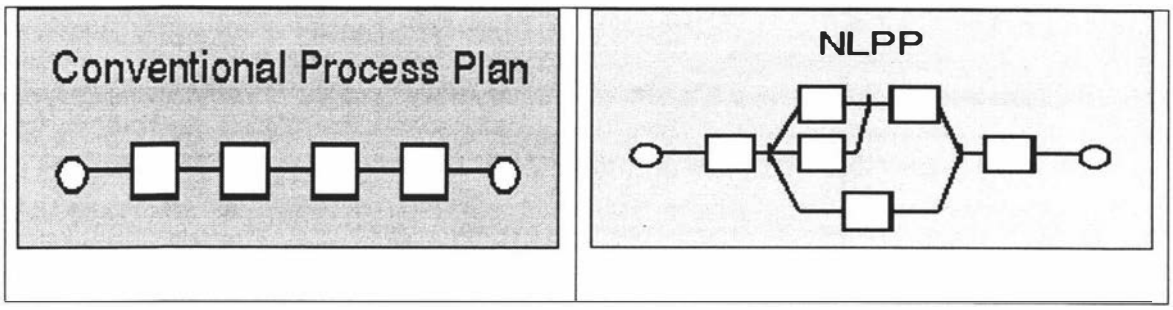


Figure 8-5. Conventional and non-linear process plans

In this research study, such an NLPP is further transformed into the form of a Processing Route Table (PRT), which is then used for dynamically scheduling work-parts on the shop floor as described in Section 11.2.1.2 "Getting the Processing Route Table (PRT)". Because each step (processing task) in the PRT file may have several alternative workstations it means that a final decision about which processing routes will be followed, that is, what segments of processing plans will be selected and executed, will depend on the current shop floor situation and will be made "on-line", during the manufacturing of the work-parts. This allows the selection of alternative workstations for example when one is unavailable due to breakdown or overload at the particular time that a given process on a work-part in the system is required. This assumes that more than one workstation can carry out the specific task required but does not require "preplanning" or recalculation of the total job shop schedule.

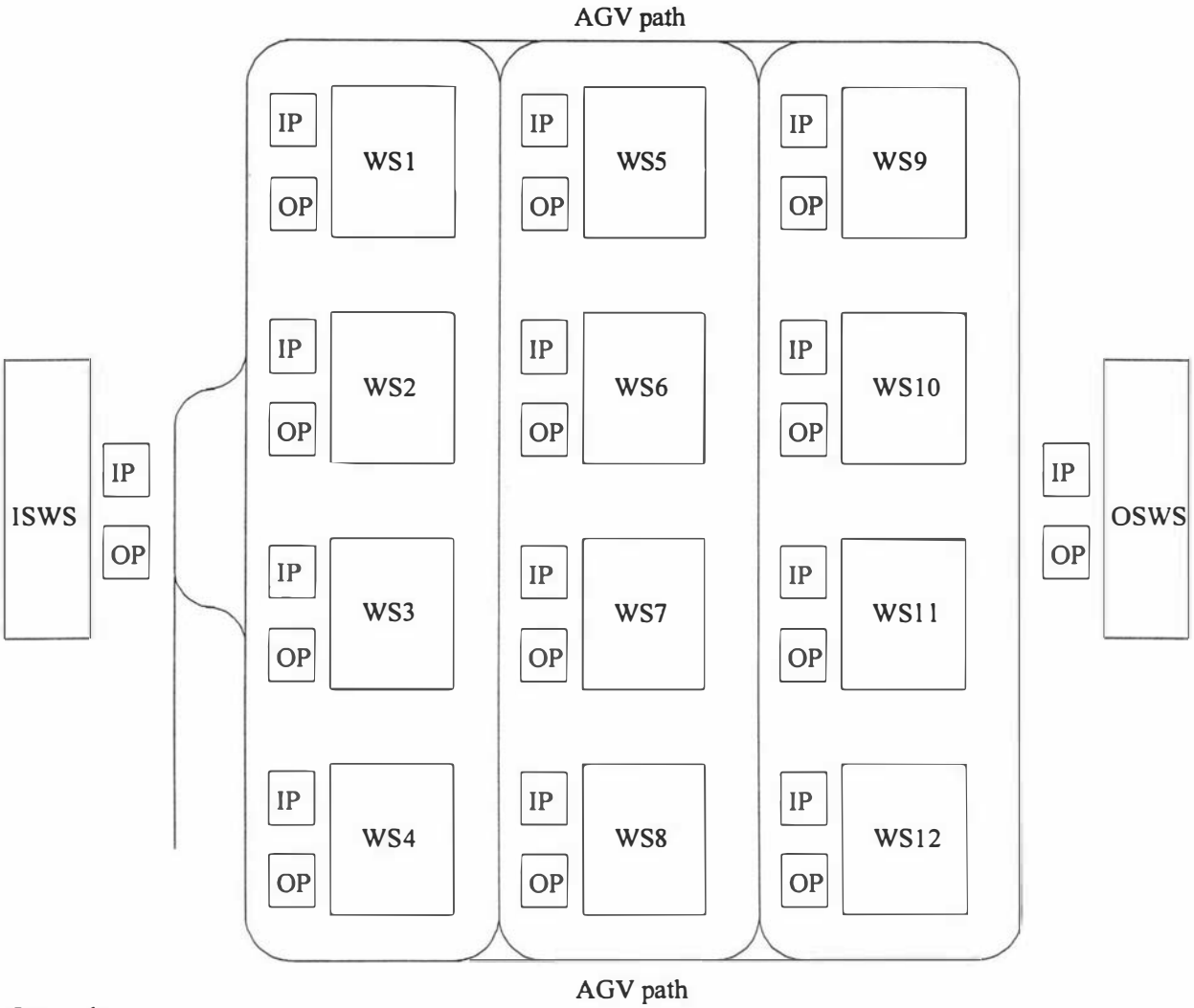
8.1.4 The proposed manufacturing system structure – a general description of the manufacturing system resource model

In this study, a workstation has been considered as a basic constructive element used for building a manufacturing system model. Because the interest in this study is in low-volume batch manufacturing systems, a general purpose, make-to-order, job shop manufacturing facility resource model with functional lay out is adopted (see Figure 8-6).

The main characteristic of the manufacturing facility, discussed in the following sections, is its ability to produce a number of versatile work-parts in small quantities. The objective of the system is to accommodate production of any work-part with minimal disruption. More precisely, for any work-part that belongs to the part type family for the fabrication of which the manufacturing system was designed. In a case when the transport batches are composed of only one work-part (such as is the case in the production of prismatic work-parts – for example, gear boxes) the fabrication of a work-part can start on a machine tool immediately after completion of the any other.

To achieve these requirements, manufacturing processes must have the necessary capacity to take on the required volume and the changes in the product mix. In addition, it is assumed that many of these part types, their order arrival rates and sizes will not be known in advance prior to

production, not to mention at the time when the production system was designed. For more features of the selected model of the manufacturing system see Section 1.4 "The manufacturing systems targeted in this research".



Legend:

WSi -Production workstation	WS1 - Machining Center	WS7 - Lathe
ISWS - Input system workstation	WS2 - Mill	WS8 - Lathe
OSWS - Output system workstation	WS3 - Mill	WS9 - Drill Press
IP - Input transport port	WS4 - Mill	WS10 - Grinding machine (plain)
OP - Output transport port	WS5 - Drill Press	WS11 - Grinding machine (cylindrical)
	WS6 - Lathe	WS12 - Grinding machine (cylindrical)

Figure 8-6. A manufacturing system model

8.1.4.1 Workstations

In theory, the job shop model can consist of an infinite number of workstations, each capable of a distinct operation for specific jobs. As depicted in Figure 8-6, three different kinds of workstations constitute the simulated manufacturing system resource model: a system input workstation (IWS), a production workstation (WSi), and a system output workstation (OWS). At the system input workstation, work-parts (row material or semi-products) enter the system. Fabrication of parts is performed on production workstations. At the system output workstation, work-parts are removed from the palette and are stored in a magazine or warehouse of finished

products. Workstations, depending on the level of automation, can operate as manual, semi-automated, or fully automated subsystems. In a fully automated workstation, all processing activities are autonomously controlled by a workstation controller and by a few equipment level controllers located inside the workstation.

For the purposes of this study, it is supposed that all production workstations in the manufacturing system model are fully automated. Therefore, the workstation "structure" that is depicted in Figure 8-2 is adopted for each production workstation in the system. To the contrary, the system input and output workstations in the manufacturing system model are assumed to be manually operated. The layout of these workstations is depicted in Figure 8-7.

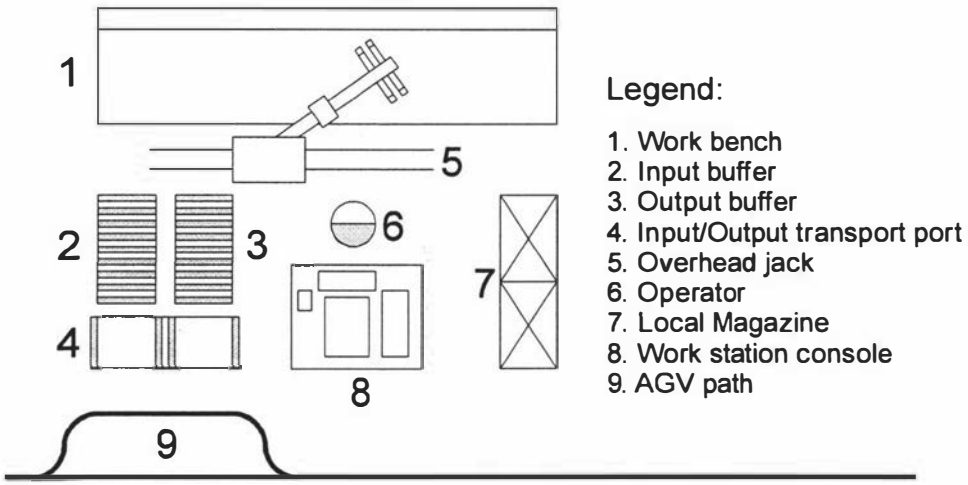


Figure 8-7. A layout of the system input and output workstations

As suggested in the Optimised Production Technology (OPT) approach, to ensure that there is always work at the bottleneck workstations the number of allowed work-parts in the input buffers of the bottleneck workstations should be able to be increased to provide larger safety stocks at the front of the bottleneck machines. Thus, whenever one job is completed, another is ready to go on the bottleneck machine. If the production of work-parts involves assembly lines, then to protect the assembly schedule against shortages that could severely cut the output of the manufacturing facility, a safety stock of completed parts from the bottleneck workstations should be held before the assembly section. Part shortages that may be encountered for the parts that flow through the workstations that are not considered as bottlenecks will not cut production.

8.1.4.2 Jobs and operations

Jobs arrive into a system randomly and each job needs to be produced to individual customer requirements. Because the arrivals of orders and their specific product types may not be known in advance, determination of job lead times and due dates often requires the calculation of a ***total work content (TWK)*** to be performed in the shop for the work-parts (part orders). This principle was used, for example in [Conway, 1965, Mahmoodi et al., 1999, Wemmerlov & Vakharia, 1991] to determine due dates for work-part orders to be manufactured in work cells. Total work content includes the combined values of the part family set up and processing times at each stage in the shop.

On arrival, a job is immediately released onto the shop floor, that is to the system input workstation. Each job can have an arbitrary number of operations and in practice will follow its own unique route for performing these operations. However, the routes for different work-parts will be different. It is assumed that each processing task (consisting of one or more processing operations) can be performed on one or more workstations. The number and type of operations, the operation times, and alternative workstations (processing routes) are determined in advance,

"off-line", for each job by a processing plan. Data from process plans are contained in ***Processing Route Tables (PRTs)*** that represent a unique repository of information for each job (see Section 11.2.1.2 "Getting the Processing Route Table (PRT)" for more information about PRT files). In this model processing time, which is sum of the operation(s) processing time(s), plus set up time(s), plus waiting time(s), for any task for any job at any workstation, is different for different work-parts – it does not have to be constant, as is the case with most other simulation models. Therefore, the time required at each workstation visited by individual jobs may differ. All jobs are treated as single jobs, though they can belong to some unique family, in which case successive jobs will require no set up time between jobs on any workstation. Details of how jobs (work-parts) flow through the system are given in Section 11.2.2 "Activities during the production process".

8.1.4.3 Transport of materials between workstations

Material flow is an integral part of the design of any manufacturing system. Designers of manufacturing systems are concerned not only with the specification of individual system components and the associations between them, but also with the interaction of the material flow system within the manufacturing system itself. In this study, the movement of materials between workstations is envisaged to be performed by an automated guided vehicle (AGV) system consisting of multiple vehicles.

The transport and machining operations can be conducted using the same palettes that have jigs, fixtures, and clamping devices for orientation, centring, and fixing work-parts on those palettes. Index-pallet changers are installed on workstation input and output transport ports and they are used for transferring palettes from AGVs to the workstations and vice versa. Input/output transport ports provide the physical interface between AGVs and workstations.

Depending of the type of manufacturing system (that is, whether it produces individual work-parts or work-parts in batches) we could distinguish two diverse kinds of batches in the system (as in Optimised Production Technology (OPT) approach): a ***transfer or transport batch,*** - quantities that are moved from workstation to workstation, and a ***process batch,*** - the total lot sizes released to the shop. In essence, no processing of work-parts can start until at least a transfer batch is allocated to a workstation.

Hence, in our manufacturing system reference model, in the case of batch production, it is assumed that the original order quantities (released to the shop for manufacturing) can be split into smaller transfer batches - predetermined integral fractions of the original order quantity. "Transfer batches" can flow immediately to the next operation prior to the operation's completion of the entire "process batch" at the current workstation. This approach is used, for example, by Jacobs and Bragg in their work on the ***repetitive lot concept*** (frequently referred to as "lot-splitting" or "overlap" scheduling) [Jacobs & Bragg, 1989]. Transfer batches provide a workstation with the flexibility to start producing an order before it is completed at the previous workstation. Such flexibility, as reported in [Jacobs & Bragg, 1989], reduces the average order flow time and work-in-process inventory, improves machine utilisation, cuts set up times, and smoothes the workflow in the shop to yield better use of the system capacity[1]. This flexibility also means that the ***operation batch size*** (the number of units produced during a given workstation set-up) can vary between the transfer batch and the original order quantity.

[1] The Jacobs and Bragg study indicates that repetitive lot concepts can provide the benefits of small lot production without requiring capacity increases or investment in reducing set-up times. They concluded that in spite of the possible rise of material handling costs and a more complex mechanism for tracking orders in a shop, the repetitive lot concept appears to be a promising new integrative approach for improving manufacturing performance.

8.1.4.4 Cutting tools, jigs and fixtures

There are three locations at which cutting tools, jigs and fixtures can reside:

- On a machine (tool magazines and machine's work tables);
- In a workstation local magazine; or
- In the central magazine of the manufacturing facility.

The most frequently used "standard" items should be located inside workstations. Modern machines are usually equipped with tool cribs that can accept several tens of cutting tools. Tools, jigs and fixtures, which are used in the current machine setting (tools in tool cribs and jigs and fixtures that are on a machine worktable), are defined in the system as located "On machine". Tools, jigs and fixtures, and other accompanying accessories that are used in everyday workstation machining operations, but are not a part of current machine settings, should be placed in the workstation's local magazine (location for these items in the system is defined as "Local magazine"). On the other hand, rarely used and usually quite special or expensive items that are shared among workstations should be stored in the central magazine (location for those items in the system is defined as "Central magazine)". It is apparent that depending on its type and location, each item will require a different set up time. The values of set up times can be determined empirically and stored in the local workstation database (as was done in the test-bed application). These data are used later in the process of bid generation that is explained in Section 11.2.2.2 "Forming bids and sending them back to the source agent".

8.1.5 Relationships in the proposed manufacturing system model

As noted in Section "2.1.3. A desirable structure for a model of manufacturing system" a structure is concerned with how the individual components of a system relate to each other and how the nature of these relationships determines the overall system behaviour. In manufacturing, three types of relationships are significant.

- The flows between components of the system. In any manufacturing system, two of these are critical, namely, the flow of material and the flow of information. In some contexts the flow or movement of people also is important (for example, in U-shaped lines).
- The communication network relationships that connect people to other people or connect people and machines.
- The organisational relationships between the people associated with the system; that is, who controls whom.

Material flow[1]. In the proposed manufacturing system the flow of materials (jobs) through the system is established by making choices between alternative flow paths. When decisions are made about the most appropriate processing path at the time during which a part is actually being produced, the material flow control is said to be performed "on-line". In the first instance, this "flow" refers to the "processing flow". In addition, we need to consider the "transport flow". Actual transport flow in a system, which is also performed "on-line" by default, generally follows the processing flow, although, it may be slightly different because a transport control system will try to avoid traffic congestion by selecting alternative transport routes. More details

[1] Note that in this case, "material flow" refers to the flow of work-parts being produced, although, in its broadest sense, the "material flow" refers to flow of all types of physical items in the system.

about how the choices between alternative flow paths are made in the proposed system are given in Section 11.2.2 "Activities during the production process".

Information flow. If there are alternative material paths through the system for a given part type, then it would be necessary to make control decisions to select the path taken by a specific job. The choice of which path to follow will depend on the available information (such as the workload in the workstations to which material could be sent). It is apparent that it is necessary to consider not only the material flow paths but also the structure of the information flows and how this information is used by the control system. In the proposed control system most of the information flow (control information for making decisions) among distributed production units on the shop floor is performed in a heterarchical - horizontal - manner. However, the secondary or vertical flow of information (information about the state of the system - such as information regarding monitoring quality and reporting operating statistics) will still be incorporated into the system. This flow will exist between plant level and other production functions such as, for example, the process planning department, the material supply department, and other similar departments in a production system. The vertical flow of information is also present inside the workstations (the flow between the workstation controller and other equipment level controllers). The way in which control information is used in the proposed heterarchical shop floor control system is explained in more detail in Section 11.2.2 "Activities during the production process".

In the proposed manufacturing system ***communication*** is conducted via a local area network (LAN) that is viewed as an integrated medium among all entities in the system. Note that there are two communications flows: firstly, within a workstation between the workstation controller and its subordinate elements, and secondly, among workstations. The LAN has to provide only the bandwidth required for the communication among workstations. More details about local area networks and communication protocols used in developed application are given in Section 9.1 "The choice of tools for the development of a test-bed application of a distributed, heterarchical shop floor control system".

The organisational relationships between the people involved in the manufacturing process are not considered in this study.

The main characteristics of the manufacturing system model that was adopted in this project are summarised in the next section.

8.1.6 Manufacturing model characteristics

- The manufacturing system model is comprised of a number of workstations that are capable of performing a range of different operations.
- The number of workstations permitted in the manufacturing system model is theoretically unlimited. The only limitation would ultimately be the loading on the communication system but the communication systems of modern networks should be able to cope with many hundreds of workstations.
- Workstations are autonomous production entities that are able to accomplish all production activities for their part fabrication operations without interaction with any higher control level during that process.
- The number of versatile part types that can be manufactured simultaneously is restricted only by the physical capabilities of the machines.
- The number of parts fabricated simultaneously in the system is also restricted only by the physical capabilities of the machines in the same way as for different part types, and this number can vary with time.

- The number of processing tasks (operations) in a job is unlimited.
- The target manufacturing environment is a system able to produce parts either as single part items or in small size batches.
- Processing times for each operation can be different.
- Set up times for each workstation can be different and involve tools, jigs and fixtures settings.
- Set up times of shared resources are predefined values based on empirical data. Set up times are dependent on the locations of the shared resources.
- Transport times can be mapped by using different values depending on the physical distance between the source and destination workstation and the characteristics of the transport system.
- The number of allowed work-parts inside the workstation's input and output buffers is strictly limited. This number is restricted either by the physical size of the buffers (the number of pallets that can be stored inside them) or by the total processing time that is required for processing all work-parts that are located in the buffers (for example not more than 4 hours). Limiting the size of the input and output buffers contributes to low levels of work in process, making the overall system more responsive to changes.

After the model of manufacturing system had been defined the next step was to select a computer platform for developing a multi-agent heterarchical shop floor control system. This is discussed in the next chapter.

Chapter 9. The selection of a development platform

Before proceeding with the development of the agent-based manufacturing control system developed in this research project, a development platform (including hardware and software) needed to be adopted. Accordingly, this chapter discusses selection of a platform for developing a test-bed application of a proposed multi agent shop floor control system.

9.1 The choice of tools for the development of a test-bed application of a distributed, heterarchical shop floor control system

The technologies used in developing agent-based systems include the use of dedicated hardware and software platforms and tools specifically designed for that purpose. They provide the "right" environment for fast developing, debugging, and testing of software agents (refer to Section 6.4 "Overview of standards and tools for developing multi agent systems" in which some of these platforms are briefly addressed). However, many of the software development systems were under research and development in 1997 and were not available for this project at that time. Therefore, one of the challenges in this development project was the development of a multi-agent system using the software tools for rapid application development (RAD) that were readily available, affordable and capable of operating on general-purpose personal computers with a commonly available operating system. The choice was made to adopt the following operating platforms:

9.1.1 The selection of computers

On the basis of a literature review it was considered that for developing and testing an agent-based system, personal computers (PCs) connected in a computer network represented a solid platform for implementing agent based architectures. The choice of a hardware platform was made between two types of the most common PCs: Macintosh and IBM compatible computers. Taking into account the number of computers required for this project, their accessibility and availability (including computer components) the ***IBM compatible PCs*** were chosen since Massey University had a management laboratory which had 8 networked personal computers which it was believed could be available in the initial stages of the project. In the event it turned out that these computers were heavily used by undergraduate classes and so some low cost PC's had to be obtained. Funding did not allow the purchase of "state-of-the-art" machines for this project. Another factor in the decision was the candidate's familiarity with the PC's hardware components and the operating environment.

9.1.2 The selection of an operating system

For the purposes of this project the most important requirement of an operating system (in addition to that it is able to run on Intel based microprocessors) is to support multitasking

(multithreading) execution of programs, and to have a good support for network communication. For IBM compatible PCs, the most popular operating systems are Windows, though others can be used, for example Unix and Linux. Since Windows operating systems dominated the personal computer world[1], the system of choice at the time the project was initiated was ***Windows 95***. In addition to fulfilling the basic requirements in terms of providing multitasking and multithreaded features[2], a wide variety of supporting programming tools and other software products existed for the Windows systems. As discussed later in Section 12.1.9 "Selection of a computer platform and software technology" this choice was not appropriate and instead of Windows 95, Windows NT 4.0 should have been used. However, it should be noted that Windows 95 was readily available at the start of the project whereas Windows NT was not. If the project were staring now the newer operating systems such as Windows 2000 and Windows XP (both professional versions) should be used. These would give the opportunity to restructure the current software to perform more reliably and be better structured.

9.1.3 The selection of a network

The area of interest in this project was a Local Area Network (LAN). Two types of LANs were considered in particular: Ethernet and Token Ring. Again because of the availability and overall support, an ***Ethernet network***, as the most popular and dominant network was selected. Features that distinguish this network are given in the following (a detailed overview of network technology can be found in a number of text books dedicated to this matter, examples include [Chowdhury, 2000, Comer, 2000, Cummins & Philip, 1999, Tanenbaum, 1996]). The accessing method is ***carrier sense multiple access with collision detection (CSMA/CD)***. Ethernet network supports both a star and a bus (linear) topology. For creating an experimental LAN a ***bus topology*** was chosen. This topology was adopted because it was less expensive to implement (a star topology required an additional central hub for networking the computers and requires all the nodes to be connected back to that hub). Because of well-developed technology and low price (compared with fibre optic cables), ***thin coaxial cables with BNC connectors*** were used in the experimental LAN. Taking into account that the experiment was to be undertaken in a clean experimental environment (office-like, rather than in manufacturing factories) and the setting-up costs for an experimental LAN, an alternative option could have been to use twisted-pair wires. However, this wiring scheme (star topology) would require procurement of wiring hubs and newer NICs (equipped with J45 connectors for attaching twisted-pair cables). This would have further unnecessarily increased the LAN establishment costs. In practice, a coaxial cable based network is less likely to suffer from interference from the high switching currents and therefore is a better choice for a harsh machine shop environment[3].

For standardisation purposes, multi layer models such as the ISO-OSI Reference Model are used to specify the components of a network. At the physical and lower network layers an Ethernet LAN uses ***Ethernet*** or the ***IEEE 802.3 protocol***. At the higher layers of the ISO-OSI Reference Model there are a variety of standard protocols from which a selection can be made. For developing the model of a multi-agent system the ***Transmission Control Protocol/Internet***

---

[1] By some estimates it was running, on 90% of all personal computers.

[2] Windows 95 supports full preemptive multi-tasking for 32-bit applications (ran multiple applications simultaneously in the extended memory), and provides support for multithreaded applications through the OLE (Object Linking and Embedding) calls from multiple threads.

[3] Due to ease of installation, maintenance, and lower cost, the Shielded and Unshielded Twisted Pair (STP/UTP) wires are the preferred method of connection. The STP wires provide more immunity against electromagnetic/electrical noise interference than the UTP wires. The Coaxial cable is more expensive but offers much better noise immunity than the twisted pair medium [Rabiee, 1999].

Protocol (TCP/IP) suite was adopted. This choice was made because TCP/IP suite has become the de facto standard for transmitting data over networks, particularly the Internet. Even network operating systems that have their own protocols, such as Netware, nowadays also support TCP/IP.

9.1.4 The selection of Network Interface Cards

Because a network architecture (bus), medium (coaxial cable), and media access control protocol (CSMA/CD) have been chosen the options for Network Interface Cards (NICs) have narrowed. After matching the medium and the protocols, there are additional alternatives that need to be considered when selecting LAN adapters:

- The vendor (what kind of support and quality the vendor can provide, price, etc.).
- Compatibility with the hardware architecture of the computer into which a LAN adapter is to be installed (ISA, EISA or PCI compatible cards).
- The architecture of LAN adapters (LAN adapters for IBM compatible systems can have 8-bit, 16-bit, or 32-bit architectures). The architecture affects the speed of transferring data between the computer and the medium (32 bits at a time, against 8-bit or 16 bit groups) but does not affect the speed at which data are transferred over the network medium.

For the experimental LAN created for this project, the following NICs were used:

- SMC 8416 (EtherEZ) ISA Ethernet adapter manufactured by Standard Microsystems Corporation.
- The Accton EN1650 ISA Ethernet adapter manufactured by Accton Technology Corporation. Included with the adapter were the NDIS3 driver ACCNT.SYS and the NDIS2 driver ETHNE.DOS.
- The NE-12A PCI Ethernet card manufactured by KTX Corporation. It had built-in Novell NE1000 and NE2000 fully compatible functions and came with the corresponding drivers. (Computers with these NICs used standard Windows 95 drivers).
- Realtek RTL8029(AS) PCI Ethernet adapter from an unknown manufacturer. It uses NDIS 3.1 driver named E32ODI.SYS.

9.1.5 The selection of programming tools

Most projects or the research work that was reviewed used traditional programming languages, such as: Java with CORBA (Common Object Request Broker Architecture), C++, C, Lisp, SmallTalk, and Prolog to develop agent-based manufacturing systems. Several other program languages also can support infrastructure requirements for agent applications, among them are: Gensum, NeXT, Eiffel, Software PIM (Parallel Inference Machine), and SWARM, but none of these provided enough rapid development facilities to be considered as suitable for the development of this project. Some of them could have been used by a sophisticated and experienced programmer but would have greatly increased the software development time. Recently, some research and development efforts to build appropriate computing platforms for reducing the development time and costs of agent applications have been undertaken. Some of these platforms are reviewed in Section 6.4.2 "Agent development platforms".

Due to the considerable scope of the project and the limited time available, the primary interest was in a programming system that enabled programmers to quickly build working prototype applications – so called systems for Rapid Application Development (RAD). Two of the most

popular RAD systems for Windows available at the start of the project were Visual Basic and Delphi. The programming language of choice was ***Visual Basic*** (VB) - initially version 4.0 and, later on, version 5.0. There are several reasons that influenced making such a decision. Firstly, prior to the development of a multi agent application the author had already established familiarity with the VB development environment and syntax (a couple of small VB applications were developed as a part of another research project - a rudimentary VB application for controlling a pneumatic manipulator by using Allen Bradley PLC and a simple client-server application on a TCP/IP network for chatting over a LAN). The second reason for adopting VB was a Microsoft policy decision to adopt VB as a programming (macro) language for Microsoft's applications[1]. Thirdly, since VB already has been used to a great extent by other peer researchers and it was believed that it would be much easier and seamless to integrate this project in the future with other projects being conducted at the Department of Production Technology (now the Institute of Technology and Engineering) at Massy University. Finally, Visual Basic (VB) offered the possibility of building applications by using software components that are based on ActiveX technology. ActiveX technology provides "language independence" so that some modules of an agent application can be developed, if necessary, in other languages that are more appropriate for addressing some specific issues rather than using VB. (In Section 12.1.10 "Design and development of the workstation agent" it is discussed how multithreading issues that were encountered during the development of workstation agents can be resolved using this feature). Also, ActiveX controls, that are available from a variety of vendors, can be used in the same way as the standard VB built-in controls to provide new functionality for Visual Basic applications. Two such components (ActiveX controls) used in this project are "IPDaemon" and "IPPort". As it will be discussed later in Section 10.2.1.2 "The communication module", these ActiveX controls were fundamental for developing a client/server application for communication over a LAN using the TCP/IP protocol. These four reasons biased the selection of VB against Delphi which was the second alternative.

The next chapter discusses the design, development, and implementation of a model of a distributed heterarchical shop floor control system.

[1] For example, Microsoft Office applications Word, Excel, and Access come with a version of Visual Basic named as "Visual Basic for Applications" with which users can write small programs to customise and automate these applications. Examples include creation of custom commands, menus, dialog boxes, messages, and so forth.

Chapter 10. Design, development, and implementation of a model of a distributed heterarchical shop floor control system

The selection of the development platform is discussed in the previous chapter. The focus of this chapter is on the development of a heterarchical shop floor control (SFC) system as discussed in Chapter 7. This chapter describes the envisaged system while its operation is described in Chapter 11.

The chapter is organised as follows:

Section 10.1 describes the architecture of the heterarchical SFC system proposed in this research study.

Section 10.2 defines the term "workstation agent" as it is used in this research. Afterwards, a functional model of a production workstation agent, together with its component modules, is described in more detail. Finally, the co-operation and communication between workstation agents is covered at the end of the section.

Section 10.3 describes the methods used for scheduling jobs in the proposed model of a distributed heterarchical SFC system. In this system it is advantageous to classify the scheduling processes as either global or local. Scheduling jobs among workstation agents (global scheduling) as well as inside the workstation agents (local scheduling) is discussed in more detail in this section.

10.1 The proposed control architecture of a heterarchical shop floor control system

Following most of the design principles given in Section 5.8 "Design principles for highly distributed heterarchical shop floor control systems", and bearing in mind the adopted physical structure of the manufacturing system, a model of heterarchical shop floor control system used in this study was proposed (see Figure 10-1). The system is based on workstations as the basic organised autonomous production units of the manufacturing system. In the system all workstations are treated equally and mapped as autonomous control units – ***workstation agents***. Therefore, each workstation has a corresponding agent – a software package, which is installed on the workstation's computer. A functional model of workstation agent and its constitutive modules are discussed in more detail in Section 10.2 "The workstation agent". All workstation computers in the system are linked by means of a Local Area Network (LAN) that is used as the integration and communication medium. Production workstation agents are identical software packages installed on each production workstation computer. The program's realisation of the system input workstation agent and the system output workstation agent are slightly different from the realisation of the agent of the production workstation.

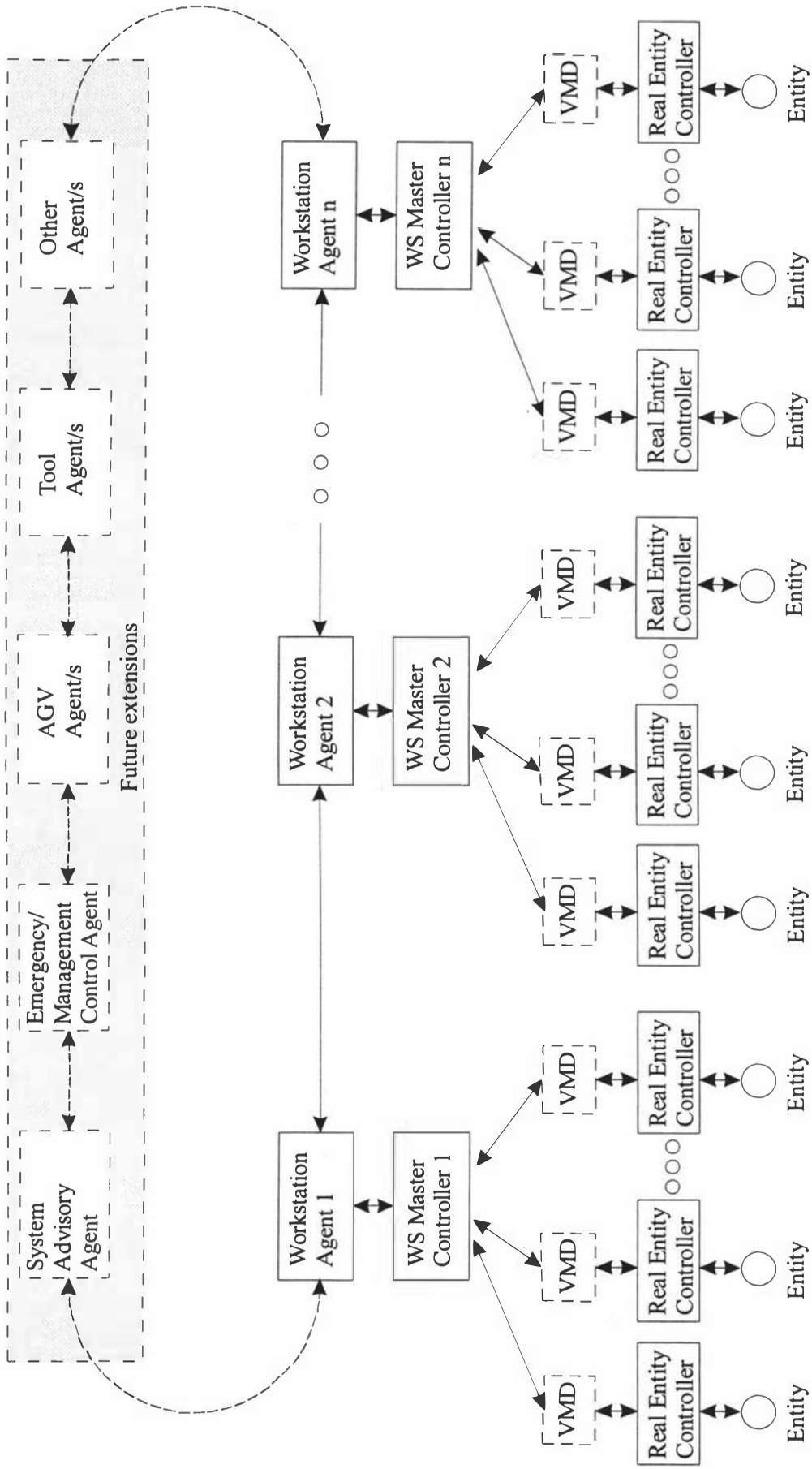


Figure 10-1. The proposed manufacturing control architecture

In contrast to some other researchers, who have adopted the approach of modelling work-parts as separate part-agents (in addition to modelling machines as machine-agents), the proposed heterarchical shop floor control system has been designed in such a way that the system is made fully operational by using only workstation-agents. Therefore, work-parts in the proposed system remain "dummy" elements of a manufacturing facility without any need for built-in microprocessors or any other form of "artificial intelligence" as has been suggested as an approach to heterarchical system control, for example in [Upton, 1997].

While in the proposed system, shop floor control is indeed performed in a distributed, highly heterarchical manner, it is important to note that the overall manufacturing control structure can also be viewed as a "hierarchical" - two level scheduling and control system:

- A part of manufacturing control system that is concerned with scheduling and control at the shop floor level, that is, the co-ordination among a number of different workstations, is responsible for carrying out two rudimentary functions: 1) for routing work-parts between workstations through the shop and 2) for controlling the interaction between the workstations and the material transport system. Both functions in the modelled manufacturing system are performed in a distributive manner.
- A part of manufacturing control system that is concerned with planning, scheduling and control at the level of the workstation, that is, among elements inside the workstations is responsible for co-ordination of all activities inside workstations, including material handling operations, machining operations, and storage operations. Also, this level is in charge of the local scheduling of work-parts on the machine. At the workstation level scheduling and control can be performed either in a centralised or hierarchical manner.

Therefore, to be more precise, the manufacturing control architecture proposed in this research represents a ***hybrid system*** composed of heterarchical (shop floor level) control and hierarchical or centralised (workstation level) control, refer to Figure 10-1. At the shop floor level, control among workstations entities is performed in a strictly distributed, heterarchical manner while the control inside workstations may be approached either in central or hierarchical manner, as it will be discussed in the following text.

The proposed concept emphasises the advantages of both architectures, namely, heterarchical and traditional, hierarchical or centralised. As already mentioned in Section 5.4.3 "Heterarchical control architectures", heterarchical control among workstations reduces the complexity and costs of creating an overall manufacturing control system, while at the same time, it significantly increases the levels of modifiability, extendibility reconfigurability, reliability, fault-tolerance, flexibility, and adaptability. On the other hand, the complexity of a workstation is sufficiently simple in most cases that it can be successfully approached in centralised manner. Having a large amount of local data, and closely coupled entities, centralised control inside workstations resolves all resource sharing conflicts in an efficient way. With the current stage of computer technology, a central workstation control unit can easily cope with the computational requirements that are necessary for creating "optimal" schedules within the workstation and for having a "complete knowledge" about the states of all of the workstation's elements, including their schedules, variables, and all relations between these variables. Furthermore, the data required for creating control information for workstation entities are locally available and can be accumulated very fast. It makes real-time control of workstations feasible. However, for practical reasons (machinery that is allocated to the workstations is usually produced by different manufacturers and is typically equipped with the manufacturer's particular controllers) the workstation control model will often follow a hierarchical structure as in conventional hierarchical control architectures.

The key advantage of the proposed shop floor control system is that the workstation agents can support both workstation architectures. Though the agents for centrally controlled workstations would be slightly different from the agents of the workstations that are hierarchically controlled, from the shop floor control point of view they are all seen in the same way. Agents are a common abstraction for all workstations on the shop floor. With no differences between the agents, the shop floor control system allows simultaneous existence of both workstation architectures in the system.

In this study a hierarchical workstation architecture is assumed, and accordingly, a corresponding workstation agent was built (a module that simulates workstation activities is part of the workstation agent). To increase the modifiability and extendibility of the hierarchical architecture, it is envisaged that information flow between a workstation's master controller and other equipment controllers inside a workstation is carried out through virtual manufacturing devices (VMDs), as depicted in Figure 10-1. Introducing the concept of VMDs, which actually represent an interface (or using computer terminology: a "driver") between the master workstation controller and any particular equipment controller, a workstation agent (simulated control module) considers all equipment controllers in the same way. This allows the development of a common set of control instructions (messages, for instance "start a machine") for all controllers independently of specific protocols that may be used by different brands of machines and equipment producers. It is the task of an appropriate VMD to map workstation control instructions and make them understandable to a particular equipment controller. As can be discerned, a strict superior-subordinate relationship between the workstation master controller and the equipment level controllers still exists. What is changed is the way in which shop floor control is performed.

In designing a heterarchical shop floor control system, that is workstation agents, it is very important that the structure of the workstations is known, and whether a particular workstation is manned or not, that is, what level of automation is applied inside the workstations. In addition, in the case of fully automated systems it is also important to know which control architecture is in use. As noted in Section 8.1.4 "The proposed manufacturing system structure – a general description of the manufacturing system resource model" three types of workstations can be distinguished:

- *Conventional workstations* are equipped with conventional machines and they are staffed by operators who are responsible for controlling both processing and material handling activities. Operators complete scheduling in these workstations and hence the workstation subsystems have limited capability to produce good local schedules.
- *Semi-automated workstations* are still manned but they are equipped with CNC machines. In these workstations, operators perform tool loading, monitoring, and inspection tasks. Some control and scheduling operations may be performed with the help of computers.
- *Fully automated workstations* involve applying computer-based technology to processing and material handling activities and they are not manned. Control and scheduling operations are performed by workstation computers.

The test-bed application was intended to support any operating mode. At present, the execution of workstation machining and material handling operations is simulated, so that the level of automation implemented in the individual workstations is not of primary importance for the functioning of the application. However, since all the activities in the simulation are event driven, the current application with the minor changes in the user interface could be deployed for solving the part allocation problem in a real manufacturing environment where human operators are employed (manual and semi-automated workstations).

This project does not address precisely the links between a workstation agent and a workstation controller. Referring to Figure 10-1, there are two possibilities. When developing a real workstation agent, the functions of workstation controller can be incorporated into the agent, and thus become an integral part of the agent (as has been done in the simulated model), or the agent can be closely coupled with the existing workstation controller. In the later case, workstation controller from the agent point of view is seen as an external entity to which the agent has direct and very close relationships.

10.2 The workstation agent

Workstations and work cells represent the basic functional, constituent elements of the manufacturing facility. From the heterarchical shop floor control point of view they are treated equally and mapped as autonomous units - agents. For the present, the implementation of agents is restricted to workstations. There is a physical correspondence between workstations and agents. Any workstation has its own workstation level software – a ***workstation agent***. Agents are mutually hardwired together through an efficient local area network (LAN) forming a ***Multi-Agent System.***

The workstation agent in this belongs to the software agents' subclass (See Section 6.1.1 "Agent definitions"). *Under the term workstation agent we consider the entire workstation level software. A workstation agent is a self-directed program that acts on its own initiative and which resides on an individual PC sized computer.* The workstation agents are organised, distributed entities that are mutually interconnected through a general Local Area Network (LAN), which is used as the communication medium.

10.2.1 The functional model of the workstation agent

As we have seen in Section 10.1 "The proposed control architecture of a heterarchical shop floor control system", the architecture proposed within a framework for the development of a test-bed application of distributed heterarchical shop floor control system was based in a set of autonomous and co-operative workstation agents. The next step in the framework design was to specify an architecture for a generic workstation agent that will belong to the system architecture. We view a workstation agent as software consisting of four major functional components:

- A local knowledge database;
- A communication module;
- A decision making module; and
- A workstation control module.

Figure 10-2 depicts a modular architecture of a generic workstation agent. In the next section each of these modules will be described in more depth. A layout of an elementary production workstation configuration on which the corresponding agent is based is depicted in Figure 8-2.

For the functional model of a workstation agent, shown in Figure 10-2, (more precisely, a production workstation agent) a brief description of each of its modules is given in the following text of this section.

10.2.1.1 The local knowledge database

The local knowledge database is a relational database at the workstation level, which deals with the production data necessary for part processing in a particular workstation. For example, for

each part machined on a given workstation (i.e. part ID), the local database might hold a few programs for a robot to handle the part through the workstation, a CNC program for part machining, a list of required cutting tools, a list of additional fixtures and jigs if any, data specific for part processing, tools life times (for the manually operated machines), etc.

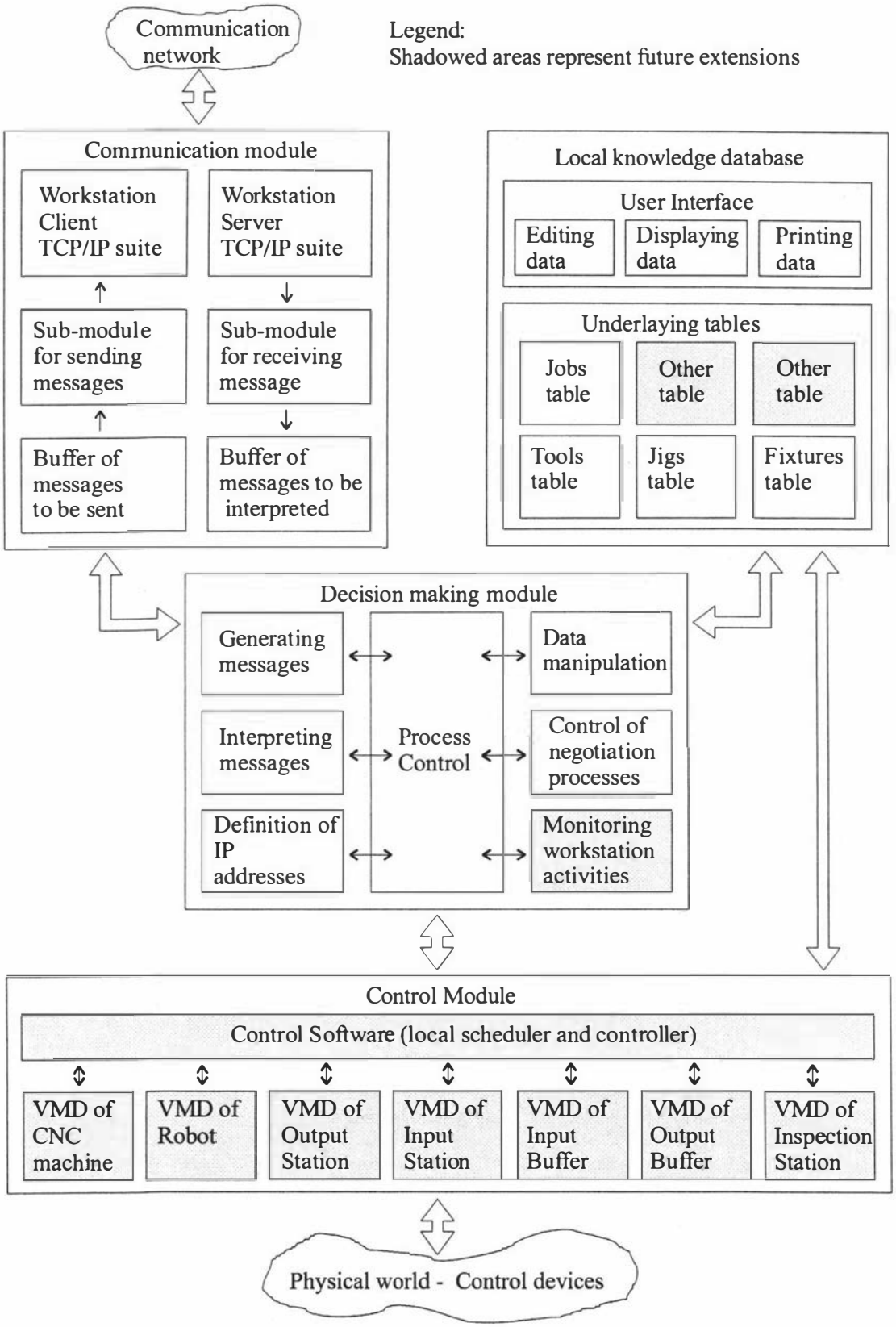


Figure 10-2. The functional model of a workstation agent

Figure 10-3 depicts a relationship diagram of a workstation agent local database. The database currently comprises five tables: Jobs, Jigs, Fixtures, Tools and Work-parts. Note that this is only an example that was developed to demonstrate the concept – the operation of a test-bed application. In subsequent developments, the content of a database needs to be modified and extended to accommodate any new requirements imposed on the system. Examples include additional fields in the "Jobs" table for CNC and robot programs, or introduction of an entirely new "Transport Times" table (if the systems is realised without AGV agents) which would contain estimated transport times for transferring work-parts from different workstations to the workstation of which this database is part.

The table "WSParts" contains data about work-parts that have been processed by the workstation. This data is updated regularly (each time when some of the attributes are changed) and might be used for monitoring the current status of work-parts that are located in the workstation. When a work-part leaves the workstation, its details (actual values, for example about times, not estimated) remain recorded in this table. This data (as a "history note") can be used for performance analyses. Alternatively, the data could be part of the Jobs table (a matter of compromise between the application performance and the elegance of design).

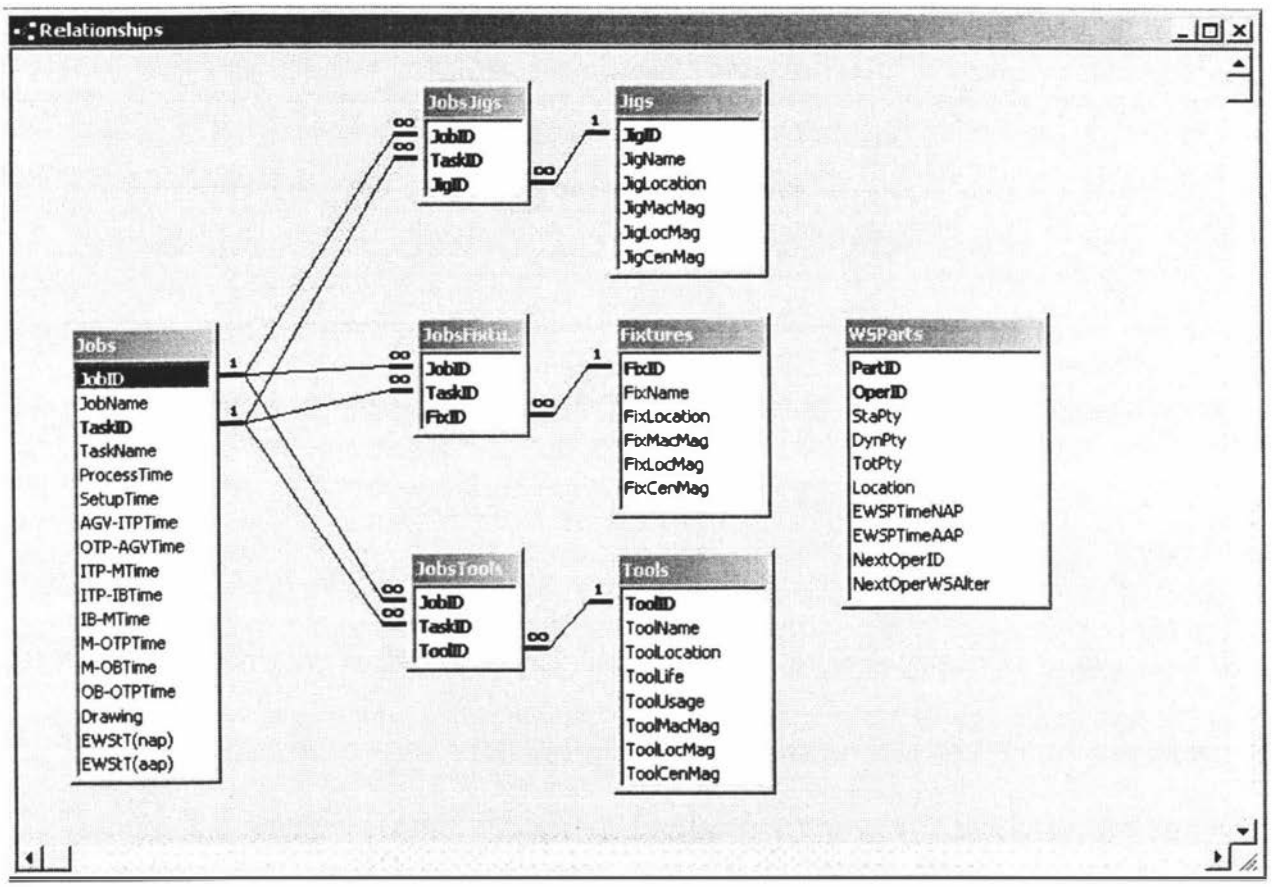


Figure 10-3. An example of a relationship diagram of workstation agents' local database

A segment of the local-knowledge database developed for this application, which represents an overview of requirements for Task 02 of the Work-part 10 (Job10), is shown in Figure 10-4. The figure depicts estimated times that must be defined for each task of any particular job that can be performed on the workstation. These times include:

- Part processing time,
- Set up time,

- Transfer time from AGV to input transport port (ITP-AGV),
- Transfer time from output transport port to AGV (OTP-AGV),
- Transfer time from input transport port to machine (ITP-M),
- Transfer time from input transport port to input buffer (ITP-IB),
- Transfer time from input buffer to machine (IB-M),
- Transfer time from machine to output transport port (M-OTP),
- Transfer time from machine to output buffer (M-OB), and
- Transfer time from output buffer to output transport port (OB-OTP).

In addition, there are also lists of the tools, jigs, and fixtures required, together with their accompanying set up times that are dependent on their locations (on a machine, a local magazine or a central magazine). These times are taken into account when a workstation agent generates a bid for accomplishing a specific task and therefore must be defined before a work-part enters the system.

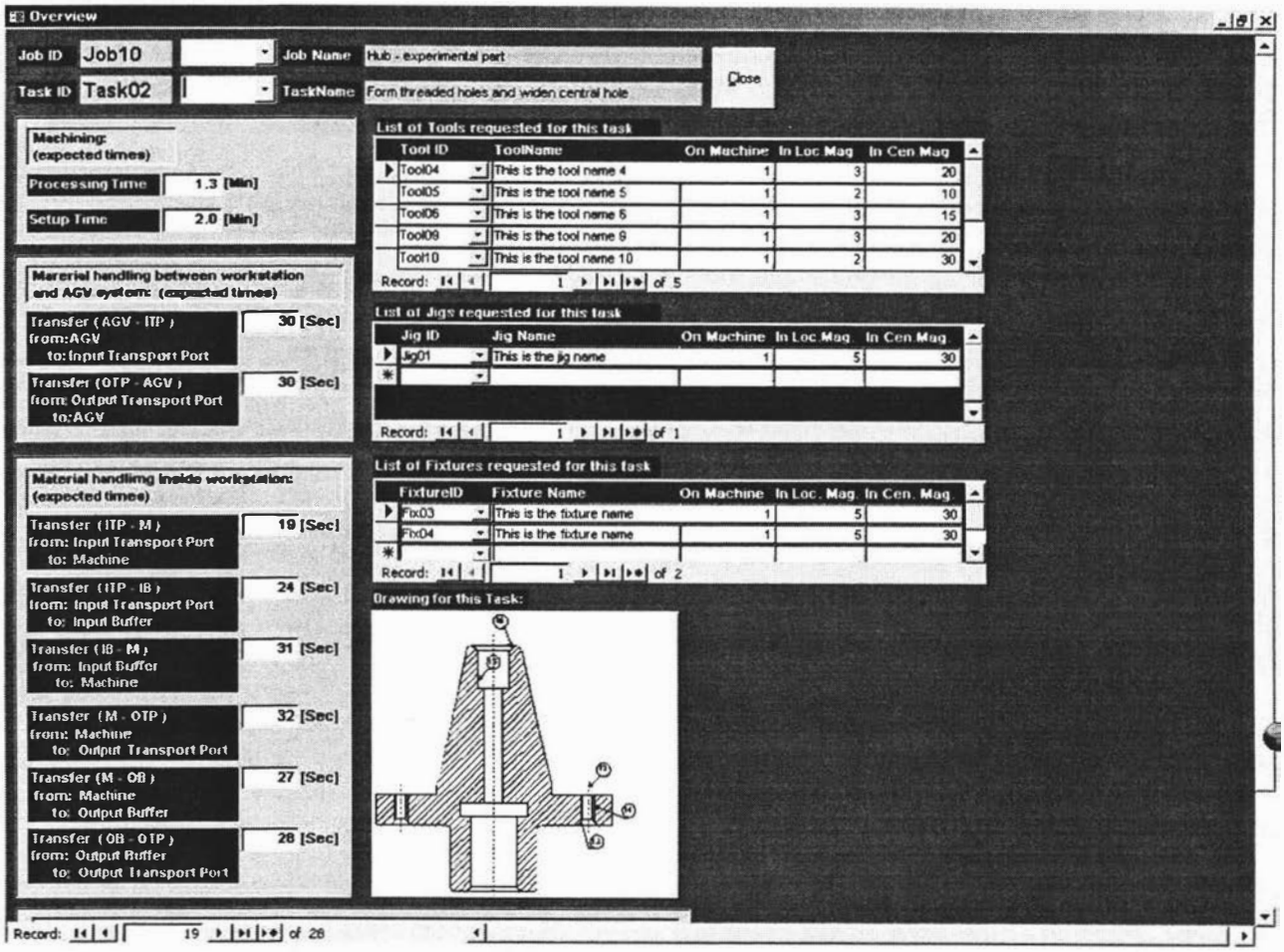


Figure 10-4. An overview of data related for Part 10, Task 02

10.2.1.2 The communication module

The purpose of a communication module is to provide reliable and asynchronous communication among agents via a LAN. A workstation client and a workstation server constitute the core of this module. The server is permanently in the listening mode so that it can receive messages from the other agents at all times. The server provides a separate connection for each agent that

wishes to communicate and can handle about 1,000 simultaneous connections. When a message is received, it needs to be immediately transferred from the server sub-module into the temporary buffer making the server available to receive other new messages. Because the time for receiving messages is usually shorter than the time for their processing, without this buffer either a received message could be overwritten by the new one, or the server resources would be kept unnecessarily busy causing extra loading on the network as messages for the server may need to be retransmitted several times before the server is able to accept them. When the module for interpreting messages is idle it checks for the messages in the buffer, and if it finds any, it takes the first message in the buffer and interprets it. Then it takes the next message and so on until the buffer is empty. In this way the messages are interpreted in the same order as they arrived.

In contrast to the server, which is active all the time, the workstation client is engaged only if there are some messages to be sent in the buffer (the buffer for messages to be sent). If the buffer has a message, the sub-module for sending messages activates the workstation client first and then passes the appropriate message to it. The client has two tasks: to establish connection with the agent whose destination address is contained in the message, and to provide reliable transfer of the message to the destination agent.

Development of a communication module of a workstation agent is based on the use of ActiveX controls IPDeamon and IPPort which in turn use "sockets", a subdivision of a network node reserved for a single application or process. "Sockets" is a general-purpose networking application programming interface (API), (a software object not a physical component), used in some operating systems (e.g. UNIX, Windows), to connect a user application to a network protocol. A program can send and receive, for example, TCP/IP messages by opening a socket and reading and writing data to and from the socket.

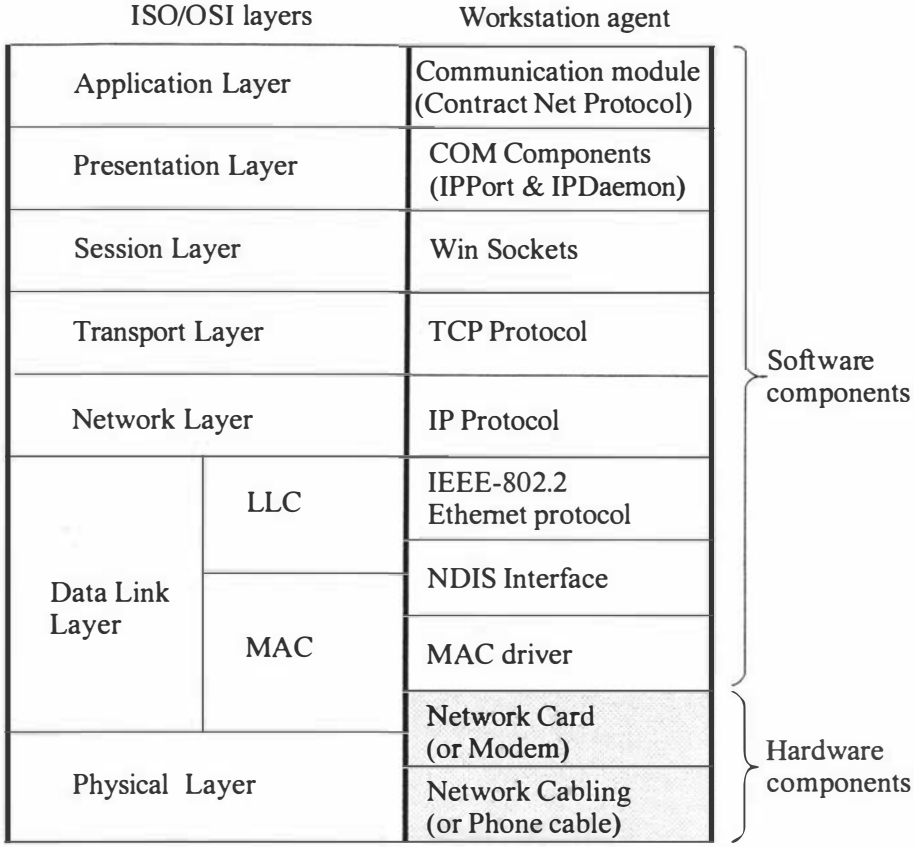
Each communication module of a workstation agent contains two levels of "abstraction of programming details". "Winsock" is a general-purpose network-programming interface that represents the first level. In relation to ISO/OSI reference model it is located at the session layer. (Deployment of software and hardware components of a workstation agent's communication module in relation to ISO/OSI reference model is depicted in Figure 10-5). Winsock saves programmers from dealing directly with the network data exchange, i.e. with the network protocols that are located at the lower network layers (in our case with the Transport Control Protocol - TCP). By using Winsock programmers rely on the Windows operating system to actually transport messages across the network correctly.

The second level of abstraction is represented by the ActiveX components IPDeamon and IPPort. These components simplify further development of an agent's communication module so that a programmer does not need to worry about manipulating the sockets. To send and receive TCP/IP messages, a programmer has to define the IP address and communication (IP) port of a remote host and to call an appropriate method for establishing connection. Once the connection is established, data can be read from and written to the port as discussed in the following text.

IPDaemon and IPPort controls are part of a package of ActiveX controls (15 controls in total) called "IP*Works!", which is a commercial product of the devSoft Inc. company[1]. In short, IPDaemon Control is used to create TCP/IP servers running on PC's connected to a TCP/IP network, while IPPort Control is used to communicate with a TCP/IP server through stream sockets. Basic information related to each of these components, their properties, methods, and

[1] devSoft Inc. P.O. Box 13821, Research Triangle Park, NC 27709, USA
email: sales@dev-soft.com Web: http://www.dev-soft.com Tel/fax: (919) 493-5805

events is given in IP*Works reference guide that comes as a help file "IPWORKS.HLP" together with the all other controls.



Legend:
MAC - Media Access Control
LLC - Logical Link Control
MAC driver - network card device driver
NDIS (Network Driver Interface Specification) - Part of Win9x/2000
IP - Internet Protocol
TCP - Transport Control Protocol

Figure 10-5. Relationship between the ISO/OSI Reference Model and the practical implementation of communication module

Utilisation of IPDaemon and IPPort ActiveX controls in VB version 4.0 represents an effective and efficient way to create, maintain and terminate a TCP/IP connection among computers in a LAN (and over the Internet). To develop a client or a server application there is no need to call low level Winsock APIs (Application Programming Interfaces[1]). By setting properties and invoking the methods of the IPWorks controls, it is possible to connect to a remote computer and exchange data in both directions. IPDaemon and IPPort controls make developing TCP/IP communication a relatively easy task. Utilisation of these controls in the sub-modules for sending and receiving messages is described in the following two subsections (Sections 10.2.1.2.1 and Section 10.2.1.2.2) in an example of a rudimentary set of instructions (including the most important properties, methods, and events) required to create a simple client-server application.

[1] API is a set of commands that an application uses to request and carry out lower-level services performed by a computer's operating system. An API makes it easier to develop a program by providing the building blocks that programmers use to write applications consistent with the operating environment.

10.2.1.2.1 The sub-modules for sending messages - the client application

To create a client application the IPPort ActiveX control is used. The client application must know the server computer's IP address (***RemoteHost*** property), as well as the port (***RemotePort*** property) on which it will be "listening." Then the ***Connect*** method is invoked[1]. Once the connection to the remote host has been established a message can be sent. Table 10-1 reviews a set of instruction performed in the client application developed for the workstation agent.

Table 10-1. A rudimentary set of instructions in the client application

Action	***Instructions***	***Description***
Establishing a connection	***IPPort1.WinsockLoaded = True***	Load and initialise the Winsock library.
	IPPort1.LocalPort = 0	Set the TCP port in the local host (client computer) where IPPort resides. If not set to 0 Winsock will choose the port number at random.
	IPPort1.RemoteHost = "192.168.10.1"	Set the IP address of the remote host.
	IPPort1.RemotePort = 3000	Set the TCP port in the remote host.
	IPPort1.Connected = True	Trigger a connection - An action property.
	Do Until CurrentTime < StartTime + 3 If IPPort1.Connected Then Exit Do DoEvents Loop	Wait maximum 3 sec for establishing connections.

Sending a message	If IPPort1.Connected Then
	IPPort1.DataToSend = "Test Message"
	Else
	MsgBox "Please, establish the connection first "
	End If
		If a connection to remote host has been established send a message. Otherwise ask for the connection to be established.

Receiving a message	Private Sub IPPort1_DataIn(Text As String, EOL As Integer) txtMsgReceived.Text = Text End Sub	This event is fired when data (in the form of complete lines) comes in. Data is contained in the Text parameter of this procedure

10.2.1.2.2 The sub-modules for receiving messages - the server application

The server application in this project uses an IPDaemon ActiveX control. Creating a server means that the server application will "listen" on a designated port. When the client makes a connection request, the server can then accept the request and thereby complete the connection. Once the connection is complete, the client and server can freely communicate with each other. Hence, if a server application is being created, a port (**LocalPort** property) on which to listen must be set (in the example given in Table 10-2 this is the port 3000), and the **Listening** method invoked.

[1] A connection must be established first because the TCP protocol used by IPPort control is a connection-based protocol.

Table 10-2. A rudimentary set of instructions in the server application

Action	*Instructions*	*Description*
Start listening	***IPDaemon1.WinsockLoaded = True***	Load and initialise the Winsock library. The Winsock library must be loaded before IPDaemon starts listening.
	IPDaemon1.LocalPort = 3000	Set the TCP port in the local host (server computer) where IPDaemon resides. The LocalPort must be set before IPDaemon starts listening.
	IPDaemon1.Listening = True	Start to listen. An action property.

Create a form for each new connection.	Dim ConArray() As New frmConnection
	Dim Index As Variant
	
	Private Sub IPDaemon1_Connected(ConnectionID As Integer, StatusCode As Integer, Description As String)
	'Determine the index of an array element
	If IsEmpty(Index) Then
	Index = 0
	Else
	Index = UBound(gConnArray) + 1
	End If
	'Add a new form into the array
	ReDim Preserve ConArray(Index)
	Set ConArray(Index) = New frmConnection
	ConArray(Index)!lblConID = Str(ConnectionID)
	End Sub
		Since up to 1000 clients can be simultaneously connected to the server, each connection has to have a separate container for receiving messages. In this example a separate form (frmConnection) for each connection is created and it is stored in a dynamic array.
		
		Procedure:
		IPDaemon1_Connected is fired immediately after a connection completes.

Sending a message.	If IPDaemon1.Connected(ConID) Then
	IPDaemon1.DataToSend(ConID) = "Test Msg"
	Else
	MsgBox "Connection " & ConID & "closed."
	End If
		If the established connection to a remote host is still valid then send a test message.

Receiving a message.	Private Sub IPDaemon1_DataIn (ConnectionID As Integer, Text As String, EOL As Integer)
	For i = LBound(ConArray) To UBound(ConArray)
	If ConArray(i)!lblConID = ConnectionID Then
	ConArray(i)!txtMsgReceived = Text
	Exit For
	End If
	Next I
	End Sub
		Procedure:
		IPDaemon1_DataIn is initiated when data (complete lines) arrives.

When the client computer establishes a connection, the IPDaemon1_Connected event will occur. Each connection is maintained in a separate container[1] (in an example in Table 10-2 this is the form frmConnection). Once a connection has been made, either computer can send and receive

[1] The container controls are ActiveX controls that visually contain other controls. A "Form" is a typical example of a container object in VB.

data. To send data, the ***DataToSend(ConID)*** method needs to be invoked. Whenever data is received, the ***DataIn*** event occurs. Data are contained in the ***Text*** parameter of this event.

A newer version of VB, version 5.0, comes with the **Winsock** control that provides easy access to both TCP and UDP network services. This control has a similar set of methods, properties, and events to those that are found in IPPort and IPDaemon controls and can be used instead of them.

10.2.1.3 The decision-making module

The decision-making module is the "brain" of the workstation agent. It is composed of several sub-modules (refer to Figure 10-2) and controls almost all activities of the agent and the information flow.

As the name implies, the ***sub-module for generating messages*** is in charge of creating messages to be sent to other agents (at the moment, four types of messages are defined: a task announcement, a bid, an award and an acknowledgment message) when certain predefined criteria are fulfilled. For example, an event for starting the first negotiation cycle (and therefore for generating a task announcement message) will be triggered when a work-part is loaded on the machine. At that time, the sub-module needs to obtain all the data that is required to create a message. To do this, the module needs to gather data from the local database (through the data manipulation sub module) and the PRT files (with the help of the monitoring workstation activities module). Examples of the data include ID number of the work-part, ID number of the workstation that is capable of executing the next operation, the IP address of that workstation, etc.). When the process of message creation is complete, the message is put into the buffer for sending messages. From that point onwards, the communication module takes over to transfer the message to the destination workstation agent.

The ***sub module for interpreting messages*** has the task of interpreting messages that are received from other agents. The sub module constantly monitors the buffer for messages to be interpreted and if a message is identified, it notifies immediately the sub module for control of the negotiation process, which in turn either passes control to the sub module for generating messages or indicates (in the case that the winner workstation has been determined) that the negotiation process has been accomplished.

The ***sub module for definition of IP addresses*** provides the means for adding, modifying and deleting IP addresses. To send a message a workstation agent needs to know the address of the destination agent. Therefore, the sub module for generating messages requires a list of addresses of all known agents in the system. If a new agent is added into the system, its address has to be added to the list of addresses for each agent. Similarly, if an agent has been removed from the system, it is advantageous to remove the address from all the agents. This is not essential, since if the agent no longer exists it will not respond to task announcements. However, the address of the agent should be removed to help keep the system clean and to avoid too many unnecessary messages being sent on the network.

The ***data manipulation sub module*** contains procedures for adding, deleting and editing data located in agent's local database.

The ***sub-module "Control of negotiation processes"*** is responsible for initiation, control and closing negotiation processes. For example, it keeps track of the progress of each negotiation process (which messages are created and sent, which are received, and which messages still need to be sent) and the number of negotiation cycles performed. Currently, two negotiation cycles (attempts to determine the winning workstation for the next operation) are allowed before the agent announces that the winning workstation is not found. For each negotiation process a

separate container (a VB form) is created that facilitates simultaneous reception of messages from a number of different workstation agents.

The ***monitoring workstation sub module*** handles the monitoring requirements related to the workstation's operational activities. The sub module comprises a set of predefined procedures in order to respond to both: passive and active monitoring requirements. The request for *passive monitoring* comes from the system's management function (emergency/management control agent) that wants to know, for example, some specific information about the workstation's current status, such as the current status of a work-part processing, the capacity of a machine (e.g. in the last couple of hours), what work-parts and how many are in the input buffer, etc. *The active monitoring* is related to alarms. For example, if the processing of a work-part cannot be accomplished on time (and therefore delays in the delivery dates are expected), or if some of a workstation's components becomes non operational, an alarm event needs to be raised to notify system's authorities about such occurrences. As a source of information the sub module uses data from local database and workstation control unit. This sub module was not realised in the current version of the application.

The ***process control module*** is the central place for processing information. This module coordinates work of other sub modules and it is in charge of obtaining, preparing and supplying information as required.

10.2.1.4 The workstation control module

The nature of a workstation control module depends on the level of automation of the equipment that constitutes a workstation. In a fully automated system, a control module might have a few sub-modules such as: a machine sub-module, robot sub-module, input and output buffer sub-modules, and input (loading) and output (unloading) transport ports sub-modules. Some of these sub modules may not be required in all workstation agents. For example, a workstation that has no CNC machine and a robot (such as an input system workstation) the workstation's control module would not have the corresponding VMDs.

If the workstation is manually operated this module has another role. Since the control functions are executed by an operator, this module is used as an interface to the rest of the heterarchical shop floor computer control system. Using this module the operator updates information regarding work-parts and the current state of the workstation equipment. For example, the operator would trigger a signal that indicates that the work-part had arrived at an input transport port of the workstation. The operator would identify the work-part ID, record it and keep track of the work-part through the workstation by updating the work-part's location (input buffer, machine, output buffer). The operator would trigger a signal to indicate that the work-part is on an output port waiting for transport, and finally, the operator would trigger a signal that indicates the work-part has left the workstation. In this way a workstation agent would be aware of the status of the process activities and would be able to generate the proper responses, that is, bids on declared task announcements (see Section 6.2.3.1 "The Contract-Net Protocol").

At this stage of the application development, as already noted, the control module has been substituted with software that completely simulates its operations. The simulation program is event driven and this makes it possible for a user to trigger some events manually. Therefore, during simulation, events can be activated manually by the user, or automatically after predefined times have elapsed. The activities with predefined times may include part machining, or the handling of a part from one location to another by a robot. The interface that provides the interaction between a user and the program for simulation is shown in Figure 10-6. The same interface is used for monitoring activities inside the workstation model during the simulation process. A similar interface could be used in a real manufacturing system in which the

workstations are under human control. By switching any of the red (off) buttons (shown in Figure 10-6) on or green (on) buttons off an appropriate signal could be generated as the workstation proceeds through its activities.

The module for discrete event simulation of the workstation activities records all the data of interest (for example, the time when the work-part has entered a workstation, the start time when a work-part's machining has begun, or the time when a work-part has left a workstation) in the local workstation's "work-part database". This data can be used later for analyses of the manufacturing processes and the system as a whole.

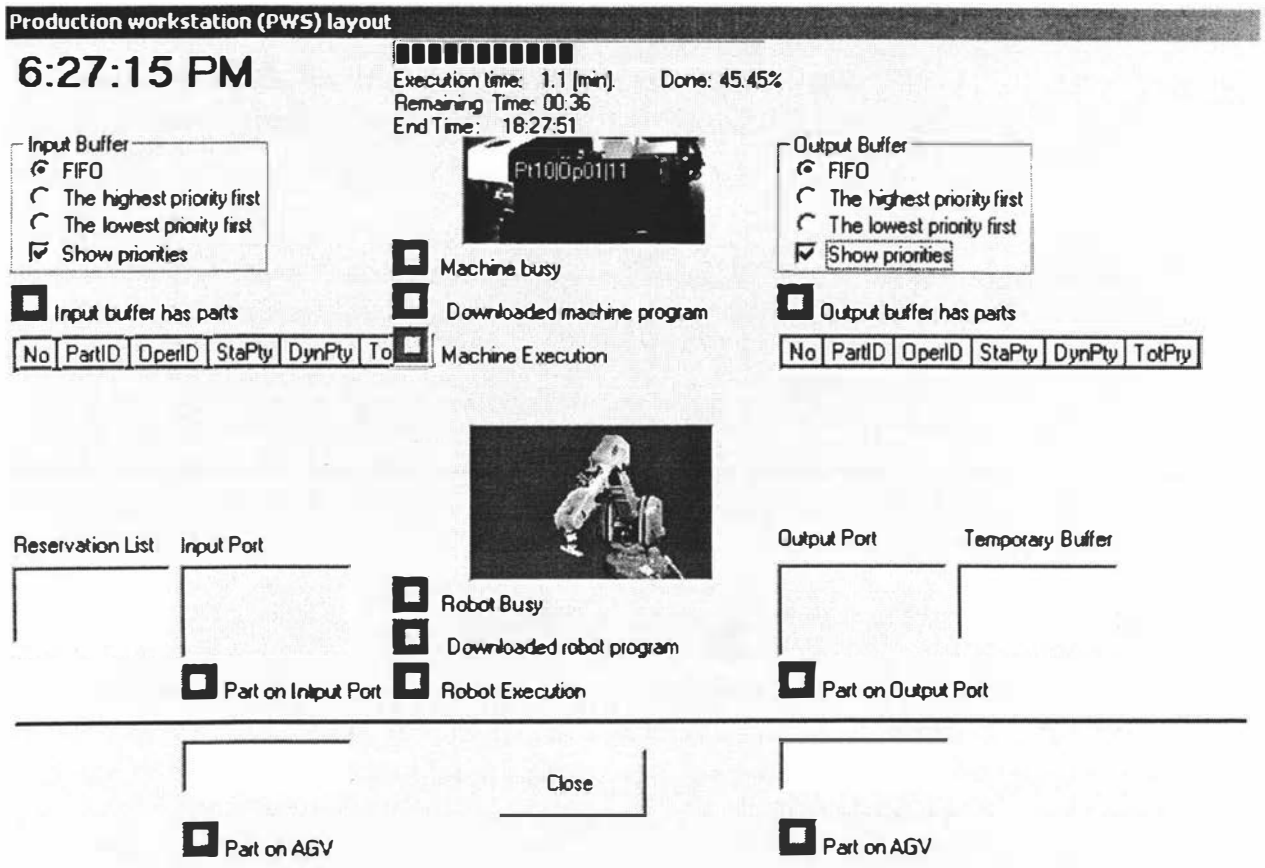


Figure 10-6. Simulation of workstation activities

10.2.2 Co-operation and communication between workstation agents - the implemented negotiation mechanism

To produce a work-part, workstation agents have to cooperate with one another. This cooperation is required, either because a workstation agent is not able to accomplish a work-part itself (a predefined set of machining operations need to be performed on several machines - workstations), or because other agents can successfully accomplish a set of processing activities more efficiently (for instance a given task will be done faster on the other workstation). To be able to cooperate, agents are supposed to have complete knowledge of the states of the machine tools and other equipment that constitute their workstations.

For heterarchical shop floor control the efficient allocation of work-parts among workstations is one of the primary interests. The work-part allocation problem is solved through the process of ***negotiation*** (more details on negotiation topic are provided in Section 6.2 "Coordination among

agents"). In this project the negotiation mechanism is designed with a focus on the immediate task on which agreement between agents is sought. Within such a limited focus, a simple question-answer or the Contract-Net Protocol is sufficient. Therefore, communication among workstations agents is based on a relatively simple, bounded protocol that represents a simplified version of the Contract-Net Protocol (refer to Section 6.2.3.1 "The Contract-Net Protocol"). At this stage a proposed protocol represents a satisfactory solution, though, it can be extended later to include any new requirements that may occur.

When selecting (developing) an appropriate protocol there is often a trade-off between process complexity and protocol complexity. More flexible protocols may lead to faster convergence than simpler ones (or even achieve convergence in cases that are intractable for simple protocols). The implementer must assess the degree of this interaction in deciding the impact of agent simplicity and speed of convergence in a given application. Similarly, if the problem complexity can be bounded, a simpler protocol may achieve satisfactory results at lower cost than if one attempts to construct a general-purpose agent that is over-engineered for the problem at hand.

Since good engineering strives for simplicity, in developing of a multi-agent heterarchical shop floor scheduling and control system, the simplest system that does the job has been preferred and adopted. Simplicity has many benefits, including shorter development time, a system that is easier to understand and maintain, and reduced platform costs. The negotiation process is distinguished by the following characteristics:

- Negotiation supports *static goals* (there is no need for renegotiation or meta-level negotiation, as is the case when the goals are changeable over the time);
- Negotiation is *cooperative* instead of competitive (there is no need to guard against malicious threats);
- The domain over which the negotiation function is defined is *discrete* not continuous (there is no need for evaluation of the current state of the negotiation and to plan the next step);
- Goals that need to be satisfied are *one-dimensional,* not multi-dimensional (there is no need to satisfy multiple objectives concurrently); and
- Negotiation is about *present actions* instead of future actions (there is no need for time management to plan those future actions).

10.2.2.1 The analogy between Task Sharing in distributed problem solving, and Task Allocation in manufacturing

In the discrete part manufacturing a production process of any work-part consists of a set of operations that are subsequently executed on a variety of production equipment. The similarities between a problem of sharing tasks in distributed problem solving (refer to Section 6.2.3.1 "The Contract-Net Protocol") and a task allocation problem in a manufacturing environment can be thought of a work-part production process in the following way. Since the whole process of producing a work-part is decomposed into smaller sub-processes called production operations, we can consider that a sequence of technological operations that a work-part needs to follow on its way through the transformation from raw material or semi-product till to final product actually represents a sequence of production tasks. Usually each of these tasks needs to be distributed among the group of production entities (workstations) capable of part processing. The exceptions are those tasks that can be done on only one workstation due to highly specific processing requirements. Therefore, in the discrete type of manufacturing we are talking about the ***task allocation process*** as a process that is very similar to task sharing in distributed problem

solving. The entire production process of a work-part is first decomposed into sub-process tasks, and then these tasks are distributed amongst a group of production entities (workstations) capable of processing them.

The key issue to be resolved in task allocation is how production tasks are to be distributed among the production entities. Workstations with tasks to be executed need to find the most appropriate workstations to execute those tasks. This ***task allocation problem*** is again quite similar to the connection problem in distributed problem solving. Solving the task allocation problem has two aspects: 1) a work-part allocation and 2) focus.

- Effective ***work-part allocation*** is achieved by balancing the production load among the workstations.
- ***Focus*** is achieved through a careful and thorough process of process planning, (determining in advance (off-line) a set of capable workstations that can execute a task) and by effective selection of workstations for execution of tasks later in real time (on-line) while the work-part is produced.

While communicating, workstation agents actually negotiate whether they will or won't execute a given task. In the Contract-Net Protocol, (as discussed in Section 6.2.3.1.2 "The Contract-Net Protocol –details") a process of negotiation is based on the principles of auctioning. The workstation agent that initiates an auction activity is called a ***source workstation agent*** and workstation agents that participate in that activity on the other side are called a ***destination workstation agent***. Any workstation agent can take on either role dynamically during the course of a negotiation. Typically, a workstation agent will take on both roles, often simultaneously for different negotiation processes.

10.2.2.2 The structure of messages

The elementary set of messages on which the communication among workstation agents is based consists of only four messages: a task announcement, a bid, an award, and an acknowledgement message. The content of the above messages is described in the following subsections. As it will be seen, all four messages in a test-bed network application have the same structure. Any arbitrary number of new messages can be easily defined and integrated into existing modules for creating and interpreting messages, making the inter-agent communication within the system much richer.

10.2.2.2.1 A Task Announcement message

After certain conditions are fulfilled (for example a workstation has completed the processing of a work-part and the work-part is located on a workstation's output transport port), acting as a source workstation, the workstation creates a task announcement for the next production task. The task announcement message is addressed and sent to those workstations, denoted in the Processing Route Table (PRT) for this task (refer to Section 11.2.1.2 "Getting the Processing Route Table (PRT)"), that are capable of executing the announced tasks (the message is not broadcast but addressed directly to a few alternative workstations only). An example of the task announcement message is given in Table 10-3. The message currently consists of nine lines (slots) where each line has two parts: a prompt and the line content. The message is organised around two parts: a compulsory part of a message (including a header and footer) and a part for which the content depends of the type of message. The latter part can have an arbitrary number of lines in which the content and the number of lines might vary depending upon the message type (indicated by a couple of empty lines in Table 10-3). There are no limitations as long as the modules for creating and interpreting messages are harmonised. The message lines are mostly self-explanatory though further explanation might need to be given to line number 4 and the last

line (denoted as a line "n"). The value in line four refers to the "number of allowed attempts" to send a message after which the message will be deleted from a buffer of messages to be sent. Line nine is a message terminator. When a destination agent receives this terminator it means that the message is completely transmitted and the open connection between workstation agents can be terminated.

Table 10-3. An example of a task announcement message

Message parts	***Line number***	***Prompt***	***Content***
Compulsory header	1	MsgType	Task Announcement
	2	MsgToIPAdd	192.168.10.1
	3	MsgFromIPAdd	192.168.10.4
	4	MsgTimeToLive	10
	5	MsgPriority	4
Arbitrary body (type dependant)	6	PartID	Part0010
	7	TaskID	Task03
	8	PartPriority	5
Compulsory footer	n	TheEnd	

Note that the PartID and the TaskID uniquely determine the negotiation process to which a message belongs. This is important in the case that the workstation agent conducts a number of simultaneous negotiation processes for several work-parts. With uniquely determined messages each agent can maintain internally the status of any negotiation processes that is currently running. In this way communication resources (for sending messages) are engaged only when there is a need, instead of keeping them operational all the time during the course of negotiation. In other words, since each message is treated separately and independently, communication resources (connection channels) are engaged only when there are messages to be sent.

The current implementation of the test-bed application does not support allocation of other resources such as, for example, shared cutting tools and fixtures. Hence, PartID is sufficient for determining which message belongs to which negotiation process. Alternatively, a new line could be introduced as a part of the compulsory message header to uniquely determine the negotiation process to which message belongs to, for example NegotiationProcessID.

10.2.2.2.2 A Bid message

Upon receiving a task announcement message each of the destination workstations generates its own bid. Bids are generated according to the appropriate criteria by calculating the values requested in the message. Measures of interest might be time (for example earliest finishing time), currency (for example cost of a task execution), or any other performance measure. In the current implementation, since the goal of the system is to minimise processing time for work-parts, the earliest finishing time (EFT) is adopted as a measure of performance. Bidders send their bids back to the source workstation, which ranks them when the predefined time for the bid evaluation elapses. The workstation that has sent the most satisfactory bid (the smallest value of EFT) is declared as the winner. An example of a bid message is depicted in Table 10-4. Content is almost identical with the task announcement message shown in Table 10-3, except an extra line (BidTime) is added in the body of the message (time represents an estimated finishing time for a task "Task03" of a work-part "Part0010" expressed in minutes). Also note that the values

of the IP addresses have swapped places (value for MsgToIPAdd in a bid message is the same as the value for MsgFromIPAdd in a task announcement message and vice versa).

Table 10-4. An example of a bid message

Message parts	*Line number*	*Prompt*	*Content*
Compulsory header	1	MsgType	Bid
	2	MsgToIPAdd	192.168.10.4
	3	MsgFromIPAdd	192.168.10.1
	4	MsgTimeToLive	10
	5	MsgPriority	3
Arbitrary body (type dependant)	6	PartID	Part0010
	7	PartTaskID	Task03
	8	PartPriority	5
	9	BidTime	138.2
Compulsory footer	N	TheEnd	

10.2.2.2.3 An Award message

If a winner workstation is determined, the source workstation that has initiated a negotiation process now sends an award message to the winner workstation. If the winner is not determined, a negotiation process for that work-part will be activated again later, as described in Section 11.2.2.3 "Selection of the winning bid and the generation of an award message". The structure of the message is the same as for a task announcement.

10.2.2.2.4 An Acknowledgement message

The winner acknowledges the award by sending an acknowledgement message back to the negotiation initiator and reserves resources for the task. Upon receiving the acknowledgement message a source workstation considers that the negotiation process has been successfully accomplished and starts the procedure for delivering a work-part to the destination workstation. The priority of the message is set at 1 so that this message is sent back to the source workstation as soon as possible. The structure of the message is the same as for a task announcement and an award message.

10.3 Scheduling in the model of a distributed heterarchical shop floor control system

In the model of a shop floor control system and in the test-bed application developed in this project, a distributed heterarchical multi-agent approach to scheduling is used. This section describes the methods for scheduling jobs that are applied in the model, that is, it discusses in more detail the scheduling of jobs among workstation agents as well as inside workstation agents.

The distributed heterarchical multi-agent approach is characterised by a high degree of local autonomy, where responsibility and authority are fully distributed among the entities in the system. Each entity – workstation agent - in the system is self-contained and independent, and does not require any kind of global information to operate (except the IP addresses of other

agents located in the system, workstation agents do not keep any global data. Data that is contained in the Processing Route Tables is moved with the work-part and as such is not possession of any workstation agent, however, this data is crucial for operation of the system) Workstation agents do not even need to know about the existence of each other in the system, although in this study, it has been decided that each workstation agent maintains a minimum amount of global data –namely the IP addresses of all workstations in the system. Workstations collaborate by exchanging information (messages) and materials (work-parts, shared tools, fixtures and jigs), and independently make arrangements for moving work-parts through the system. A workstation negotiates to accept a work-part for which it has the production capability and the necessary information to perform the required processing operations on the work-part. When the work-part is finished it negotiates the dispatching of the work-part to the next workstation.

10.3.1 Global scheduling - scheduling among workstation agents

In this study the scheduling of jobs among workstation agents is performed in a reactive manner (see Section 6.3.5. "Scheduling in distributed multi-agent heterarchical control systems"). In this approach workstations generate local schedules using local heuristics and relying on local information. The system has no global co-ordinator to look after the global performance of the system. Instead global performance emerges from co-operation among distributed entities in the system. Depending on the algorithm applied for scheduling work-parts inside a workstation, a global schedule may be affected in such a way that a work-part may take different routes on its flow through the system (different workstations will be selected for work-part processing). Therefore, the flow time (how long it will take a work-part to be produced) may be different depending on the local scheduling method used. For example, if the first-input/first-output (FIFO) rule is applied on all workstations for sequencing jobs on machines, then a work-part would be globally scheduled through the system in one way, which would be different if the work-parts were sequenced on the machines by applying the highest priority rule. (Further it should be noted that, in the approach developed here, each workstation may apply different scheduling rules allowing more flexibility in adapting the part selection for processing to the needs or condition of the particular workstation). Currently the application supports three predefined local scheduling criteria for selecting work-parts from the input and output buffers of the workstations (FIFO, the highest priority first, and the lowest priority first). Criteria can be independently chosen for input and output buffers by selecting an appropriate radio button on the user interface (refer to Figure 10-6).

Reactive scheduling also means that the forecasting of events in the system has been reduced as much as possible. Decisions for selecting the most appropriate workstations to perform the next operation are postponed to the latest possible extent, so that decisions are made by taking into account only the information that reflects the current state of the workstations and not the state that is expected to occur some time in the future. In a purely reactive scheduling method, a local schedule does not consider work-parts that are expected to arrive on a workstation other than the work-parts that are already directed towards the workstation and are currently being transported from the workstations that have finished previous operations. Conforming to the principle of latest commitment, an auctioning procedure (a negotiation process in which a decision for selecting the next workstation is made) is triggered when a work-part reaches the output transport port, that is, when the current operation on the work-part is completely finished. At that time all workstations that have received the task announcement will generate bids according to their current states. The current state of a workstation is determined by the work-parts already allocated to the workstation (including work-parts being transported towards the workstation and

work-parts inside the input buffer), a work-part that is being processed on the machine, and the current set up of the machines in the workstation.

The current implementation of a test-bed application is extended from its original form; so that at the moment, it allows two negotiation cycles to be conducted in an attempt to determine the winning workstation which will execute the next manufacturing task. The first negotiation cycle is triggered when the work-part reaches a machine tool and if unsuccessful the second cycle is triggered when the work-part reaches the output transport port. Figure 10-7 illustrates how a negotiation process is conducted in the current realisation of a workstation agent.

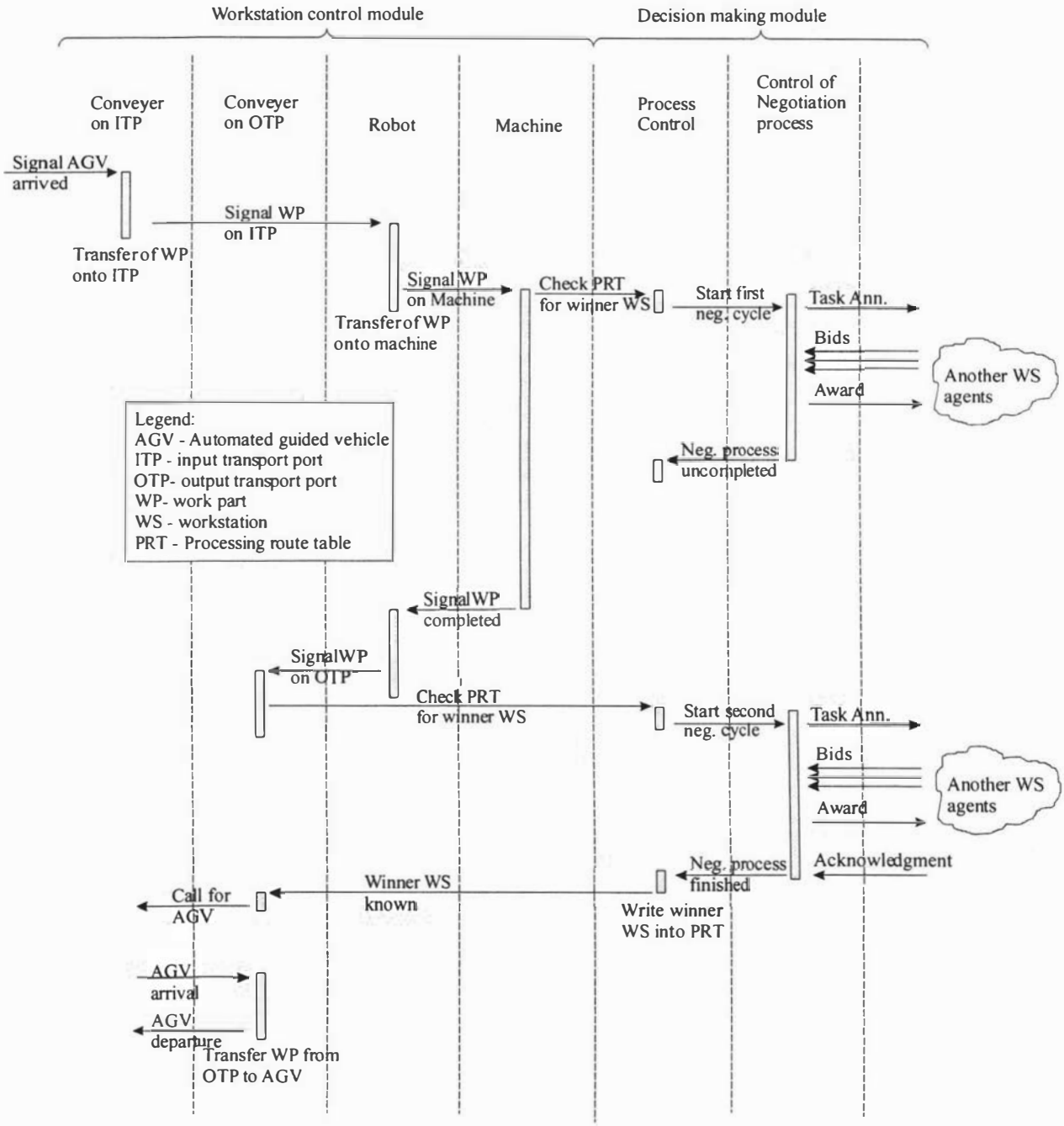


Figure 10-7. Process of determining a "winner workstation" for conducting the next production task

The figure shows that the first negotiation cycle is activated at the time when a signal that a work-part (WP) is on the machine is obtained. At that time, the process control module (see Figure 10-2) will check if the winner workstation (WS) for the next manufacturing task is known. The module opens the Processing Route Table (PRT) file and reads a value of the field which is determined as the intersection between the first column (going from the left hand side of the table and progressing to the right) which has status "pending" and the row which is denoted with "WS- winner" tag. If the field is empty (which is always the case when the first negotiation cycle is about to start) the first negotiation cycle is started. In the example depicted in the figure, the first negotiation process was unsuccessful so the second negotiation process is started at the time when a signal that the work-part reached output transport port (OTP) is obtained.

10.3.2 Local scheduling - scheduling methods for each individual production workstation agent

Workstation agents can use any of the traditional heuristics (or optimisation) techniques to solve the local workstation's scheduling problems – sequencing jobs on workstations. For each individual production workstation agent, job scheduling is actually a single machine queuing problem and currently, the dispatching rules are used as the queuing discipline (the FIFO rule and the rule which is based on priority indexes as discussed further in this section). Therefore, the local scheduler deals with very restricted one-machine scheduling problems with a small number of jobs (for example, 2-5 jobs) that need to be scheduled on the machine.

Work-in-process inventories in the factory are severely limited. By limiting the number of work-parts in the input buffer that are waiting to be processed on the machine, the complexity of the scheduling problem is significantly reduced. If the limit for the number of work-parts allowed in the input buffer is reached (for example 5 work-parts), then the workstation will not participate in any negotiation process that is initiated for a work-part intended to be allocated to this workstation.

In the current implementation of the workstation agents, jobs are sequenced according to their priorities. Jobs get priority indexes taking into account several aspects, so that the priority index is composed of three parts:

- ***A fixed part.*** This static part of the priority index allows the definition of the "urgency" of work-parts (jobs). More urgent work-parts will have higher value for this part of the priority index than the less urgent. A value of this fixed portion can be determined according to the time difference between the time the work-part is required (due date time) and the time that the part is expected to be completed (estimated time calculated on the basis of the total work content which needs to be done on the work-part).
- ***A lateness part.*** This is part of the priority index whose value rises each time that a work-part is skipped by other (more urgent) work-parts that arrive at the workstation. The lateness portion allows a workstation to be cleared of late jobs. It will keep rising as long as the other work-parts with higher priorities continue to arrive at the workstation. At one stage, the lateness portion of priority index will become so high that the overall priority index of the "skipped work-part" will be higher than any other priority index of new arriving work-parts. Eventually, with the highest priority this work-part will be processed next.
- ***A permission part.*** This part of priority index has a role of a flag. If it is explicitly set (for example by the staff of manufacturing facility who is in charge for operation and control of that facility), work-part will be not considered under any circumstances, that is,

it will be treated, as if it is not present in the workstation. This portion provides possibility to explicitly control processing of the work-parts on each workstation. If for any reason a work-part should not be machined on the current workstation, the value "false" in a "considering portion" will prevent the part from being considered for processing on a machine. For example, if during a negotiation process the lateness portion of some part (located in the input buffer) would be sufficiently high to jeopardise the immediate machining of a very urgent part, then the machining of this late part can be suppressed on this particular occasion. In this way, the "ultimately urgent part" would take the first place in the input buffer queue, which otherwise would not be possible (by using only the automatic priority setting system without human intervention). This permission part of a priority index has not been implemented in the current software implementation.

According to the priorities of work-parts that are in the workstation input buffer and work-parts that are scheduled to arrive at the workstation, the local scheduler generates a schedule called a "workstation schedule plan". In the current implementation this plan is represented simply by a queue of work-parts waiting to be processed on the machine that are sequenced according to the work-parts' priority indexes, though in the future, as a part of possible application improvements, this schedule plan can be presented in a more sophisticated form, for example, as a Gantt chart. (It should be noted that as there are only a relatively small number of tasks to be scheduled at the workstation when compared with a complete shop floor with many workstations, the schedule can be calculated in real time and can be updated sufficiently rapidly to maintain its currency with the present status of the workstation.)

The workstation schedule plan is generated every time when a new work-part is allocated to the workstation (that is, when a reservation - booking - for the arrival of the new work-part is made). In the implemented approach, the test-bed application uses (by default) a sequencing rule that puts the work-part with the highest priority index at the first place in the waiting queue. For a given workstation schedule plan, the work-part that is to be processed next (the work-part at the first place in the queue) is always known. In this way the virtual workstation controller can take control over activities for loading of work-parts from the input buffer, or the input transport port, onto a machine as soon as previous work-part is finished and the machine becomes available.

The local scheduler always deals with the jobs that are currently located in the workstation's input buffer. However, during the negotiation process when the new bids are to be created, the workstation agent also takes into account all the jobs that are allocated to the workstation and for which the arrival at the workstation has already been arranged in a previously finished process of negotiation. When such a work-part finally arrives at the workstation the local scheduler will be activated and the new sequence of work-parts waiting for processing will be created. Note that the expected time for work-part completion on a given workstation, which is initially determined during negotiation at the time when bids were created, may be exceeded (prolonged) for some work-parts. However, because the lateness portion of a work-part's priority index will raise the priority of these parts if they are overly delayed, these local disturbances should not cause significant ultimate delays in the work-part production.

As noted, the use of priority indexes can be substituted with other sequencing rules. In this context, in addition to data that is available from the workstation control module, local schedulers can use data contained in a Processing Route Table (PRT) files. If required, the PRT files can be easily extended to include additional data, such as, the total estimated processing times, estimated remaining processing times, due times, and so forth.

To prove the feasibility of the proposed heterarchical shop floor control system and to demonstrate its operation, an experimental local area network was created. The next chapter discusses how this network was established.

Chapter 11. How the heterarchical shop floor control system works? – The operation of the manufacturing system model

The proposed model of a heterarchical multi agent shop floor control system and the model of the workstation agent are discussed in Chapter 10. This chapter describes the operation of the control software developed by the author working over a small peer-to-peer LAN, which was assembled and used initially as a platform for development and testing purposes and later for demonstration of the software operation. Using an example of the production of an arbitrary work-part, the sequence of the main activities performed by the application for modelling and simulation of heterarchical shop floor control systems is described briefly. This example shows how the application can be used as a tool for solving resource allocation problems in automated job shop manufacturing systems.

11.1 The assembly of the demonstration system and the creation of an experimental Local Area Network (LAN)

To create the demonstration multi-agent heterarchical scheduling system the author assembled ten 486 IBM compatible computers and one Pentium I computer. Most of the 486 computers had Intel processors that ran at 50-60Mhz, with from 8 to 32Mb of RAM, hard disks with capacities of 200-500Mb and video systems consisting of VGA cards (800x600) and 14" monitors. These computers were connected into a peer-to-peer Local Area Network.

When building up a network system, it is important that each computer is properly configured. (Computer configuration is the process of setting up the computer's hardware devices and assigning resources to them so they work together without problems.) Each computer (once it was assembled and Windows 95 operating system was installed) was checked for any resource conflicts among individual computer devices. A properly configured system has no resource conflicts, and runs smoothly. An improperly configured system leads to strange errors and problems, making the whole system unstable and unreliable. (This problem was difficult to solve using the Windows 95 operating system. However the newer operating systems such as Windows 2000 and Windows XP have made the setting up of a network and the configuring of the individual computers much easier.)

Figure 11-1 illustrates the physical layout of computers in the LAN used for this project. The computers in the experimental network were connected using a thin Ethernet (RG58) coaxial cable. To synchronise the system time (computer's internal clock) of a local computer with the system time on the "Main" computer (see Figure 11-1) the Windows NET command with the option "TIME" was used (the syntax of a command is: NET TIME \\Main/SET /YES).

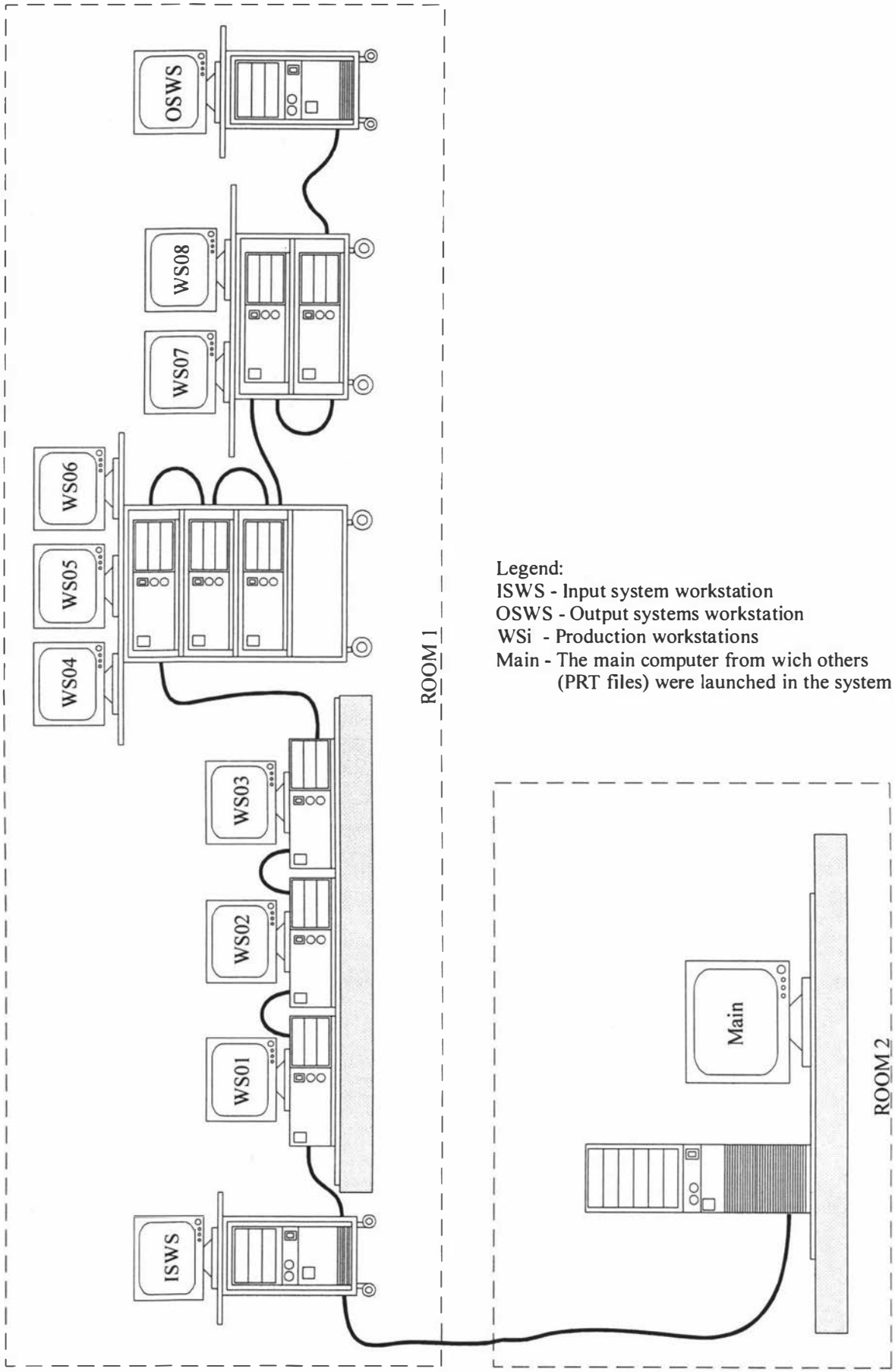


Figure 11-1. A physical layout of computers in the network

11.2 The operation of the demonstration system

The main goal of the applied scheduling algorithm is to ***minimise the processing time*** of the work-parts in the manufacturing system (note that other algorithms can be implemented, for example, to minimise production costs, to increase machine utilisation, etc.). As indicated previously, the manufacturing environment at which this system is targeted is characterised by having a large number of distinct part types produced in low volumes, approaching single part production. Part processing is performed on general purpose automated machine tools organised into workstations. Each workstation is mapped by a corresponding workstation agent – software at the workstation level. Agents are installed on the workstation computers and they co-operate with each other via a Local Area Network (LAN) by using a protocol, which is based on the Contract-Net Protocol. For ease of understanding, as noted, we will follow the sequence of activities on a simple example work-part travelling through the system. The example work-part is depicted in Figure 11-2, a corresponding set of processing operations for the work-part is given in Table 11-1 while Figure 11-3 and Figure 11-4 depict the process plans at the shop and workstation levels respectively.

The intention of this chapter is not to provide a detail program description, but rather to illustrate how the application worked. Occasionally, the illustration will be supported by a few screen shots that complement a given explanation. (A live, full scale program demonstration using 11 computers connected in a LAN was organised and shown to a few interested parties at the time of project completion). In addition to this chapter, further information about the program setup and its operation can be found on the accompanying CD. The CD contains video clips about

- Defining program's data (IP addresses, data contained in the workstation agent database, and data contained in Processing Route Tables);
- Setting program's parameters; and
- Video clips taken during the system operation in a "simple mode" (using only one computer[1]).

11.2.1 Pre-process activities - the system input workstation

A part, together with the corresponding shop level process plan, enters a system at the system input workstation that represents the system entry point (refer to Figure 8.6 "A manufacturing system model"). At this station the pre-process activities for the machining of the work-part are performed as follows:

- A work-part is fixed on the pallet (as discussed in Section 11.2.1.1 "Fixing a work-part on a pallet");
- A shop level process plan is transformed into a Processing Route Table (PRT). (See Section 11.2.1.2 "Getting the Processing Route Table (PRT)" for more details); and
- Then all workstation agents' local databases are checked for the work-part processing data and, if needed, the processing data is provided to them. Each workstation indicated in the PRT has to have this data prior to starting a work-part's fabrication process. (Refer to Section 11.2.1.3 "Supplying production workstations with processing data" for more details).

[1] At the time when 11 computers were running it would have been too difficult to record the appropriate screen shots for all computers in the network. However, it was considered that the recording on the CD demonstrates sufficiently the operation of the system.

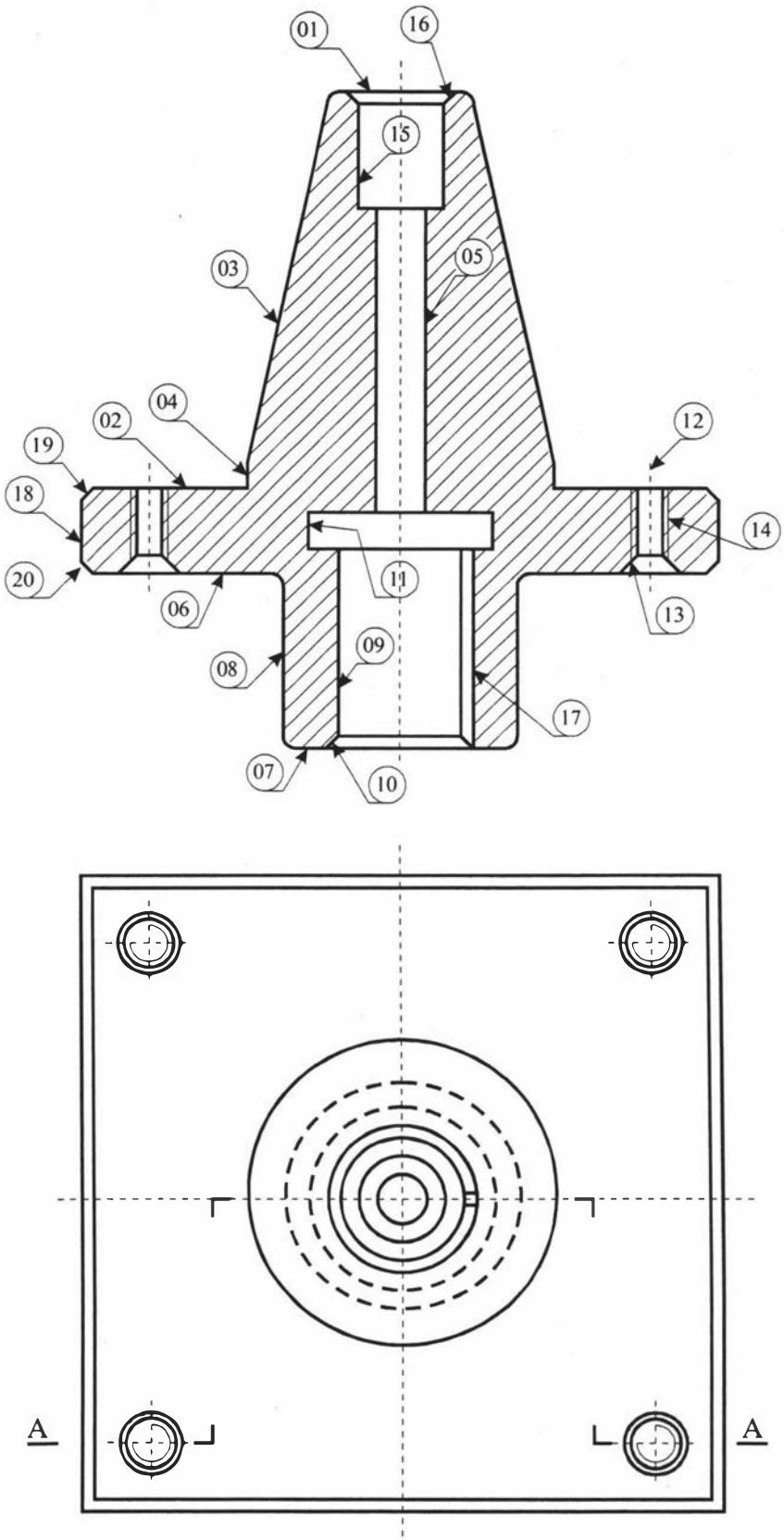


Figure 11-2. An example work-part Pt0010 – Hub

Table 11-1. Operation summary for the example work-part Pt0010

Task No	***Task Type***	***Oper. No.***	***Operation Type***	***Feature***	***Machine Type***	***Drawing Position***
Task01	Shape head-on surfaces, conical surface, and external and internal cylindrical surfaces	Op.01	Head-on turning	Head-on surfaces	Lathe	Pos.1, 2
		Op.02	Outer turning	Conical and external cylindrical surface		Pos.3, 4
		Op.03	Twist drilling	Central axial hole		Pos.5
		Op.04	Head-on turning	Head-on surfaces		Pos.6, 7
		Op.05	Outer turning	External cylindrical surface		Pos.8
		Op.06	Inner turning	Internal cylindrical surface		Pos.9
		Op.07	Inner turning	Counter sinking		Pos.10
		Op.08	Inner turning	Inner radial groove		Pos.11
Task02	Form threaded holes	Op.09	Twist drilling	Peripheral holes	Drill press	Pos.12
		Op.10	Counter sink	Peripheral holes		Pos.13
		Op.11	Threading	Peripheral holes		Pos.14
		Op.12	Reaming	Central axial hole		Pos.15
		Op.13	Counter sink	Central axial hole		Pos.16
Task03	Create groove	Op.14	Shaping	Axial groove	Machine Centre	Pos.17
Task04	Shape plain Surfaces	Op.15	Side milling	Outer squared surface	Mill	Pos.18
		Op.16	Chamfering	External squared edge		Pos.19
		Op.17	Chamfering	External squared edge		Pos.20
Task05	Finish supporting surfaces	Op.18	Cylindrical grinding	External cylindrical surface	Cylindrical grinder	Pos.8
		Op.19	Cylindrical grinding	Internal cylindrical surface		Pos.9
		Op.20	Conical grinding	Conical surface		Pos.3
Task06	Part examination	Op.21	Inspection	Part	Human operator	

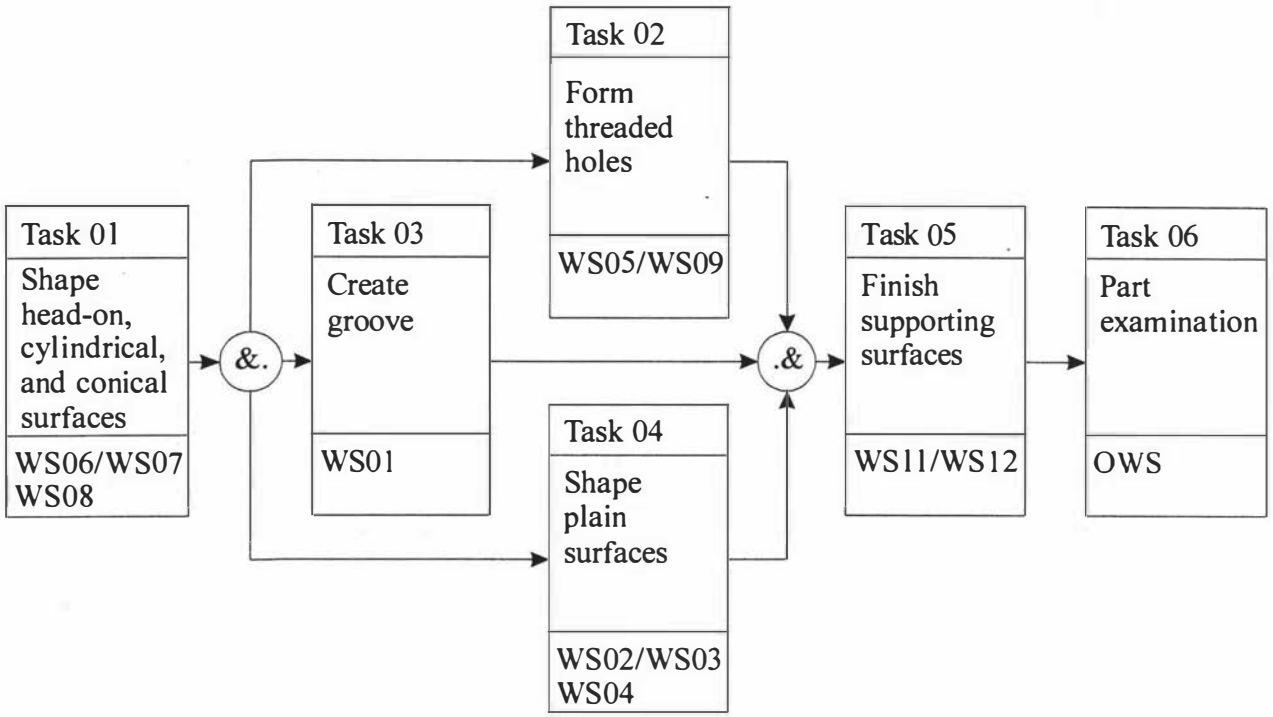


Figure 11-3. The process plan for the example work-part Pt0010 – A shop floor (task) level plan

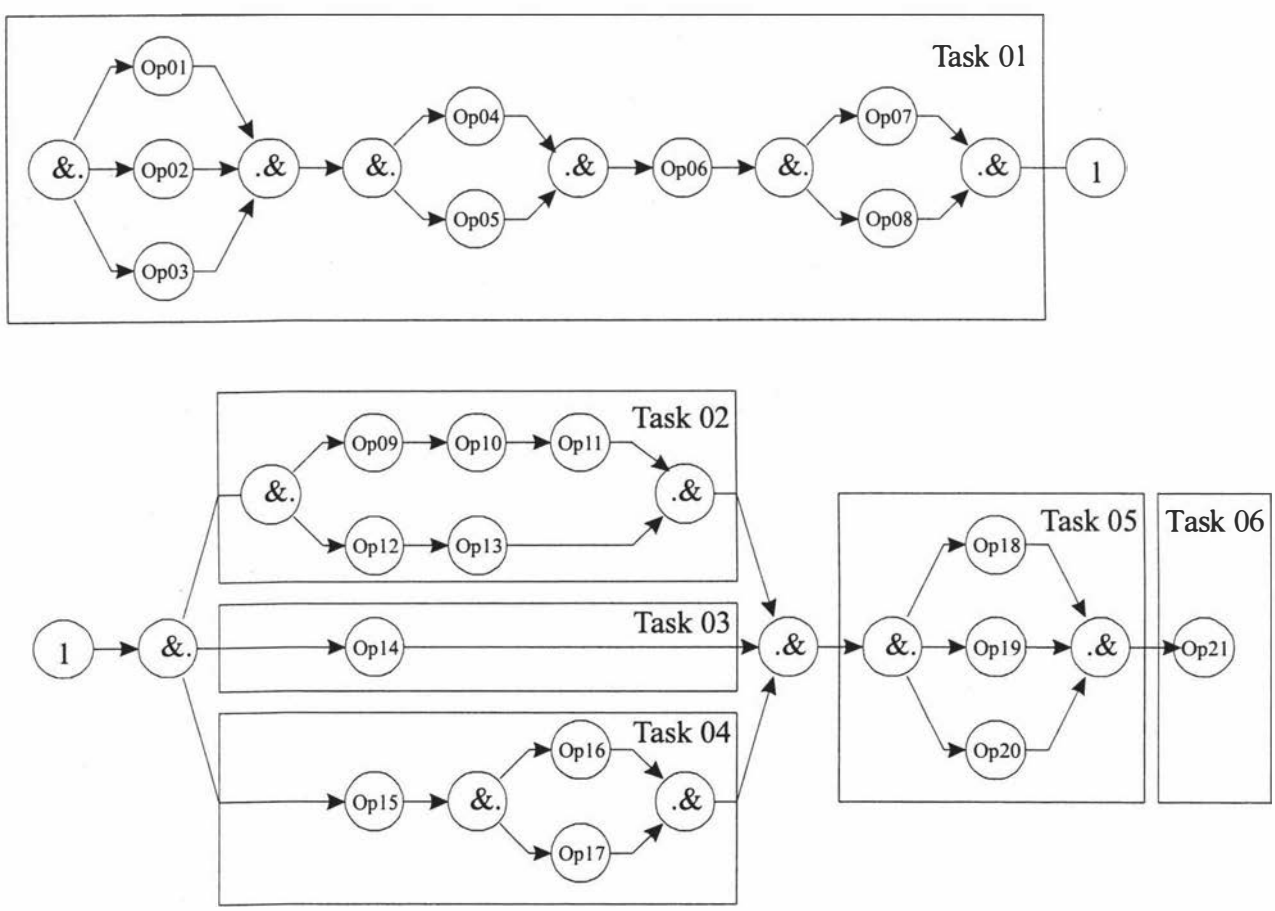


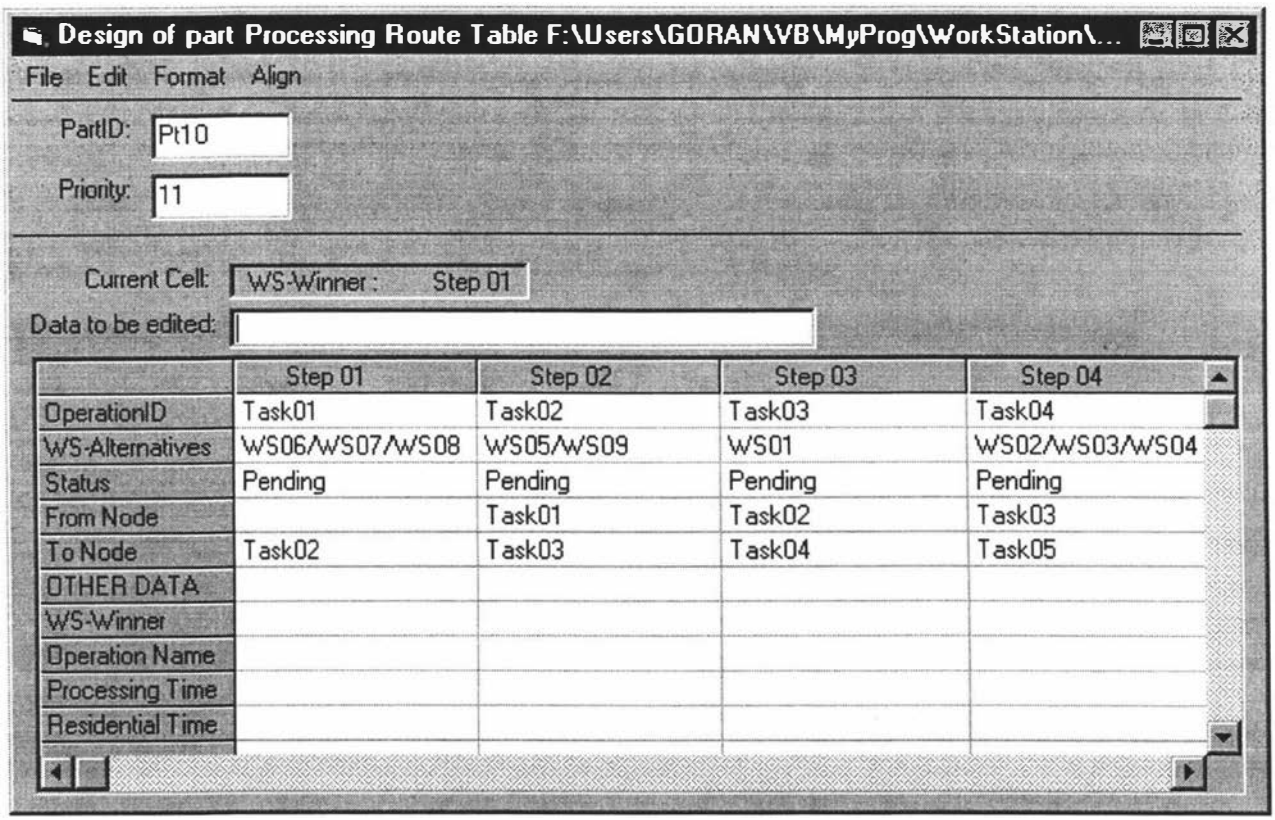
Figure 11-4. The process plan for the example work-part Pt0010 – A workstation (operation) level plan

11.2.1.1 Fixing a work-part on a pallet

Following a predefined set of instructions, and using fixtures and jigs, the work-part is fastened on the pallet when it arrives at the system input workstation. The work-part must be set in the proper position and orientation. The same palettes are used for machining and transport of work-parts between the workstations. If the work-part, during the manufacturing process, needs to be repositioned on the palette for performing certain processing tasks on some workstations, then, depending on the complexity of the repositioning operation there are two options. The work-part can be either sent to the system input workstation again, in the case when repositioning requires special jigs, fixtures, material handling equipment, or special operator skills (in practice this is happens very rarely), or the work-part can be repositioned at the workstation - in a case of less complex and demanding repositioning requirements.

11.2.1.2 Getting the Processing Route Table (PRT)

At the system input workstation, the shop level process plan for the work-part is transformed into a ***Processing Route Table (PRT).*** This transformation is normally done prior to the arrival of the work-part, off-line, during the process planning, in which case the PRT would be sent to the input system station as a file ready for use. However, in the absence of a Computer Aided Process Planning (CAPP) system, an application (the system input workstation agent) provides a user interface for conducting this transformation at the system input workstation. Figure 11-5 depicts an example of a PRT for the part Pt0010. This is the format that was developed for this project.



	Step 01	Step 02	Step 03	Step 04
OperationID	Task01	Task02	Task03	Task04
WS-Alternatives	WS06/WS07/WS08	WS05/WS09	WS01	WS02/WS03/WS04
Status	Pending	Pending	Pending	Pending
From Node		Task01	Task02	Task03
To Node	Task02	Task03	Task04	Task05
OTHER DATA				
WS-Winner				
Operation Name				
Processing Time				
Residential Time				

Figure 11-5. A processing route table (PRT) for the example work-part Pt0010

In the example, each column represents a task that can be performed on one or more workstations. Identification numbers of workstation that are capable of executing the tasks are entered in the *WS-Alternatives* field. According to the workstation IDs contained in that field, at

the time when a bidding procedure needs to be initiated, the module for generating messages will identify to which workstations a task announcement needs to be sent. The *status* field refers to the status of a task. It is initially filled with the value "pending" for all tasks in a PRT. Later, during the work-part processing, if a particular task is currently being executed on any of the workstations, the value of the status field will be changed into "current". The tasks that have already been accomplished will have a value "finished". From the data in this field it is possible to determine where the machining process of a work-part has reached and what the next task is, at any time. *From Node* and a *To Node* fields are introduced to describe shop level process plans which have "AND" and "OR" nodes (see Figure 11-3). Entries in these fields can be a task ID or a sequence number of an "OR" or "AND" node from the process plan. The remaining rows under *Other Data* field are self-explanatory and the list can be further extended as far as is needed.

The Processing Route Table (PRT) is the main repository of work-part information. It is transmitted from one workstation to the next one and follows a work-part through the system. A PRT can be considered as a precursor of a respective work-part. At the moment, when a work-part reaches an output transport port inside a workstation, a PRT will be sent to the workstation that has been selected for the next task, (the winner workstation that was determined during the negotiation process) announcing that the work-part is arriving. A PRT is transmitted from the current workstation to the next one on the communication network in the same way as the other messages are sent.

11.2.1.3 Supplying production workstations with processing data

All workstations whose identification numbers are included in the work-part's shop level process plan (that is the PRT) need to be supplied with the all relevant processing data before the work-part machining process, starts. Thus, each workstation will be able to participate in the negotiation (auctioning) process and will "know" how to handle a work-part through the workstation when the work-part arrives at its input workstation's transport port. For instance, a workstation needs to be provided with CNC and robot programs, a tool list, a list of additional jigs and fixtures (if required), estimated processing and set up times, and so forth. For that reason, as a part of pre-processing activities, the local databases of the workstations whose identification numbers are contained in the shop level process plan for a given work-part need to be checked for the identification number of that work-part (Part ID). If the Part ID is found in the local database, it means that the part has already been fabricated by the system so that each workstation has already been provided with the processing data. Otherwise, if the Part ID is not found in the workstation's local databases, processing data for that work-part (Part ID) needs to be provided.

It is worth noting here that the original processing data for each work-part is stored in one place - the process planning database. What is sent to the local workstations, as part of pre-processing activities, are instances of these processing data. This transfer of data from the process planning database to the local workstation databases, in theory, needs to be done just once, only when a new work-part, which has never been produced by the system before, enters the system. Also, note that the role of the central process planning database should be seen as providing a backup function. Because data about processing work-parts on individual workstations can be modified either on the local workstation databases or in the central process planning database, appropriate authority and security policies and methods for synchronisation of data need to be applied.

This approach provides two important features of the overall system. Firstly, it increases the rate of *data coexistence in the distributed system* because work-part's processing data intended for each workstation can be defined and modified only at one place, in the process planning

database. Secondly, the *autonomy of workstations* is increased because during the work-part machining process workstations do not need to access any higher level of control since they already have the data they need in their local database. Under normal production circumstances, workstations independently and totally autonomously perform their processing tasks. Workstation agents only co-operate in peer-to-peer manner between each other, and they are thoroughly independent of any higher level of control.

Since at this stage of the application development, applications that deal with pre-production functions (such as, for example, computer aided design (CAD) and particularly computer added process planning (CAPP) systems) are not integrated into the simulation model of a heterarchical multi agent system advocated in this project, the work-part processing data were entered directly into the workstation local databases[1].

11.2.2 Activities during the production process

After the preparation activities have been accomplished the pallet with the work-part is transferred to the output transport port of the system input workstation. At that moment, a negotiation process for determining the workstation for executing the first operation can start.

Using the agent program's main menu shown in Figure 11-6 it is possible to change some basic program's settings. By pressing the "Settings" button, a window with a few setting parameters will be opened. Program provides option to choose between "step-by-step" or "continuous" mode of execution and to set some other basic parameters as depicted in Figure 11-7 (times are given in seconds). If step-by-step mode is selected, then all workstation agent activities related to agent negotiation process, which are otherwise performed in the background during the program execution, will be shown on the computer's screen.

11.2.2.1 Generating a task announcement and sending it to the destination agents

The system input workstation agent (the source workstation agent) generates a task announcement message and according to the part Processing Route Table (PRT) determines the workstation agents that need to be contacted. The message is sent to each workstation (the destination workstation agent) whose ID number is entered in the "WS-Alternatives" field in the column named as "Step01" in PRT. In the example given, (refer to Figure 11-5) the task announcement is sent to the workstations WS06, WS07 and WS08. Note that one task announcement message is created for each of these workstations and all messages are stored first in the buffer for sending messages. From this point on, the control for sending messages is the exclusive responsibility of the sub-module for sending messages (refer to Figure 10-2 "The functional model of a workstation agent").

An example of the buffer for sending messages is depicted in Figure 11-8. In the figure, the buffer contains only one, task announcement message, which in this particular case, is intended to be sent to the workstation WS04. TimeToLive is set to 1 meaning that only one attempt for sending a message is allowed. If unsuccessful, the message will be deleted from the buffer.

[1] Data can be entered on each workstation individually or can be entered from one "central place" since each local workstation database can be accessed from any computer on the network. To automate this process, integration between the workstation agent software and some of commercially available CAD/CAM systems could be considered as a part of future development. However, this integration should be approached only after the application had been proven in a real manufacturing environment.

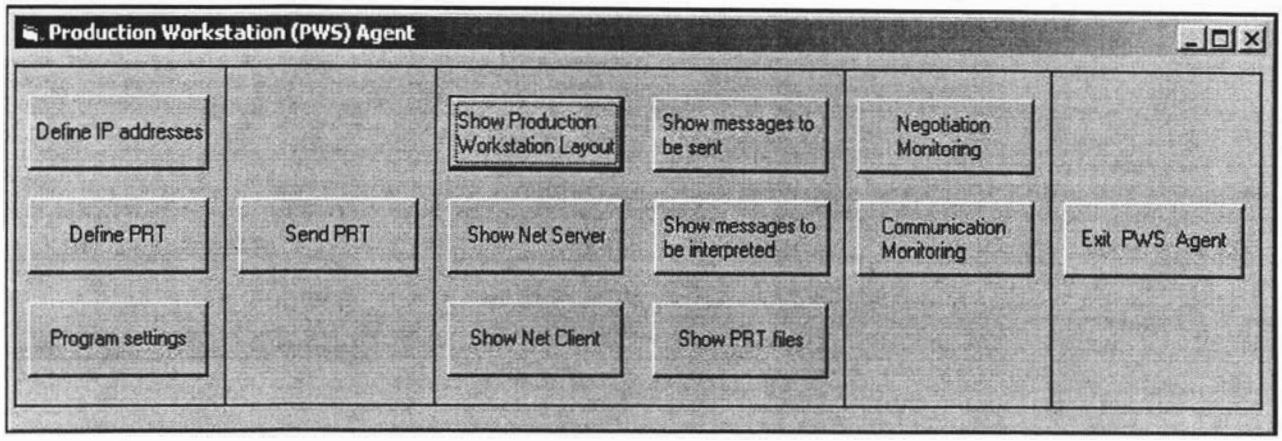


Figure 11-6. The main menu

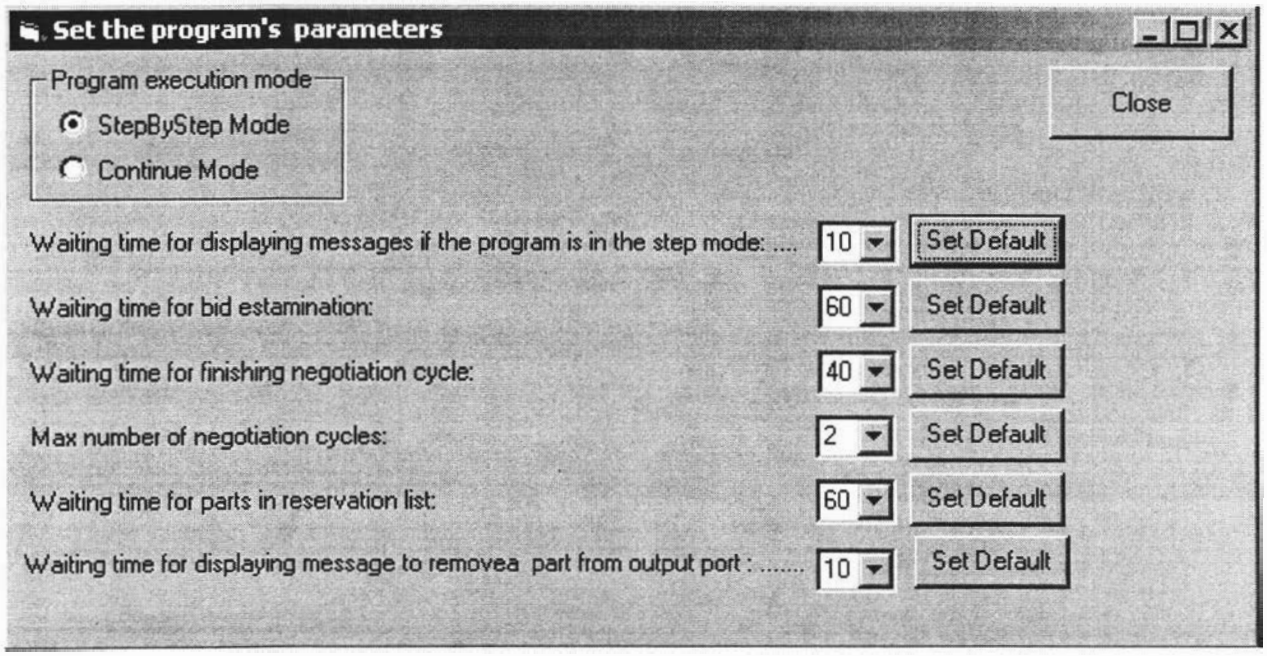


Figure 11-7. Parameter settings

Since each message has its priority index, messages in the buffer can be sorted accordingly to their total priorities. In the current program realisation the highest static priority is assigned to acknowledgment and PRT messages (a PRT message is not part of negotiation process and does not require a reply). Other message types have lower priorities and they decrease in the following order: acknowledgment message (priority 1), award message, bid message, and task announcement message (priority 4). Each time a message is not successfully transmitted over the network TimeToLive is decreased by one and MsgDynPty (Message Dynamic Priority) is increased by one. If the option "the highest priority first" is selected (see Figure 11-8) it means that unsuccessful messages will be pushed forward in the buffer, getting a new chance to be sent again more and more frequently with each new failure (till TimeToLive becomes equal 0). Each message, which is in the buffer, can be manually deleted at any time (using a right mouse click while pointing at the message and by selecting the delete option).

Each time when a message is about to be sent or interpreted (arrives in the buffer for sending or interpreting messages) an "information window" similar to one depicted in Figure 11-9 pops up on the screen. In addition to providing information to a user that a new message has arrived, the window provides the means for a user to take direct control over the negotiation process. Each

message can be discarded, or alternatively, the time when a message is put in the buffer (for sending or interpreting the message) can be postponed.

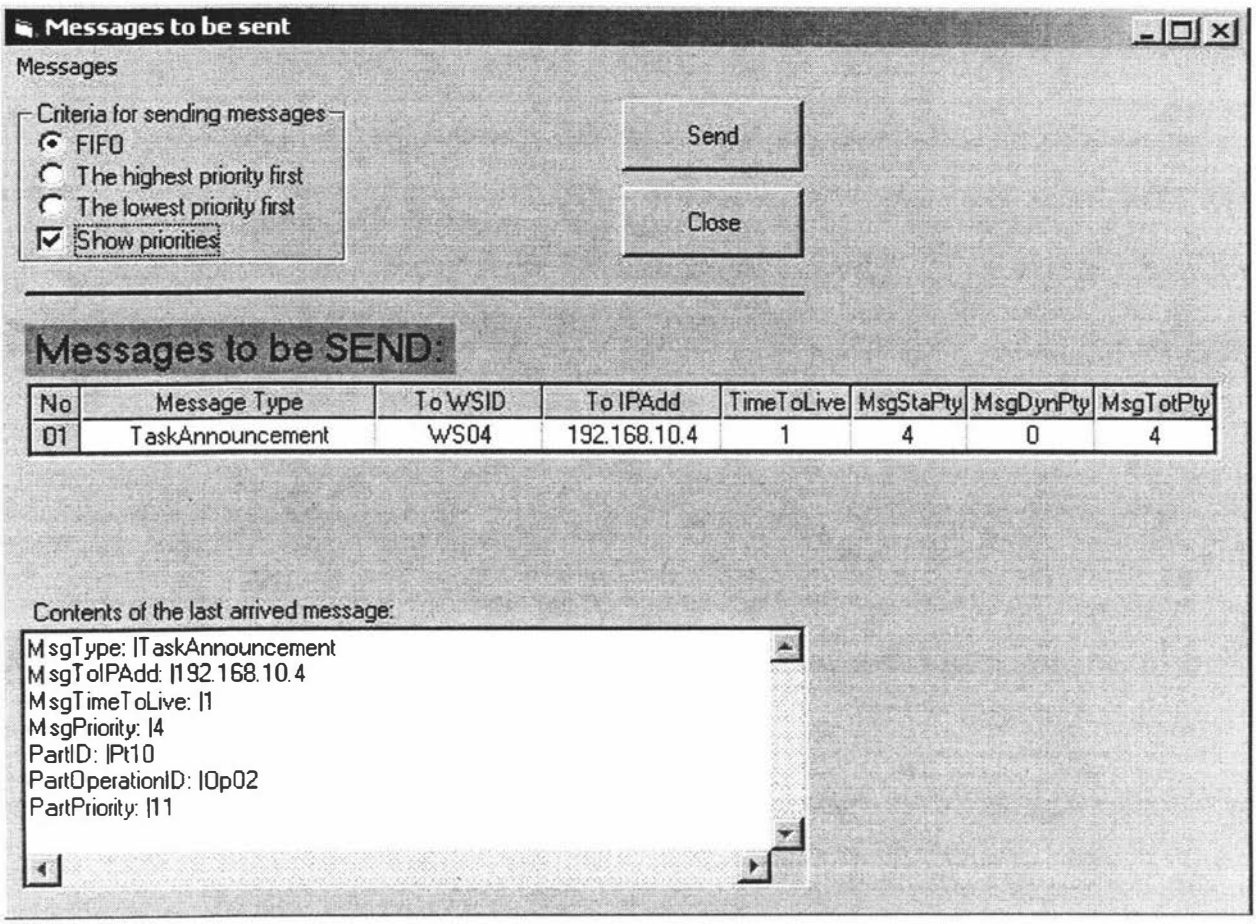


Figure 11-8. A task announcement message

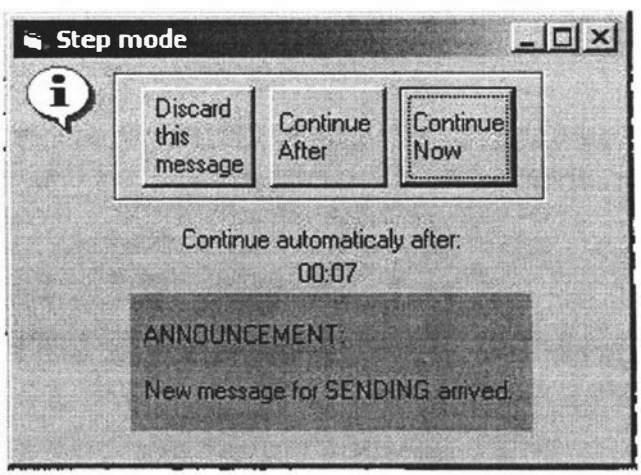


Figure 11-9. An information window - announcing arrival of a new message that needs to be sent

Prior to sending a task announcement message, a workstation agent creates a "negotiation container" in which the progress of negotiation process is tracked. Figure 11-10 depicts the window that shows the content of such a container. This window is also used for monitoring (and controlling) the negotiation process. The entire process can be aborted or frozen for an

arbitrary time (postponing a moment when negotiation process is to be terminated). As soon as a task announcement message is sent to the destination workstation, a container (bid evaluation table) for receiving bids is created. Figure 11-11 shows an example of such a container. The container comprises records for each workstation from which it expects to receive bids (to which a task announcement has been sent).

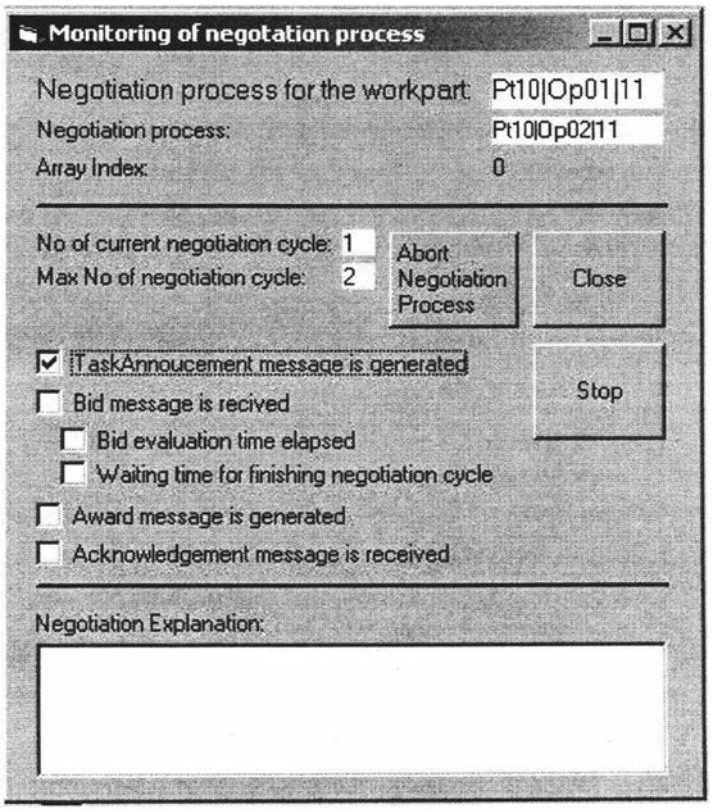


Figure 11-10. Monitoring of negotiation process

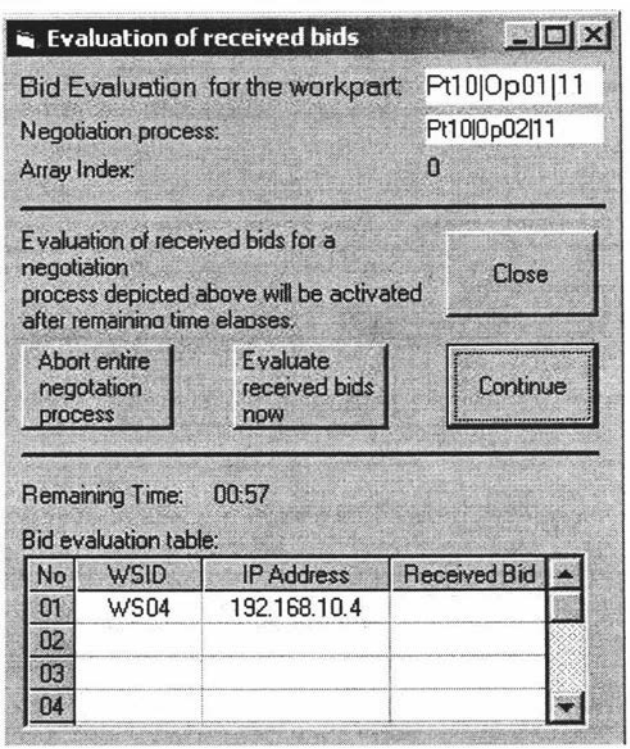


Figure 11-11. A container for bids that are to be received

11.2.2.2 Forming bids and sending them back to the source agent

After receiving and interpreting the task announcement message, each destination workstation agent starts a procedure for examining workstation's equipment status and collects all the necessary data for forming a bid. The agent's internal logic for forming bids is based on heuristic rules. The applied algorithm for forming bids can be based on any of the dispatching rules discussed in Section 4.3.2.2 "Priority dispatch rules (PDR)". For instance, bids can be created by using: shortest processing time (SPT), earliest finishing time (EFT), earliest due dates (EDD), shortest slack time (SST), customer priorities, and so forth.

Currently, since the goal of a control system is to *minimise the processing time* of the work-parts in the manufacturing system, the workstation agents form their bids by calculating earliest finishing times (EFT) for the work-parts. When doing that calculation workstation agents take into account:

- Current work (remaining processing time of the work-part which is currently being processed on a machine).
- Work already allocated to the workstation (work-parts in the input buffer).
- Booked work (work-parts that are allocated to the workstation but that have not yet arrived).
- Work required completing the work-part under consideration.

The current work, the allocated work, and the booked work include times for loading and unloading work-parts to and from a machine tool, the machine set up times for providing tools and for setting up additional jigs and fixtures if any.

Note that a workstation agent during the negotiation process (at the time when a bid is formed) takes into account the content of the input buffer to form its bid. However, bids represent only estimated times when a work-part is expected to be completed. In practice, these times may often be extended since the original sequence of retrieving work-parts from the input buffer is not firmly determined and can be changed over time. For example, if a work-part arrives at the workstation that has higher priority than any other work-part that is in the input buffer, then that work-part will override all the other work-parts (of a lower priority) and it will be machined before the others. This is contrary to the outcome that was expected for other work-parts at the time when their bids were formed.

When producing its response (a bid) each workstation agent pursues its own interests. For example, each production workstation agent will try to keep machine utilisation as high as possible (idle workstations will create bids which will receive a higher ranking in the selection process than those which have some work-parts already allocated to them). At the same time they will comply with the constraints (for example workstations with the maximum allowed number of waiting work-parts in the input buffer will not respond to a task announcement at all.) After bids have been generated by each of destination workstation agents they will be sent back to the source workstation agent that initiated the negotiation process.

An example of the contents of a "buffer for messages to be interpreted" is depicted in Figure 11-12. The figure shows only one message (a bid) that has been received from workstation WS04. Note that the message is automatically processed after a given time provided there is no human intervention such as deletion of the message. As with the "buffer of messages to be sent" any message in the buffer can be deleted at any time (for experimental purposes) by selecting a message and, after a right-mouse button click, selecting a delete options. An accompanying "information window" that informs a user that a new message for interpreting arrived is depicted in Figure 11-13.

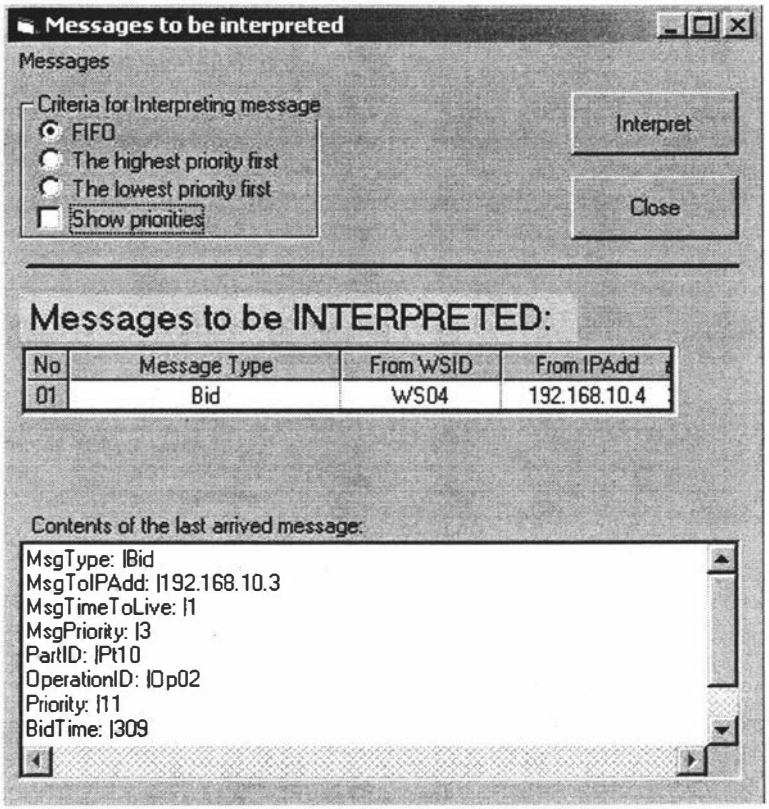


Figure 11-12. Contents of the buffer for interpreting messages

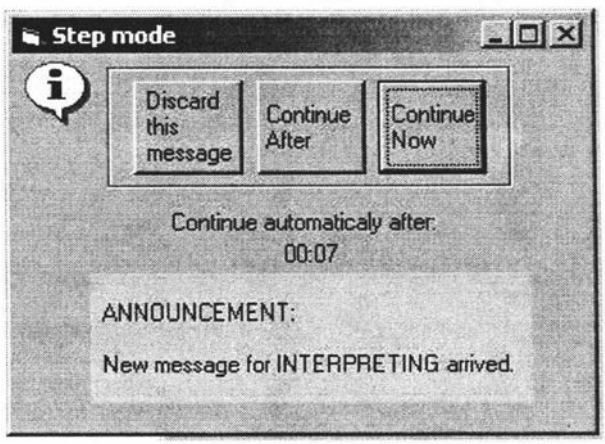


Figure 11-13. An information window - announcing arrival of a new message that needs to be interpreted

11.2.2.3 Selection of the winning bid and the generation of an award message

After sending a task announcement message, the source workstation agent creates a temporary buffer (a container called a "bid evaluation table") for receiving bids for the announced task and activates a time counter that counts down the predefined "time for bid evaluation" (an example of a bid evaluation table is shown Figure 11-11). The number of rows in a "bid evaluation table" is determined by a number of alternative workstations to which a task announcement was sent – for each workstation one row is created. When a predefined time has elapsed the agent activates the procedure for selecting the winning bid. If there are any bids in a "bid evaluation table" the agent will determine the best one and will generate an award message to the winning workstation. In our example we will make an assumption that all three workstations (WS06, WS07, and WS08) have submitted bids and that the best bid was a bid submitted by WS07.

If no bids have been received a workstation agent will terminate the negotiation procedure. In that case, a new negotiation cycle is triggered automatically at the time when the part reaches workstation's output transport port by sending another task announcement. If after this second attempt at negotiation, bids have not been received, a warning message is generated that a destination workstation has not been determined. The work-part will be removed from the output transport port into a temporary buffer (it should be noted that this is a software buffer while the real work-part will be temporarily stored in the workstation's output buffer) and after a while the user will trigger the negotiation process again by setting the work-part on the workstation's output transport port. At this point the workstation agent will start up a new negotiation process by generating another new task announcement and so forth.

11.2.2.4 Booking the workstation capacities and generating an acknowledgement message

After the award message has been received the winning workstation agent (in our example WS07) will reserve its resources. The part ID number will be recorded at the winner workstation's reservation buffer and the processing time of that work-part will be taken into account for determining the future bids generated by this workstation. The winner workstation agent will send an acknowledgement message to the source workstation agent (in the example, the system input workstation - IWS) confirming that it has received the award message and reserved the appropriate resources. When acknowledgment message is received, an "info window" similar to one shown in Figure 11-14 pops up advising a user that the winner workstation is acknowledged and that the ID number of the winner workstation is written in the PRT file. Otherwise, after a set time the system input workstation also checks whether an acknowledgement message has been received. If not, it cancels the previous task award and recommences the job negotiation.

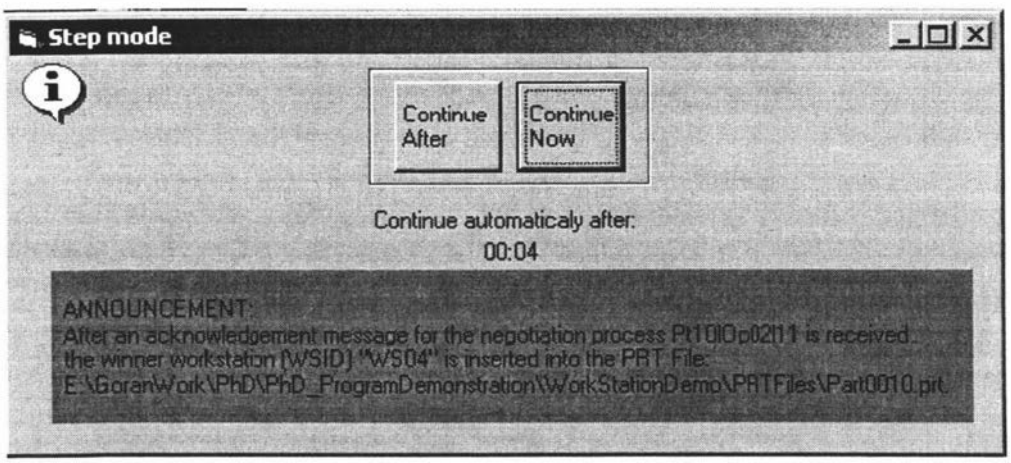


Figure 11-14. An announcement that the winner workstation ID will be recorded in PRT

11.2.2.5 Dispatching of a work-part

After receiving the acknowledgement message, a procedure for arranging the transport of the work-part can be activated. When the work-part leaves the system input workstation (IWS) the time is recorded in the work-part's Processing Route Table (PRT) and the PRT is sent to the destination workstation (WS07). When the manufacturing process of a work-part is finally completed, the PRT will contain data regarding the work-part's stay at each workstation involved in the work-part fabrication. This data can be used later for performing production analyses. As noted earlier, a PRT can be extended to accommodate any data that is of interest. Alternatively, data about work-parts, which have been processed by a workstation, can be

recorded in the workstation's local database. A final decision about what kinds of data is to be recorded and where the data should be stored has not been made at this point of the application development.

11.2.2.6 Arrival of a work-part on the destination workstation

The transmitted PRT (that corresponds to the work-part ID) will appear in a buffer of incoming work-parts (a software buffer) on the destination workstation (WS07). After a predefined time, which has been determined for performing a transport operation of the work-part, elapses (in the case of simulation), or when the work-part arrives on the workstation (in a real system), the Part ID number will be shifted from the buffer of incoming work-parts to the input transport port. In the current implementation of the heterarchical shop floor control software, this event will indicate that a simulation of workstation activities for that work-part can begin, or, in a real world application, that the work-part processing is under the jurisdiction of the workstation controller.

From this point on, the program execution in the test bed application of a heterarchical shop floor control system is similar to the procedures that are described previously in sections from 11.2.2.1 to 11.2.2.6. The negotiation process will be repeated for each task in the process plan, starting from the immediately following task, Task 02, to the last one, Task 06.

11.2.2.7 Main algorithms related to the negotiation process

The algorithm of the negotiation process, which is described in the example of two workstation agents in the previous sections, is depicted in Figure 11-15 and Figure 11-16. The algorithm embraces three asynchronous processes (negotiation process (a), (b) an (c)) which should be executed in separate threads. The algorithm and its components are self-explanatory and they will not be discussed further. (The algorithm is part of, and therefore is executed inside, one workstation agent).

11.2.2.8 Flow of work-parts inside the workstation

Once a work-part has been delivered to the workstation's input transport port, which is the input point where work-parts enter the workstation, a work-part is completely under the control of the workstation's controller, that is, any other controller outside workstation has no control over the work-parts inside a workstation. From that point on, work-parts follow a workstation level process plan. The workstation controller is responsible for handling work-parts inside the workstation and for synchronisation of all activities between the processing equipment, and the workstation material handling and storage equipment. When the work-part processing is completed, the work-part is transferred to the workstation output transport port. From there the work-part will be picked up by the shop floor transport system and it will be delivered to the next workstation.

The simulated workstation control module will undertake some predefined actions that depend on the work status of a robot and a machine tool (busy/idle), and the location of the work-parts inside a workstation as described further in the text that follows. Co-ordination among these actions inside workstation must exist and this is the primary role of a workstation's control module. In the current implementation of the workstation agent these control actions are simulated by a control module (refer to Figure 10-2 "A functional model of workstation agent"). The workstation controller in this case will carry out the following actions:

- Transfer a work-part from the input transport port to the input buffer
- Transfer a work-part from the input buffer to the machine

- Transfer a work-part from the input transport port directly to the machine
- Transfer a work-part from the machine to the output buffer
- Transfer a work-part from the output buffer to the output transport port
- Transfer a work-part from the machine directly to the output transport port

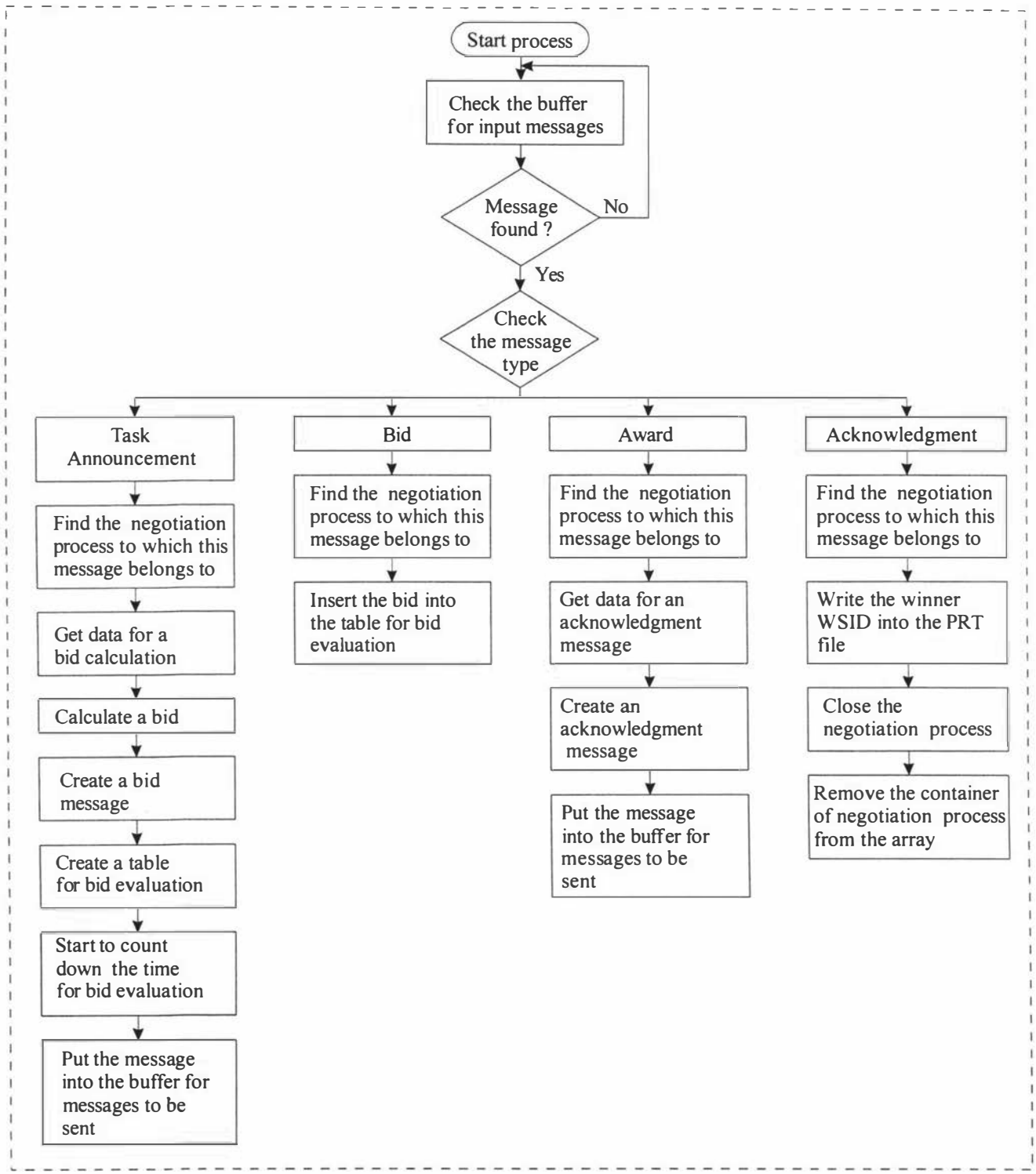


Figure 11-15. The negotiation algorithm (a)

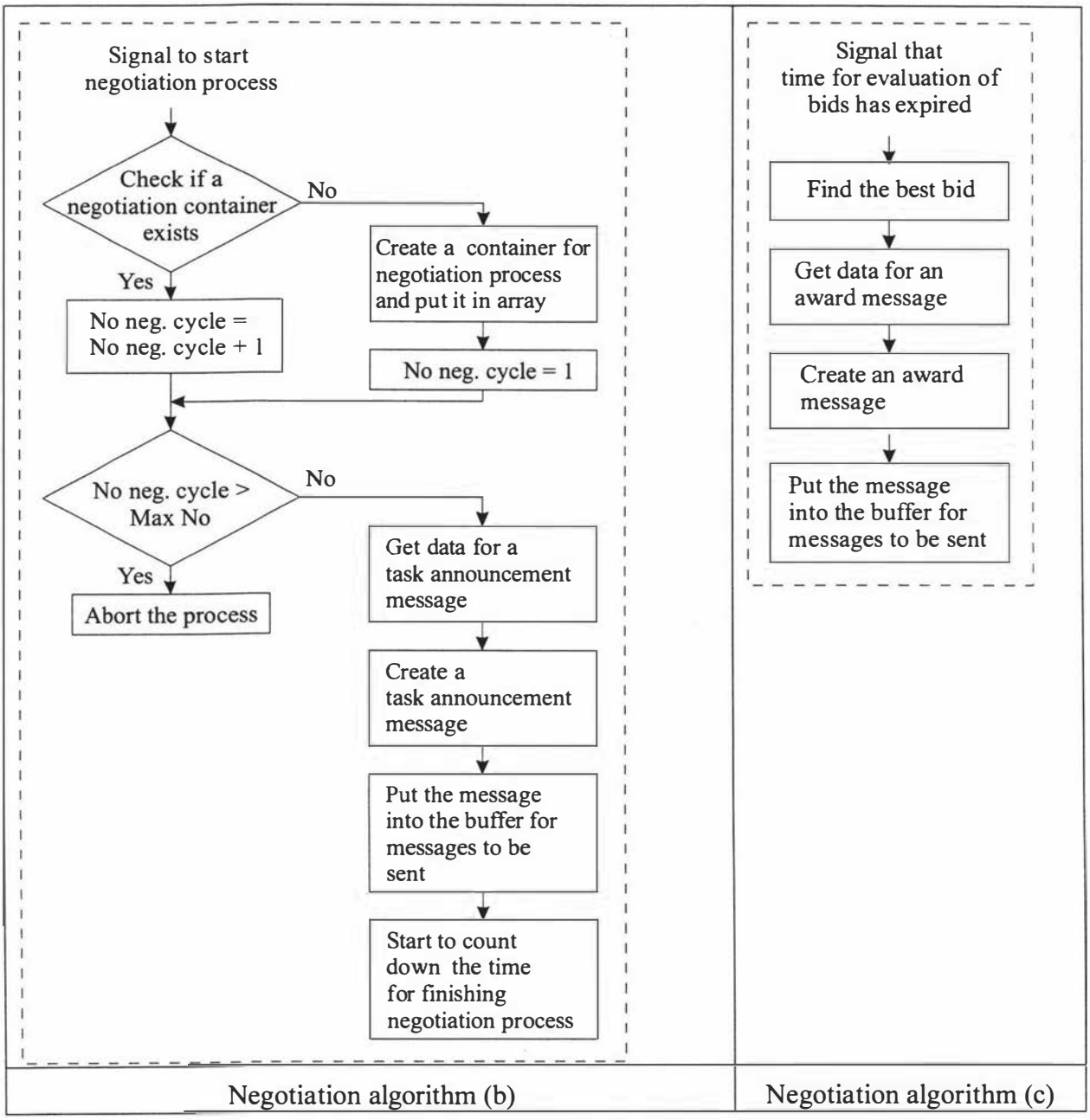


Figure 11-16. The negotiation algorithm (b) and (c)

When the work-part reaches the input transport port, the further flow of the work-part depends on the machine availability and the state of the input buffer. If a machine tool is busy machining other work-parts, the work-part will be transferred from the input transport port into the input buffer. If the machine is idle and there are no work-parts in the input buffer, the work-part will be transferred directly from the input transport port to the machine. If the machine is idle and there are some work-parts in the input buffer, then further action will depend on the work-part priority. If the work-part that has just arrived has the highest priority when compared with the priorities of the work-parts that are already in the input buffer then the work-part will be transferred directly to the machine. Otherwise the work-part will be transferred into the input buffer.

When a machine finishes work-part processing, a robot will transfer the work-part either to the output transport port, if there is no other work-part at it, or otherwise, into a workstation's output buffer if there is another part waiting at the output port to be transported to another workstation.

When the machine becomes idle the workstation controller will first check the input transport port. If there is a work-part at the input transport port and if it has the highest priority compared with the work-parts located in the input buffer, then the work-part will be transferred directly to the machine. If the input transport port is empty or if the work-part on the transport port is not of the highest priority, then the work-part in the input buffer, which has the highest priority, will be transferred to the machine.

Note that the movement of work-parts inside workstations is arranged in the opposite way to the work-part movement outside workstations as follows:

- In the negotiation procedure described above, when the work-part machining process is finished on a workstation, the workstation agent looks for the next, most suitable workstation (through the negotiation process between workstations) for execution of the next processing task on the work-part. In this way, by selecting the best bid, work-parts push themselves towards workstations. Therefore, external to the workstations, work-parts are ***pushed through the system.***
- Contrary to the push process external to the workstations, inside workstations the work-parts are moved in the opposite way. After finishing a current work-part, the machine looks for the next part in the input buffer. Using the work-part priorities, the machine is always able to select a work-part with the highest priority, namely the most urgent one. In this way, the machine pulls work-parts from the input buffer towards itself. Accordingly, inside a workstation work-parts are ***pulled towards the machines.***

The current version of the test-bed application supports both arrangements.

11.2.3 Post-process activities – the system output workstation

The production process of a work-part is completed when the work-part finally reaches the system output workstation. The work-part is removed from the pallet and after final inspection it is stored in the magazine for finished products or outwards goods warehouse. The pallet is sent back to the system input workstation (ISW). The work-part's PRT file is stored on the system output workstation computer. As noted, this repository of processing data can be used for conducting all kinds of production analyses and for other business purposes.

11.3 Demonstration to the New Technology Development Operations Group

As part of the process of demonstrating that the system did work and in an attempt to obtain funding from the Massey University New Technology Developments Operations Group the working prototype was shown to the above group. The purpose of the Group was to evaluate new developments which could be turned into commercial products providing Massey University with additional research income through royalties or sales.

Although the Group agreed that the system worked and had good potential it was recognised that with the changes that had occurred in the Technology Faculty and the Institute of Engineering it was unlikely that the Massey University could support the future development with both research funding and staff salaries and Massey did not have sufficient funds available to undertake the full commercial development so the project was not supported as stated in the letter from the Group a copy of which is provided in Appendix 2.

11.4 Summary

In summary, the functional software for the heterarchical agent based shop floor control system was developed to perform the tasks and provide the outputs described above using Visual Basic and MS Access. The communication module of the workstation agent was based on ActiveX controls IPDeamon and IPPort as discussed in Section 10.2.1.2 "The communication module". The complete software was implemented on a small TCP/IP based network of low specification IBM compatible personal computers.

The next chapter discusses the results achieved in this project using this software.

Chapter 12. Discussion of results

In the previous chapters the main design points of a distributed, heterarchical, multi-agent, shop floor control system envisaged in this project were outlined (see Chapter 10) and the operation of the network application, developed to demonstrate how the proposed system works, was described (refer to Chapter 11). This chapter discusses the results achieved in the research study (refer to Section 12.1), addresses the project's main contributions (refer to Section 12.2), and finally, it summarises some of the key project achievements (refer to Section 12.3).

An essential element of the project was that the system was to be based on low cost and commonly available computing elements acting autonomously and thus giving many of the advantages claimed for the theoretical concepts outlined in chapters 05 and 06. It was not the intention to investigate the particular theoretical approaches but to select the appropriate elements from the above concepts and ideas for the proposed system. The project was, therefore, to develop a demonstration system including processing and networking hardware and software from existing commercial units. However as the project developed so did the technology available and it became clear that even with "low cost commonly available elements, a highly sophisticated system could be developed which could demonstrate not just a single factory scheduling and control system but, in fact a system that could be extended to a "web centred" or even a "virtual" organisation using the Internet and Intranet technologies. This extremely rapid introduction of the new technologies made the progress of the project somewhat haphazard and at times rather frustrating in that often new software would be released when considerable effort had been made to overcome the shortcomings of the existing software, with the new software solving many of the difficulties of the old software. However, it was possible to assemble a small scale demonstration system within the limited resources available to the author that demonstrated that the concept was practical and sound.

12.1 A retrospective view of the research objectives

Following the sequence of the research objectives that were set out in Section 7.6 "The research objectives", the results of the study are discussed in the following subsections.

12.1.1 A job shop manufacturing system reference model

To design an efficient heterarchical Shop Floor Control (SFC) system it is important to determine the "right level" of manufacturing system decomposition. As discussed in Section 8.1 "A modelling framework", a compromise between two asymmetrically opposite tendencies has to be made: to decompose the system as much as possible (to achieve the maximum flexibility and reliability of the system) and as little as possible (to increase efficiency and to simplify design of the system). To assist the determination of the "right level" of manufacturing system decomposition, firstly, the elementary manufacturing system components are defined and classified (as processing and supportive entities). Afterwards, the key, higher-level functional/organisational production units are identified and defined (as a workstation, a workcell, a production cell, and a shop). After a preliminary analysis of relationships that would exist among potentially different distributed entities and the complexity that would originate from these relationships considered from the view point of design of a heterarchical shop floor

control (SFC) system it was decided to adopt the workstation as the fundamental controlled element - a kernel - around which the whole heterarchical SFC systems was built. The workstation is the basic autonomous production unit that is capable of performing a variety of manufacture operations on different work-parts.

After adopting the workstation as the basic production unit, a reference model of a job shop manufacturing system was defined. This model was used as a blueprint for developing a heterarchical SFC system.

12.1.2 A heterarchical shop floor control reference model

On the basis of the adopted manufacturing system model, the operational model of the agent-based SFC system was defined and this model was discussed in Section 10.1 "The proposed control architecture of a heterarchical shop floor control system". In this model each physical workstation in the job shop manufacturing system is mapped by an autonomous control unit – ***a workstation agent.*** Agents are mutually interconnected via a LAN forming the heterarchical control structure at the shop floor level. Inside workstations the control is performed in a traditional, hierarchical manner.

The agent was designed and developed as a software package to be installed and run on every workstation's computer (either the workstation control computer if it is based on a PC with a Windows operating system or on a separate low cost agent PC). This design contributed significantly to the "reusability" of the code developed for each agent. That is, much of the software developed for one workstation can be used with minor modifications in other workstations. All production workstations in the model of manufacturing system discussed in this thesis had the same structure. However the system input and output workstations had different structures. That is, each production workstation was composed of an input port, an input buffer, a machine tool, an automated piece handling system such as a robot, an output buffer, and an output port. A production workstation agent application can be easily modified (due to its modular structure) to accommodate different workstation structures. For example, if a workstation was deemed to consist of two machine tools instead of one, by introducing an additional sub-module, which would simulate the activities of that second machine, the workstation agent application could be easily modified to accommodate this change. In addition to this, a module that simulates the control logic of workstation's controller also needs to be modified (extended). Another example of a simple and practical modification is to replace the robot by a human operator and to have the operator indicate when the part movement has started and when it is finished through the control display shown in Figure 10.6.

The best example that illustrates software reusability was the development of the workstation agent software for a system input workstation and a system output workstation. The software of the production workstation agent was modified slightly to reflect differences in the structure of the physical workstations on the shop floor. System input and output workstations do not use machines and robots in their configurations. Instead they have "work benches" and are manually operated. Accordingly, the robot and machine modules were taken out from the production workstation agent software, and the control logic (which provided the triggering of the events automatically, such as the events that indicated start and finish of the machine and robot operations) was replaced by modified user interfaces. These graphical interfaces, which represented the layouts of corresponding workstations, enabled the simulation of work-part movements inside the system input and output workstations. That is, symbols of work-parts were moved around the graphical interface, by using mouse drag and drop operations. Events that register these changes in the location of work-parts inside workstations, therefore, were triggered manually, instead of automatically by the module that was responsible for simulating

machine and robot operations in the production workstations. To recall, maintaining locations of the work-parts inside workstations accurately and at all the times during the fabrication of work-parts is of great importance for proper functioning of the proposed system.

12.1.3 Requirements of distributed heterarchical control systems

Because of the basic concepts that are embedded in the proposed approach, the model of a shop floor control system adopted addresses many of the requirements identified in Section 5.3 "Main requirements of the new generation of manufacturing control systems". The following text explains how these requirements are met by the system.

- ***Autonomy:*** – Each workstation agent represents self-contained software unit that has decision-making, planning, and action taking capabilities (in a tight cooperation with the workstation controller). This autonomy enables a workstation agent to behave reactively and pro-actively in a dynamic environment.
- ***Co-operation:*** – Workstation agents cooperate using the negotiation technique. This allows agents to establish flexible interactions among themselves making addition and removal of agents to and from the system a possible and an easy task.
- ***Self-Organisation:*** – The proposed control system immediately re-negotiates the way in which the manufacturing operations will be performed whenever the environmental conditions change.
- ***Reconfigurability, modifiability:*** – The workstation agent has been built using a modular, object-oriented approach. Therefore, each agent can be easily modified to accommodate any workstation's configuration. At the system level, workstations can be added or removed from the system reliably and quickly during the run time, while the system is operational. To accommodate system changes a local workstation agent database has to be updated (IP addresses and data related to work-parts).
- ***Adaptability:*** – In the case of disturbances which appear in the system during the work-part production, the manufacturing system will adapt to new circumstances and will continue to operate (in a degraded state) as long as at least one possible processing route for a given work-part exists in the system.
- ***Fault tolerance:*** – The control system offers a high level of fault-tolerance for two reasons: 1) the dependency on other entities in the system is minimised and 2) the system does not rely on the existence of global information. When a fault occurs in the system it is confined to the workstation level rather than being propagated throughout the system. The control system will try to accommodate additional requirements for extra capacity by routing work-parts to the alternative workstations (if such "spare capacities" exist in the system). Refer to Section 6.3.6.1.4. "Increased fault-tolerance" for additional details.
- ***Scalability:*** – The system can be extended without disrupting previously established links among workstation agents. New workstations can be easily added in the system, and moreover, they can be located in different geographical areas. Instead of via a LAN, workstations can be interconnected over the Internet. At the workstation agent level, the agent can be easily upgraded (because of its modular structure) and individually tailored to specific requirements that might be necessary in the future.
- ***Heterogeneous environments and Interoperability:*** – The control systems can accommodate heterogeneous software and hardware in both their manufacturing and information environments. Generally speaking, individual agents can be implemented

using different programming languages and can operate on different computing platforms. Note that the current workstation agent software realisation does not provide this particular kind of interoperability namely running on versatile computer platforms.

12.1.4 Methodology for resolving real-time scheduling and resource allocation problems

The methodology used for resolving real-time scheduling and resource allocation problems in the heterarchically controlled manufacturing environment is discussed in Section 10.3 "Scheduling in the model of a distributed heterarchical shop floor control system". It should be noted that the scheduling function in the proposed model of a heterarchical SFC system is performed in two stages, that is, two scheduling processes can be distinguished. One is conducted among workstations (so called "global scheduling"). This scheduling is based on two main principles: the principle of the "reactive scheduling", and the principle of "last commitment". This approach ensures that the hard and elusive task of forecasting conditions on the shop floor is reduced as much as possible. A decision for selecting the most appropriate workstation to perform the next operation on a work-part is postponed to the latest possible extent (see Section 10.3.1"Global scheduling - scheduling among workstation agents"), so that a decision is made by taking into account only information that reflects the current state of the workstations on the shop floor and not the state that is expected to occur some time in the future. The other scheduling process is conducted inside workstations (so called "local scheduling"). In the proposed approach this local scheduling is based on the use of sequencing rules – priority indexes (see Section 10.3.2 "Local scheduling - scheduling methods for each individual production workstation agent"). However, since the complexity of the scheduling problem is significantly reduced (scheduling several work-parts from the input buffer on a machine tool is a well known "one machine" scheduling problem) more sophisticated techniques for finding a better solution can be applied.

The test-bed application demonstrates how both global and local scheduling processes are successfully conducted in the proposed system.

It should be noted that some researchers argued that distributed heterarchical control systems would produce inferior schedules in comparison to the schedules produced by centralised systems. They considered that a global optimum of the system would not be obtainable. Though this may be theoretically a valid statement, one could argue that:

- In manufacturing systems where a large number of disturbances happen frequently, any schedule is valid for a short time only anyway. In such environments, in which a system never reaches a stable state, to get even a "good" schedule is a difficult and often impossible task;
- The most commonly used scheduling algorithms in industry, forward/backwards algorithms, can be placed in an agent architecture without any loss of functionality. *"In fact, once these algorithms are distributed among a network of intelligent agents, they can achieve superior performance due to the increased distributed computing resources available to solve the problem".* [Parunak, 1996a].

12.1.5 The information that is required for making rapid and effective scheduling decisions

Determining the information that is required for workstations agents to make rapid and effective scheduling decisions is known to be a difficult and often elusive task. There are no strict rules that could be applied. Instead, data should be carefully analysed and the final selection of data

would depend upon many factors. One of them is the criterion that is used in the negotiation processes for determining a most appropriate workstation for executing the next processing task on a work-part. For example if a goal of the system is to select a workstation that will accomplish a task in most economical way, one kind of data is required (for example, data required for determining a monetary equivalent of costs for executing a processing task on a workstation). If the goal is to find a workstation that will accomplish a processing task in the fastest way (as it was done in this study), then other data will be required (cycle time, set up time, loading/unloading time, etc). Section 10.2.1 "The functional model of the workstation agent" discusses the data used in the current realisation of the test-bed application.

12.1.6 A mechanism that links process planning and scheduling functions

In the present study, the process planning and scheduling functions are integrated by using Processing Route Tables (PRTs), as discussed in Section 8.1.3 "Integration of the process planning and shop floor scheduling and control functions". PRTs are used as the means to deliver several processing alternatives (plans) down to the shop floor. Instead of selecting only one process plan prior to the manufacturing of a work-part (usually from a set of several feasible alternatives), the several different process plans were defined in a "net" form by using Processing Route Tables (PRT). In this way, a final decision about which process plan is to be executed (that is which a process route will be followed) is postponed to the latest possible time, that is, until the time when the process of fabrication of a work-part is actually being conducted on the shop floor. The motive for adopting this approach is found in two important points.

- This approach significantly increases the processing flexibility of a manufacturing system by offering several processing plans (alternative routes) that can be undertaken.
- This approach enables the decision about which process plan is to be taken is made "on-line" accordingly to the current conditions on the shop floor. The PRT enables the scheduling function to select the best process plan (a processing route) on-line and in the real-time, during the production of a work-part, taking into account current conditions on the shop floor.

It should be noted here that the overall performance of the system is not globally planned in advance, prior to the work-part production. Instead, the performance emerges through a dynamic interaction that occurs among the workstation agents in real-time and on-line while the system is running. In the similar manner, a short-term schedule of manufacturing resources at the shop floor level arises from the concurrent, independent decisions of the autonomous workstation agents. The resulting schedule is responsive to changes in the availability of individual workstations and eliminates a need for a complete rescheduling of the whole manufacturing facility because, for example, some of the elements in the original plan have become unavailable or cannot meet the original throughput time for their operations due to reduced speed.

12.1.7 Analysis and distribution of manufacturing data among workstation agents

A problem of data location in distributed heterarchical control systems is yet another difficult task that has to be approached carefully. Again, there are no a strict rules to assist this task, but generally speaking, the more data that is kept locally (inside a workstation) the better as 1) the time for data gathering is much shorter compared with the situation when they need to be collected from different distributed resources and 2) the current validity of information is much

higher due to short elapsed time between setting a request for information and processing this information. By introducing local workstation databases a great amount of data can be encapsulated and kept locally inside workstations. Having local workstation databases many data can be promptly retrieved by a workstation control module for conducting timely critical decision-making scheduling and control processes. This "availability of information" is an important feature not only for controlling versatile machinery inside workstations (in real manufacturing systems) but also for facilitating an intensive inter-workstation cooperation (negotiation) processes that is a prerequisite for successful operation of heterarchically controlled manufacturing systems.

However, keeping all data locally is not always possible or practical. Sometimes, some data has to be (or it is more convenient to be) located in the other repositories, rather than inside the workstation's local database. For instance, data that are closely related to the work-parts and that is constantly changed (or might be changed) over time, during the work-part manufacturing process, should be allowed to "travel" through the system together with the work-parts. The best examples of these data are the total estimated remaining processing time and the due time. In the proposed model of heterarchical SFC system, for this purpose the Processing Route Table (PRT) is used. The data used in the current realisation of the test-bed application, and how and where the data are stored, are discussed in Section 10.2.1 "The functional model of the workstation agent" and Section 11.2.1.2 "Getting the Processing Route Table (PRT)". It should be noted that the above distribution of data is only one proposal that was demonstrated to be feasible. Other data distributions can be envisaged and adopted. However, two main flows of information should stay the same.

The two major flows of information in the proposed system are inside a workstation agent and between workstation agents. The former is conducted between:

- A local database and a workstation's control module,
- A workstation's control module and a communication module, and
- A workstation's control module and any other sub-ordinate equipment level controllers that might be located inside workstations.

The flow of information between agents to some extent follows the physical flow of work-parts through the system. For example, if a work-part is to be transferred from workstation WS01 to workstation WS02 a flow of information will follow the same route.

Beside these two main flows, yet another flow of information could be perceived in the system: the flow between workstations and a monitoring system. This flow is used for gathering data about the "production status" on the shop floor (in one direction) and for a system manager's control interventions (in another direction). This flow would be of much less intensity compared with the previous two since the corrective interventions would occur relatively infrequently in the system.

12.1.8 Definition of messages that provide efficient negotiation processes between workstation agents

To facilitate efficient negotiation processes between workstation agents a set of messages was specified and defined. In the proposed system only four types of messages were used, namely: a task announcement, a bid, an award, and an acknowledgment message (refer to Section 10.2.2 "Co-operation and communication between workstation agents – the implemented negotiation mechanism"). These messages represent slight modifications of those that are found in the original Contract-Net Protocol. This initial set of messages can easily be extended by an

arbitrary number of new messages to facilitate more sophisticated negotiation processes. The test-bed application demonstrates the execution of an auction-bidding scheme and demonstrates how the negotiation processes among workstation agents are conducted.

It should be noted, that in contrast to the classical Contract-Net Protocol, in the system developed for this project each workstation is responsible for monitoring and the execution of its own tasks.

12.1.9 Selection of a computer platform and software technology

Since one of the goals of this project was to demonstrate feasibility of the proposed concept, as well as the feasibility of developing and implementing a heterarchical shop floor control (SFC) systems by using commonly available hardware and software tools, attention was paid to the selection of suitable computer platforms, network systems, operating systems, and programming tools. A discussion of the available alternatives, as well as the reasons why each of adopted options was selected, is given in Section 9.1 "The choice of tools for the development of a test-bed application of a distributed heterarchical shop floor control system". In this section, in addition to some extra comments, only the outcome of the previous discussion is presented.

For developing, testing, and implementing purposes of the model of a heterarchical SFC system, a small experimental ***Ethernet Local Area Network*** (LAN) was established. The LAN consisted of eleven ***IBM compatible personal computers*** (eight 486 PCs running at 33/40/50/66MHz and three Pentium PCs running at 100/133/366MHz). Each PC used the ***Windows 95 operating system.***

The computers were first assembled and tested as stand alone machines. The major issues were related to resolving conflict situations and eliminating faulty components. Once these problems were successfully resolved and when the computers were brought to the operational stage, further work with them was trouble free. Setting up the LAN was a task that was also accomplished with only minor problems. The network was unstable at the beginning (which was very upsetting because sometimes it worked well and other times it did not). The rule that must be followed is that both ends of a coaxial cable must be terminated with 50ohm terminators and ***only one terminator has to be grounded***!). During developing and testing the test bed application of a heterarchical SFC system, the Ethernet LAN worked well and smoothly, without causing any problems.

The software model of a heterarchical SFC system was developed using the ***Visual Basic (VB) integrated development environment*** (for developing a core part of a workstation agent) and the ***Microsoft Access application*** (for developing a workstation agent's relational database). This set of tools was selected because it offered a powerful, yet one of the quickest platforms to master for software development. In spite of this, it should be said that a great amount of time was invested for just that purpose – to learn the state-of-the-art software technologies and the accompanying techniques for successful development of the test bed network application.

As it will be discussed in the next section, it was found that initial adoption of programming tools should have been extended with C++ or Java program languages. Also, it turned out that the Windows 95 operating system was not an adequate choice. To provide a more stable, powerful and a real multitasking developing platform, Windows 95 should have been replaced with Windows NT operating system. These considerations would have been appropriate at the time when decision about the development platform had been made. Nowadays, if staying loyal to Microsoft, Windows 2000 or XP professional versions would be adequate choices for the operating system. Refer to next section for further discussion.

12.1.10 Design and development of the workstation agent

Section 10.2 "The workstation agent" described the design of the workstation agent software. A workstation agent provided a set of graphical interfaces, that a user could use to control both:

- A simulation of machine processing and robot material handling operations (a main screen of a workstation agent application has a few control buttons that could be turned on and off affecting the course of execution workstation activities, for example, simulating a situation when a machine or a robot are out of order) and
- The execution of negotiation processes between workstation agents (specifically design forms provide the means for looking into the buffers for sending and receiving messages. By using these forms messages could be edited or deleted).

One of the most time consuming tasks during the application development was to get acquainted with the network technologies and protocols in order to properly design and develop the workstation agent's communication modules. However, once the third party ActiveX controls: IPDaemon and IPPort were obtained from the Internet this task was significantly simplified. It should be noted that because these components were demonstration versions, they did not have accompanying manuals, so that developing the communication modules (in Visual Basic 4) and establishing communication between two PCs took some time as well. However, as much as these ActiveX components were helpful they also caused the most trouble, as it will be discussed later in Section 12.1.11 "Demonstration of a test-bed network application on an experimental LAN".

Initially, client and server communication modules (for sending and receiving messages over a LAN) were designed and developed as separate applications. After they had been thoroughly tested, the communication modules were integrated with the rest of the workstation agent software. These modules cooperate closely with the modules for creating and interpreting messages that were added later.

To demonstrate the operation of the proposed heterarchical SFC system, manufacturing activities inside workstations had to be simulated. The intention was not to provide sophisticated simulation of workstation activities (including attractive and effective animation effects), rather, it was decided to simulate manufacturing processes in the most simplistic form that provided a bare minimum for demonstrating the work of workstation agents as well as the application as a whole. Progress bars (one for a machine tool, and another for a robot) were adopted and they were found to be a satisfactory solution in terms of visualising the "progress of simulation" of time consuming machining and material handling operations. At that stage of developing workstation agent software, that is, the modules for the simulation of the processing machine's and the robot's operations, another challenge was faced. Namely, in the current realisation of the software, these modules had been developed as separate procedures driven by Visual Basic (VB) internal "Timers". Because these procedures were essentially a part of one monolithic, single-threaded program (refer to Appendix 01 "Multi Threading") this caused a problem in the development of the workstation agent software. An explanation of this issue is given in the following two paragraphs.

To provide responsiveness for the application during the simulation of long machine and robot handling operations each procedure for their simulation included a "DoEvents" command that is widely used by VB community for this purpose. However, the use of "DoEvents" commands did not provide a satisfactory solution when developing the workstation agent software, because when they were used it was not possible to have full control over the program execution flow. For this reason, it was decided that these procedures should be developed to run as separate, independent, out-of-process COM (Component Object Model) objects that would be hosted by

the workstation agent application. Unfortunately, in the attempt to do this, it turned out that Visual Basic (VB) had severe intrinsic limitations for developing "multi-threaded" applications (refer to Appendix 01 "Multi Threading").

To elaborate this point further the following explanation is provided. As noted in the previous paragraph, since VB supports the event driven programming model, the "DoEvents" command is often used inside "time-consuming" procedures to make an application responsive to the events that might occur during the execution of these long lasting computational tasks. A good example of these tasks is a printing job or a simulation of a machine processing operation, as is the case in our example. In ideal circumstances, in a single-threaded application, the event will trigger execution of some other procedure (a portion of programming code) and after accomplishing it, the program control will be return back to the original procedure. The execution of the current task would then continue starting from the point where it was interrupted. This is seen as a good solution when the new task (a procedure) is to be executed in the much shorter time in comparison with the time that it would take for the original procedure to be accomplished. (In the example of printing, a user would not notice this interruption at all because of the existence of a printer buffer). However, because the procedures for simulation machine's and robot's operations are driven by VB "Timers" (that are in turn driven by the internal computer clock) it sometimes could happened that, when the control is returned back to the original procedures (particularly if the execution of this "new task" (a procedure) requires significant computer time) the predefined times for simulation of these activities may elapse causing further execution of the program to fail. The program would be blocked and the application would appear as if it was "frozen", and indeed it was. Otherwise, if the execution of a "new task" is finished before the simulation process was supposed to be finished, the program would continue to function "normally" as it was envisaged to do. This behaviour is clearly visible on the corresponding progress bars. At the time when the "new task" starts the progress bar will be "frozen" for a while (in spite of the fact that the simulation time will continue to progress further in the background). Then, at the time when execution of the "new task" is finished, the progress bar will jump to the stage where it is supposed to be (as if the "new task" had not been executed at all). In summary, when a "new task" requires less time to be executed than the remaining simulation time (remaining simulation time at the moment when the event that calls for the execution of that new task is triggered) the application works well; otherwise it does not.

The solution to this problem, (at the time when the program was developed in 1999) as noted above, is to extract and encapsulate procedures for simulating machine and robot operations from the workstation agent software and to develop them (for example by using C++) as separate out-of-process COM objects - ActiveX components (see Appendix 01 "Multi Threading"). These COM objects can run in separate independent threads and would expose only their properties, methods, and events to the host VB workstation agent application. Also, it would be highly advisable to restructure the whole application in the same fashion (to be designed as a set of a few independent, multi-threaded processes). To use the full power of multitasking execution, instead of being running on the Windows 95/98/Me operating systems (the kernel of which is DOS based) the application should be transferred on Windows 2000 or XP professional workstation operating systems whose kernels were designed to support real multitasking operations. It should be noted that the task of developing COM components in C++ to run in separate threads is a programming task for a skilled C++ software engineer or an experienced C++ programmer. In this case it should be able to be accomplished in a relatively short time (in the range of couple of weeks).

Contrary to the limitations that were encountered with VB during developing a workstation agent software, work with MS Access progressed smoothly. It has been found that the selection of MS Access for designing and developing a small relational database for storing and retrieving

information kept locally in the workstation agents was the right choice. MS Access provided the right set of tools for the successful design of the underlying tables, definition of relationships between tables, and for developing forms (user graphical interfaces) for entering, editing, and deleting data in the local workstation agent's database.

The workstation agent software is interfaced with the software with the workstation agent local database and can automatically read, edit, and write data to and from the database.

Note that in the meantime (since 1999, when the project was completed) new technologies for software development emerged. For example, Microsoft's Dot Net Framework with new concepts/techniques such as: Common Language Runtime (CLR), Basic Class Libraries (BCL), Intermediate Language (IL), Just-In-Time (JIT) compilation, Assemblies, Garbage Collection (GC) etc., provides much more powerful environment for software development. Encountered multithreading issues can be easily resolved today using, for example, Visual Basic Dot Net (VB.Net), which is a truly object-oriented language, C sharp (C#), or some of more than 30 other programming languages (presumably not all can support multithreading) that can run on the .Net platform nowadays.

12.1.11 Demonstration of a test-bed network application on an experimental LAN

In Chapter 11 it was demonstrated, by using the application that was running on an experimental LAN, that a proposed concept embodied in the application was valid and that the fully functional and operational heterarchical SFC system can be realised using personal computers and commonly available operating systems, networks, and programming tools. The system has demonstrated that the computational loads for a heterarchical SFC system were very low and that they were not difficult even for obsolete IBM compatible 486 personal computers.

During testing and experimenting with the system, two issues regarding sending and receiving messages over a LAN were noted. One was easily resolved while the other still requires consideration by professional software engineers. Namely, in spite of the fact that messages have been transferred quickly and smoothly over a LAN, (all the messages were sent almost immediately[1]); there were situations when messages were not transferred at all!

Firstly, it was noted that when a source workstation agent could not establish a connection to a destination workstation agent in the first attempt (for example, when a destination agent was not operational since it was deliberately turned off - simulating a workstation's breakdown) and when there were a lot of messages in the temporary buffer for sending messages on the source workstation agent (see Section 10.2.1.2 "The communication module") some messages did not get a second chance to be retransmitted. To recall, each message to be sent is temporarily stored in the buffer for outgoing messages. A client communication module is permanently monitoring this buffer and if there are some messages to be sent, the client module takes the first message from the top and tries to send it to its destination. Then it takes the next message and tries to send it to its destination. This process is repeated for any message that is in the buffer. Note that a cycling time for sending messages depends on a number of messages that are in the buffer and

[1] To recall, the speed of a LAN was declared as 10Mbit per second. All messages except PRT files were less than 0.5Kb (the size of the PRT files were between 1-2Kb). This means that in theory a time for data transfer would be in the range of 0.05-0.2 milliseconds. However, due to activities required for establishing a connection between a source and a destination workstation this time could be in the range of a second or two. It should be noted that this will reduce the amount of speeding up possible for faster than real time simulation and does represent a potential limitation on the system but since most operations in the real world take several minutes a few seconds for communication is not significant in the normal operation of the system for shop floor control.

the time that is needed for each message to be transmitted. It means that the cycling time can vary significantly. On the other hand, each message has a field called "MsgTimeToLive" (see Section 10.2.2.2.1 "A Task Announcement message"), which represents the number of seconds after which the message will be deleted from the buffer (in this way the buffer was kept clean from the obsolete messages). In the case when there are a lot of messages in the buffer this time can elapse and the message will be deleted before it gets second chance to be retransmitted. By increasing the time for "MsgTimeToLive", messages in the buffer could pile up quickly if the first attempt for sending failed and the cycling time would be increased even further.

The problem was solved by replacing a message "time dependency" with a number of allowed attempts for a message to be sent. Therefore, the "number of seconds" in the field "MsgTimeToLive" was replaced with the "number of attempts allowed" for a message to be sent before it is removed from the buffer and the program logic was modified accordingly. Each unsuccessful attempt decreases this number by one. If a message is not send after predefined number of attempts, the negotiation process for a work-part under the question will not be accomplished and a workstation agent would consider that the negotiation process was unsuccessful. Finally, the message will be deleted from a buffer keeping the buffer clean of the obsolete messages.

The second issue was much more complex and, as noted, was not resolved or more correctly, was not "fixed-up" (even though the solution is known) in the current version of the application. There were situations[1] when two or more workstation agents wanted to transmit their messages to the one destination workstation at the same time. Theoretically, a data collision on the network medium would occur at that time, and only one message is supposed to be transferred while the others would fail. For retransmission of failed messages to the attendant recipients a network protocol at Data Link layer of ISO-OSI reference model should be in charge. However, sometimes all messages are sent successfully, another time just two, yet another time only one! Since workstation agent's communication modules were based on the use of third party ActiveX controls (IPDaemon and IPPort – see Section 10.2.1.2 "The communication module"), and because network debugging tools were not available, what actually happened inside these components and inside the communication software (TCP/IP protocol suite) that was responsible for reliable communication on a LAN (that is how they handle this situation when a collision on the network occurred) has remained mystery.

It should be noted that this problem could be easily overcome (but not solved) by "insisting" that each message has to be acknowledged before it is considered that it had been successfully transmitted. That is, the message could be deleted from the buffer for sending messages only when the message is acknowledged. Otherwise, if the message is not acknowledged after a predefined time, the "number of allowed attempts" would be decreased by one. This would be a "patch" for this problem and for any other problem caused by uncertainty that may occur during network transmission. Notwithstanding this solution, appropriate attention should be paid to fully understand the cause of the problem, and consequently to apply adequate methods and techniques to resolve it, particularly if the application is to be employed in real world industrial environments.

In the end, it should be said that for developing network applications, a set of good network debugging tools is something that is highly advisable to have at ones disposal. Without good network debugging tools, the program debugging was very hard, rather painstaking, and in some cases impossible. All anomalies regarding network communication were identified by observing

[1] During testing, it was (deliberately) arranged that three different workstations sent three "task announcement" messages to the one destination workstation at the same time.

carefully the flow of messages over a LAN - as much as it was possible with a naked eye – and by creating "traps" in the source code in each of the workstation agents.

Though measurement of the communication related parameters has not yet been completed, the initial experience with the application suggests that the system does not require excessive communications overhead. The system uses very short, structured messages with a standard format (see Section 10.2.2.2 "The structure of messages). From the experimental arrangement that was set up, it seemed that short messages of a size less than 1Kb (excluding PRT files) over the fast Ethernet LAN with transfer rates of 10Mbit/sec posed no problems for a communication system. (Nowadays, 100Mbit/sec Tbase Ethernet network is a common, "standard" installation). However, for reasons that are discussed later in Section 14.1 "Measuring the performance of agent based heterarchical control systems", measurement of the communication performance parameters needs to be undertaken. In fact, this is something that has to be done before continuing any further work on the development of the heterarchical SFC system.

Finally, when considering a possible implementation of the proposed heterarchical shop floor control system in a fully automated, real job shop manufacturing facility, it should be noted that there are two possible ways of doing that.

In the first method, the application could be accommodated within the existing workstation control systems to be used independently of, but in parallel and simultaneously with the existing control programmes. There are two advantages of this approach. Firstly, the application would be used to facilitate real-time scheduling (routing) of work-parts through the system. For predefined criteria, the workstation agents would select the most suitable workstations to execute the next processing tasks, and most importantly, this would be done on-line, during the production of work-parts, as discussed in Chapter 11.

Secondly, by using agents independently of the actual workstation controllers, the manufacturing system would be equipped with uniform, standardised components such as common workstation agents' local databases. This means that the agents can be interrogated at any time by a central monitoring system for information on progress and the current status of the whole job shop. On the other hand, if the controllers that come with each machine are used it is almost certain that they would have had incompatible databases and conversion routines would have to be written. Also, this consideration applies for manually operated and semi-automated systems, and it would be very beneficial for these systems as well.

In the second approach, the workstation agent could be integrated with the other existing control systems (at the first place it is thought on the systems inside a workstation such as the machine's, or the robot's, control units). As such, the workstation agent would take a role of the overall workstation's controller. This approach requires that the control module of a workstation agent be based on the concept of the Virtual Manufacturing Devices – VMDs. VMDs act as interfaces between the workstation's controller and the other subordinate control units that are found inside a workstation. They tackle issues of incompatibility that exists between different communication protocols and database formats. In this way, VMDs decouple and encapsulate the workstation agent software and provide its independence from the other versatile control units used for controlling different equipment in the workstations. This approach would require significant work on further development of the current workstation agents (more details are provided in Section 14.3.7 "Development of a real workstation").

Note that the heterarchical system of distributed computers placed very little load on the individual processors. This means that in many cases the agent software could be operated on the machine controllers where, as occurs in modern systems, the controller is based around a compatible PC.

12.1.12 Similarities with the holonic approach

Some similarities can be drawn between a holonic manufacturing system and the system presented in this research study. Following a set of definitions related to holonic systems [IMS international, web] several similarities are indicated as follows:

- A workstation agent can be regarded as a production holon's information processing part. Also, since workstation agents map directly production workstations on a shop floor, a workstation agent can be considered as is consisted of an information processing part and a physical processing part – like a holon.
- In the holonic approach, holons form a holarchy to produce a work-part. Similarly, in the presented approach, workstation agents also create a "holarchy" to produce a work-part. A "recipe" for creating a holarchy is contained in the PRT file of a corresponding work-part. Namely, with defined alternative workstations the PRT defines holarchy, in its broader sense. Exact holarchy, is not known in advance and it is determined only after completion of work-part – holarchy is established on-line, during the production process and it is defined by the actual physical route which work-part takes through the system. Workstations can be part of several different holarchies simultaneously but during the work-part fabrication each workstation belongs exclusively to only one holarchy. After the work-part is processed on the workstation, the workstation is free to be engaged in other holarchies.
- Like a holon, a workstation agent has the same minimum set of attributes: autonomy and cooperativeness.

In summary, there are no conceptual differences between a workstation agent and a holon.

12.1.13 Some advantages of the proposed systems in comparison with traditional shop floor control systems

By applying a distributed heterarchical scheduling method, the following issues related to the scheduling manufacturing systems are successfully addressed:

- ***Size and complexity of the material and capacity planning system.*** In a proposed heterarchical shop floor control system there is no need for a detailed material and capacity planning system that is otherwise normally present at the shop floor level. A general estimate of the requirements can be obtained from the task and processing flow data used to generate the PRTs. The elimination of the need for a detailed system is achieved by:
  - The design of a shop floor control system that allows a significant reduction in the amount of data that would otherwise be required to be transferred between the shop floor control module and the other modules of a production planning and control (PPC) system. In traditional MRP central systems, shop orders are tracked as they progress through the factory by processing detailed transactions of work at every workstation. However, because of existence of local, autonomous, "intelligent" workstation agents that resolve many resource sharing and control problems at a local workstation level, the approach used here significantly reduces the volume of these shop transactions. The amount of information and the size of the messages or data files such as the feedback reports for tracking orders and monitoring workstation activities, therefore, can be appreciably reduced.

- Reducing or totally eliminating the need for detailed input/output analysis as is performed in traditional MRP systems. Scheduling jobs among workstations is performed in such a manner that conventional input/output analyses required for scheduling purposes are simply not needed anymore. The approach used also reduces the need for shop loading reports to spot potential problems (indicating bottleneck workstations). Bottleneck situations will be immediately reported as they appear in the system. For example, if some of the workstations have utilisation rates above the allowed level (during some predefined time periods, for example, a week or a month) then this situation will be automatically reported by a corresponding workstation agent to the higher control levels of the manufacturing system (for example to the overall system supervisor/manager) indicating the places in the system that require further capacity consideration. Also, if the workstation breaks down, the workstation agent would immediately report this failure indicating that the work-parts inside the workstation could not be completed and that central intervention may be required. Finally, if any of the workstations "notice" that some other workstation does not respond for a long time (meaning that the workstation agent is not operational) this anomaly will be also reported indicating that some corrective actions should take place.
- Reducing the size of the central manufacturing database, (by implementing distributed workstation local databases).

The system also reduces the number of people otherwise required for conducting production planning and scheduling functions.

- ***A problem of partial information***. The scheduling process can proceed even in the case when information from some workstations is not available or if their availability is appreciably deteriorated for any reason. The scheduling process is performed in a reactive manner (see Section 10.3.1 "Global scheduling - scheduling among workstation agents") without the need for having complete data on the loading of all workstations in the system. In addition, issues regarding scheduling of urgent work-parts can be solved on-line, without the need for extra time to tune or adjust the complete schedule as occurs in centralised systems.
- ***A problem of uncertainty***. The general approach to uncertainty in traditional systems is to react and reschedule. However, by understanding the types of uncertainty and how they are developed, the logic for their handling can be incorporated into the control mechanism. A proposed distributed heterarchical shop floor control system decomposes the problem of uncertainty into two levels: shop floor level and workstation level.

  At the shop floor level, distributed scheduling logic provides a mechanism to successfully cope with the worst possible scenario: the case when one or more workstations in the system is completely disabled and out of order. Due to the overall system design that provides the appropriate level of redundancy for performing workstations operations (usually a few workstations are capable of performing any given processing task) and since the bidding system chooses the "best" bid at the time when the next operation on a work-part is about to commence, the system adapts automatically to the changed circumstances on the shop floor such as a complete failure of a workstation except for the case when there is a complete failure of a unique workstation without an alternative. However, if a complete failure of a unique system workstation occurs (a workstation that is defined as the only one capable of performing a processing task in the PRT) then (most probably, except if the failure is promptly fixed) all the work-parts that are produced on

this unique workstation will be delayed and their completeness will be postponed. To cope with this situation, particularly if

- a unique workstation breaks down frequently,
- the work-parts that are produced by a unique workstation are of great "importance" for a company (for example, a capital work-part that cannot experience extensive delays), and
- a high probability exists that these work-parts will be produced again in the future with a stable demand,

a plan for ensuring the appropriate level of redundancy must be decided and adopted at the higher management levels. In essence there are two possibilities for making this more strategic level decision: either to procure new machinery, or to find an alternative for execution of these critical manufacturing operations by contracting other nearby companies that have surplus capacities. As indicated above, in the proposed heterarchical SFC system it was understood that an effective reporting system needs to be employed to alert the overall system supervisor that central intervention is required each time when unpredicted (complete or partial) workstation failures occur in the system.

At the workstation level, it is envisaged that the workstation control system provides a mechanism for recovering from unpredictable events that can occur inside the workstations. Not all events will have the same effect and, therefore, many of them can be accommodated (for instance, if a predefined cutting tool cannot be used as it is broken, by using another alternative tool, which might have worse cutting characteristics, the machine still can be operational). A workstation can continue to operate, presumably in a degraded state but can still be functional. The workstation agent should take new circumstances into account when generating new bids during negotiation processes. If the workstation cannot be restored to be fully operational within a predefined time, the entire workstation might be considered as out of order and maintenance and repair would be required.

The key concept for the scheduling framework is based on the assumption that it is possible to create adaptive heuristics for dealing with uncertainty. That is, there is no question "if something happens" (for example, it is quite "normal" to expect that the entire workstation will be out of order many times during its production life) but "when something happens" how should the system respond. Different heuristics may need to be developed, depending on the type of expected events and the consequences that such events might have on the entire system, but the bottom line is that the process of work-part fabrication in the proposed system will normally be less sensitive to any perturbation.

- ***A problem of temporality***. By defining and choosing different criteria for selecting the most appropriate workstation for performing the next processing task, the scheduling process can be adjusted easily to the latest production objectives, which in turn can be changed over time. For example, in one instance the applied criterion for selecting a workstation could be to select a workstation that is capable of accomplishing the next processing task in the fastest manner, for example, when there is time pressure to complete the overall job because some processes have taken longer than expected creating concern about meeting the due date. Next time the appropriate criterion could be to select the workstation that is capable to do a given task in the most economical way, for example when there is little time pressure but the cost of production varies considerably depending on which machines are used.

- ***A forecasting problem.*** It is possible to adjust the short-term planning horizon by changing the time when the negotiation process starts. Namely, by changing the time when information for making decisions about the next workstation is required (the time of releasing a task announcement) it is possible to change the length of a "time frame" (in a literature also referred as a "time window") inside which the estimations about the workstation operational states will be made. Boundaries of a "time frame" are defined by start and end points. The start point of a "time frame" is determined by the moment when a source workstation sends a task announcement to the destination workstation (if we are not going to take into account the time that is needed for conducting a negotiation process, then the start point of a "time-frame" is determined by the moment when an acknowledgment message is received back from a destination workstation, indicating that the negotiation process is finished). The end point of a "time frame" is determined by the moment when the machining of the work-part is expected to be accomplished on the destination workstation. To minimise a forecasting problem, the intention is to make a planning horizon (a "time frame") as short as possible so that the validity of information about estimated states on a destination workstation is preserved. In such a way, the decisions that are made would be less reliant on predictions whether the planned activates will or will not be performed according to the plan on a destination workstation (including a transport means for delivering a work-part from the source to the destination workstation).
- ***Problem of computational stability.*** As indicated in Section 6.3.6.2.3. "System of autonomous agents can become computationally unstable", when many workstations in the system malfunction almost simultaneously, a heterarchically-controlled system may become computationally unstable and the network could become quickly overloaded. To avoid such extreme situations, appropriate negotiation algorithms should be envisaged. For example, the current program realisations, allows only two negotiation cycles per negotiation process. If both are unsuccessful, a system management function/agent will be notified that a workstation encountered some problems in allocating the workstation for executing the next processing task. Also, each message has a "TimeToLive "field which defines how many times a workstation communication module will try to send a message. After each unsuccessful attempt this value decreases and eventually, when equals zero, the message is deleted from the buffer for sending messages – avoiding an unnecessary load on the network. At this time the overall system manager would normally be alerted to this situation. Finally, due attention should be paid to the design of the system and the decisions as to which system's entities are to be mapped as agents. This is why a minimalistic approach was adopted in the first phase of this project – the system consists of workstation agents only. This matter needs to be addressed in more detail and investigated as a part of the future work on the project.

12.2 Contribution of the study

The main contributions of the study are:

- The overall design and development of the core part of the heterarchical shop floor control system in which the operation of workstation agents, as the only "intelligent" elements of the system, is "coordinated" on the basis of information contained in the processing route tables (PRTs). PRT files are crucial for operation of the system. Each work-part must have its corresponding PRT that contains processing information ("DNA code") of how to fabricate the work-part. PRT files contain processing information that is work-parts specific. Other processing data related to the machining work-parts, which

are workstation specific (such as CNC and robot programs, data about the cutting tools and fixtures required, and so forth), are located in local workstation databases.

- The way in which PRT files are used to link process planning and scheduling functions. PRT may contain several alternative process plans, which in turn can have a number of processing routes. Processing plans are determined "off-line" (as part of process planning function), however, a decision about the actual processing route that it will be taken by a work-part (scheduling function) will depend on the current shop floor circumstances and it will be made "on-line" during the fabrication process of a work-part.
- The development of a valuable research tool for conducting a wide range of experiments in the domain of distributed heterarchical manufacturing control systems. The software developed offers an "open system architecture" which can be upgraded at a later stage with new features, making ongoing improvements over time. The system can be modified to meet a number of research requirements such as experimenting with different negotiation algorithms, implementing new negotiation protocols, developing new agents, expanding the system on the Internet, etc.
- Developing a system that is feasible, practical and affordable, particularly for small size job shop manufacturing systems. As such, the system has considerable commercial potential. At the current stage of a system development, processing and material handling operations that are conducted inside a workstation are simulated. However, the core part of heterarchical control system itself, has already been, and could be further, developed so that it interacts with real manufacturing systems. At this stage, instead of responding to input/output signals from a real world manufacturing factory, the control system produces responses to software events that are triggered when certain simulated activities happen inside the workstations. In a real world application of the system, these simulated events would be replaced by the actual operational events generated by the production machinery.

Some other points of importance are listed below:

- The proposed structure of manufacturing system model. The manufacturing system could consist of a few manufacturing cells (for example, for the production of prismatic, shaft, and disc shaped work-parts) each comprising of a large number of workstations (basic autonomous production units) that mutually co-operate in the course of work-part production.
- The way in which the decomposition of the shop floor control system is performed. The system is comprised of workstation agents that are autonomous control units that have complete control over planning, scheduling, controlling, and monitoring manufacturing processes inside their workstations.
- The proposed design and architecture of workstation agents. The modular design of the workstation agent software closely follows the agent's functional model. The main agent's functional units such as: a unit for communication, a unit for creating and interpreting messages, a unit for storage of the workstation's local data, and a unit for control and scheduling of manufacturing operations, are encapsulated into the software modules. The program modules are clearly distinguished and all of them (except the module for control and scheduling) can be used in real, agent-based heterarchical shop floor control systems. Note that the main role of the control and scheduling module is to simulate manufacturing activities inside the workstation. A user interface and a mechanism for triggering events when some activities occur in the workstation (for

example, an event when a work-part enters the workstation or when a machine finishes a work-part fabrication) are intrinsic parts of this module.

12.3 Summary of the main achievements

A list of some of the key project achievements is presented below.

- ***Established the target manufacturing organisation and environment.*** The research project was oriented towards general purpose, random job shop manufacturing systems, which produce work-parts in small quantities and broad variety, and which operate in an unstable production environment (the time of arrival of incoming orders of different sizes and product types in the manufacturing system cannot be foreseen).
- ***Defined the model of job shop manufacturing systems.*** The model was based on the concept of workstations which were considered as basic autonomous production units of the manufacturing system.
- ***Defined the model of heterarchical shop floor control system.*** On the basis of a literature review, a model of heterarchical shop floor system in which the workstation agents are the only "intelligent" elements of the system was defined and designed from the scratch. Work-parts, tools, jig and fixtures remain "dumb" elements of the system. It was shown that this model is feasible and practical. Note that the previous research work in the area of distributed heterarchical shop floor control systems did not exist at Massey University. The project was established and started from the "blank piece of paper" as a pioneering work in this domain.
- ***Selected the methodology for resolving complex scheduling problems.*** In this study, the scheduling of work-parts in a job shop manufacturing system is viewed from a global and local perspective. Scheduling of work-parts among workstations (global scheduling) is based on the principles of distributed, reactive scheduling with a variable scheduling horizon[1], and the use of a negotiation mechanism and an auctioning technique. Scheduling of work-parts inside workstations (local scheduling) is based on the work-part priority indexes and can be approached by using any of the traditional scheduling techniques. In both cases, scheduling is performed in real-time and on-line during the system operation.
- ***Defined the data required and their organisation in the proposed model of heterarchical shop floor control system.*** The main data repositories are Processing Route Tables (PRTs), which are defined for each work-part to be produced by the system, and workstation agent local database, which is defined for each workstation located in the system.
- ***Defined the information flows in the system.*** In the adopted model of the SFC system there are two major flows of information: Among workstation agents and inside a workstation agent.
- ***Design of Processing Route Tables (PRTs.).*** PRTs are selected as a mechanism for expressing alternative processing plans that can be selected/chosen on-line, during the fabrication of processing parts, considering the current circumstances on the shop floor.

[1] With a "variable scheduling horizon" the system can make a reservation of workstation resources by considering overall manufacturing system circumstances that will occur at a different point in time in the near future. This point is determined by the time that takes one or more processing tasks to be accomplished (this is why it is variable) including also the time for finishing the current task.

The particular design of PRTs is of crucial importance for operation of the system developed in this project since it contains key information for conducting negotiation processes among workstation agents.

- ***Design of the structure of a workstation agent.*** The workstation agent has a modular structure composed of the four main parts: a local knowledge database, a communication module, a decision making module, and a workstation control module.
- ***Design of the inter-agent message structure.*** Communication among agents can be conducted by using rudimentary principles extracted from the Contract-Net Protocol (CNP). Namely, without applying any of propriety protocols, it is possible to develop a rudimentary negotiation mechanism that can be extended (as far as is needed and required) at later stage. By carefully devising a set of messages that are to be exchanged among workstation agents, and by carefully contriving a negotiation "flow", the negotiation mechanism used in this project can cope with communication failure, which is treated at the same way as one of the workstation agents becoming unavailable in the system.
- ***Selected the development environment.*** To develop the prototype software, the following elements were identified: computer hardware, software development tools, computer network, and communication protocols.
- ***Design and developed the workstation agent database.*** As a part of workstation agent, a small relational database was designed and developed in MS Access
- ***Designed and developed the software for interaction and control.*** One of the key objectives was the development of a simulation and modelling system. The system was designed so that it could be easily adapted to a real world manufacturing system.
- ***Assembled and configured the IBM compatible personal computers***. To create an experimental test-bed system eleven computers were assembled.
- ***Created a small Local Area Network (LAN).***
- ***Demonstrated that it was possible to develop a distributed heterarchical shop floor control system using simple PCs.*** The operation of the working system was demonstrated on the test-bed created.

Personal Achievements:

- Obtained basic knowledge in the area of design and development relational database systems.
- Developed programming skills using the Visual Basic language.
- Obtained sound knowledge about computer network systems and associated communication protocols and practical experience in assembling and setting up peer-to-peer LANs.

Note that the final workstation agent program involved 8000+ lines of code (796Kb of executable program) and the local workstation agent database comprised 332 lines of code and 563Kb of executable program (without data).

The next chapter provides some concluding comments about the project and gives a summary of the main characteristics of the model of distributed heterarchical shop floor control system developed in this project.

Chapter 13. Conclusion

The project suggests that development of an operational distributed, heterarchical shop floor control (SFC) system using commonly available computer hardware and software is both a feasible and an affordable undertaking.

Workstation agents are used to resolve complex and difficult scheduling and resource allocation problems, on-line and in real-time.

The whole system is based on a network of autonomous[1] but cooperative workstation agents that replace both, a centralised and hierarchical shop floor control architecture and a centralised database used as a central repository for manufacturing data related to the workstation subsystems. In comparison with the traditional centralised and hierarchical control systems this approach is more flexible and less expensive to develop and maintain. Also, it is more adaptable, reliable, and fault-tolerant and the system is easier to extend and reconfigure.

The system is applicable to an entire range of job shop manufacturing systems, from manually operated systems to the fully automated computer controlled systems. However, these statements have not been put to a full experimental test at this stage.

By creating a test-bed application, the proposed distributed heterarchical SFC system moved from the stage of an architectural description and a theoretical analysis (a modelled application) to the point where it was demonstrated in a simulated environment (an emulated application). Although the system is still in its research phase, the application provided a fully functional and operational model of the system. It demonstrates how a core part of the heterarchical shop floor control system model operates and it is evidence that the proposed concept is both feasible and practical.

The produced application demonstrated that the development and implementation of a distributed, heterarchical, multi-agent shop floor control (SFC) system should not be an expensive project. No special hardware or software requirements were imposed to achieve an effective, practical operational control system. The system can be implemented on relatively low performance personal computers connected into an inexpensive Local Area Network (LAN). Also, the system development does not require any special purpose high level programming languages specifically designed for developing agents (supported by compilers, libraries and debugging tools etc.) or special computer operating systems. It uses standard programming languages, Windows® operating systems and TCP/IP network systems.

Since there is no difference between the simulated and the real manufacturing environment (from a shop floor control perspective) in the test-bed application, it can be used as a valuable research tool for conducting a set of very high quality experiments in the area of heterarchical manufacturing control.

The network test-bed application has been used for testing and validation of the proposed control structure and for demonstrating how the system functions, but it can be readily adjusted to serve

[1] Each workstation agent in this model has a local view of its environment and the ability and authority to respond locally to that environment.

as a tool for solving resource allocation problems in actual manufacturing systems. However, to use the application for conducting experiments in a real, prototype manufacturing environment, the application requires that some software improvements be made first.

Because the time available for the project was restricted the research was limited to the production of the simulation system and a demonstration that it worked. If more resources and time had been available the system that was developed would have provided a very useful platform for the simulation and measurement of the parameters of the operation of a full factory system. However, sufficient runs of the system were completed to demonstrate that there were no significant difficulties in the application of the system in practice.

Many prominent researchers believe that manufacturing control systems that are based on the principles of a heterarchical control are the future of manufacturing control [Baker, 1998, Brussel et al, 1999, Bussmann & Schild, 2000, Duffie & Prabhu, 1994, Maturana et al, 1999, Parunak, 2000b, Shen & Norrie, 1999, Sikora & Shaw, 1997, Sousa et al, 1999, Veeramani & Wang, 1997, Wang & Usher, 2002]. The author shares their opinion and expectations. It is considered that this is particularly true for large scale job shop manufacturing systems (with a large number of machine tools) that are characterised with production in which work-parts are fabricated in a great variety and in small quantities (low volume - high variety production), and in which production orders of different types and sizes arrive in the system in random intervals that cannot be predicted in advance. For solving scheduling and control problems in such manufacturing systems the proposed scheduling and controlled system should perform better than any other conventional control system in use nowadays.

It should be noted that a significant amount of research (including human and material resources) has been conducted in the area of holonic manufacturing systems. These systems include many attributes of heterarchical control, such as autonomy, cooperation, modifiability, etc. and can operate in either hierarchical or heterarchical manner. Holonic manufacturing systems represent a more advanced but also more complex and more expensive form of control than the system presented in this thesis.

Although much work remains to be done it is believed that this research will 1) promote discussion and stimulate further research in the area of the heterarchical shop floor control systems, and 2) assist the research community to get one step closer to the one of the most challenging visions for manufacturing – to create a self-configuring manufacturing system

More than half of the cost associated with computer controlled manufacturing equipment is in the preparation and maintenance of its software. One of the visions advocated by manufacturing strategist is to get a machine tool from a vendor that can understand its own capabilities. Once such a machine is placed on the shop floor and plugged into a multiplexed source of power and communications, the only configuration necessary is to key in its physical position (location and orientation). A machine would be able to identify other machines in the shop by itself (rather in the fashion of the "plug and play" approach in PC configuration) and together with them determine how to execute the various work orders that appear on the network.

Also, regardless of the fact that the final software of the demonstration system is not structured at this stage for efficiency and reliability, it is clear, however, that the software works and could be developed into a commercially valuable product. It is with considerable regret on the part of the author that this has not been able to be undertaken because of lack of funding and commercial sponsors in New Zealand for such a system.

The summary of the main characteristics of the proposed model of distributed heterarchical shop floor control system and the application (workstation agent software) developed to demonstrate operation of the system is given in the following two subsections.

13.1 Characteristics of the heterarchical shop floor control system model

- A core part of the overall heterarchical shop floor control system has been designed, developed, and created in the form of a test-bed application. The system was successfully tested on ten IBM-PC compatible computers connected in a Local Area Network (LAN).
- Although the core part of application was tested on 10 computers only (eight production workstations, an input system workstation, and an output system workstation) the system was clearly not limited to this number of computers (workstation agents).
- The system is made fully operational by using only workstation agents as the "intelligent" elements. Work-parts remain "dumb" elements (without any need for "artificial intelligence" - built in microprocessors) of a manufacturing facility.
- Agents are common abstraction for all workstations on the shop floor. The control system allows simultaneous existence of both hierarchical and centralised workstation control architectures in the system. From the shop floor control point of view they are all seen in the same way.
- The simulation heterarchical shop floor control model is well matched to the actual situation in a manufacturing system - each workstation agent is part of the workstation itself and not part of any central system that may become disconnected from the workstation. As a result the workstation's computational state tracks the state of the processes very closely and is independent of the other workstations or any central planning system.
- Scheduling of work-parts through the manufacturing system and inside workstations is performed in the real time, on-line, while the system is running, not in advance, off-line, prior to manufacturing activities as is the case with centralised planning and scheduling.
- The system is capable of coping with unexpected machine failures without rescheduling. Scheduling of work-parts among workstations is performed dynamically and in a real time on the "fly" (on the principles of reactive scheduling where the decision for selecting the most appropriate workstation to perform the next processing operation (task) is made taking into account only the information that reflects the current state of the workstations that are capable of performing this task, and not the state that is expected to occur some time in the future) so there is no need for rescheduling. Workstations that are not operational at the time when a decision about the next, most suitable workstation for performing the next processing task on a work-part is made, simply do not participate in the negotiation process.
- Work-parts can have different priority levels.
- A work-part will be fabricated by the system as long as at least one possible processing route for its production exists.
- External to the workstations parts are pushed through the system, while inside the workstations parts are pulled towards the machines.
- Workstation agents are generic and can be developed to support manual, semi-automated and fully automated systems.

- The application uses a bidding scheme where bids are initiated by the workstation. Workstations generate bids and compete with each other to get a work-part for processing.
- There is no need for a central database other than as a back-up for each workstation since each workstation agent maintains its own local database, which is likely to reflect the current state of the workstation more closely than a central database which may be isolated from the workstation by communication or processing problems. With modern personal computers it should be possible even to provide a regular up-to-date back up at the workstation itself reducing the time to recover from a local workstation failure.
- The system is robust to change and easily reconfigurable. The system readjusts itself automatically to the removal or addition of agents (though IP addresses of new agents need to be recognised by each agent in the system).
- The workstation agents can be added and subtracted from the system while it is running.
- Multi-agent architectures are inherently scaleable and modular. The capacity of the control system will adapt to the changes in the configuration of the manufacturing system.
- A system with autonomously functioning components did not collapse when one or more of the components were made to fail or malfunction.
- The overall system behaviour emerges from local decisions made at the workstation level and not from potentially isolated central planning systems.
- The current application with the minor changes could be employed in manufacturing systems comprising workstations that are under a human operator supervision or control (manual and semi-automated workstations).
- Processing tasks can be assigned to alternative workstations in advance of orders being known. However, specific tasks (operations) are dynamically assigned to a particular workstation when the task is due to be undertaken. Such a system is always better than any alternative in which tasks (operations) are "firmly" assigned to the workstations ahead of the actual processing by a centralised planning function.
- With a priority assigned to individual parts, the way in which tasks are assigned to the workstations may affect the time required to perform the individual processing operations. For example, in a case when an urgent part (which has a higher priority than any other part that has already been allocated to the workstation) arrives on the workstation, the estimated start times for the machining of all other parts waiting in the input buffer will be shifted back for the time that it will take for the urgent part to be processed.
- Workstation set up times are included in the process of forming bids. In most manufacturing systems, even those with flexible machining centres, significant set up times are required to prepare machines for production of different work-parts. For heterarchical control to be a valid alternative to conventional shop floor control architectures the issue of resource reconfiguration must be properly addressed. This is why empirical values of set up times are included in the bid creation process as developed.
- The PRT (processing route table) for a work-part could provide a mechanism for tracing a work-part that has been found to be defective either in final inspection or after delivery. This would enable corrective action to be taken to eliminate the problem area.

13.2 The software related characteristics

- Any workstation agent can activate a number of simultaneous work-part negotiation processes.
- Each production workstation agent is essentially a replicate of the same workstation level software.
- The software for each agent is short, simple, and as a result is easy to write, debug, and maintain (compared, for example, with the software used in centralised control architectures).
- Co-operation among workstation agents is accomplished by using a negotiation protocol, which is based on the contract net protocol, as an application layer protocol of the ISO/OSI reference model.
- Workstations agents communicate with each other using the TCP/IP protocol suite
- Each workstation agent is able to maintain up to 1,000 simultaneous connections.
- Messages have a transient character. They do not persist after they have been sent.
- Messages are only available to agents who are listening when they are sent.
- Messages are specifically addressed to workstations capable of carrying out the next process and not broadcast to all workstations. The sender is required to know the address of the agent to which the message is going to be sent.
- Processing and material handling activities inside workstation are simulated. The level of automation implemented in the individual workstations is not of primary importance for the functioning of the application. The application is intended to support any operating mode: manual, semi-automated, and fully automated.
- Simulation of workstation activities is an event driven process with a built in degree of randomisation. Unexpected variations in the times used in the model are allowed and simulation demonstrates the ability of the system to cope with random, unexpected variation.
- Location of the parts inside the workstation is known at any time during the simulation process.
- The multi agent application was developed using Visual Basic 5.0.
- The local databases of the workstation agents were developed in MS Access'97.

13.3 Concluding comments on project goals and objectives

All but one research goals set in Section 7.5 "The research goals" have been achieved. The model of agent-based heterarchical shop floor control system was designed and the accompanying software application was developed. Also, it was demonstrated that it was possible to develop a practical and inexpensive system using commonly available technology at the time when the project commenced. However, the project did not manage to attract the interest of the local research community, or industry. Consequently the subsequent phases of the project were not initiated and the further work on the project was abandoned. (The goal to initiate further work towards producing a commercial system was not achieved because the

group interested in this area of research was disbanded during a reorganisation of the Faculty of Technology at Massey University).

The remaining project objectives set in Section 7.6 "The research objectives" were all accomplished. Section 12.1 "A retrospective view of the research objectives" provides a brief summary on how these objectives were achieved and Section 12.3. "Summary of the main achievements" provides a list of the main achievements in the projects that are again closely related to the project objectives.

13.4 Summary

In summary, the project demonstrated that is was possible to build a practical model of an agent based, heterarchical shop floor control system using low cost computers and commonly available software components. The majority of the project goals and all of the project objectives were achieved.

The next chapter provides some guidelines for the possible future extensions to the project.

Chapter 14. Future work

The objective of the project was to demonstrate that a heterarchical shop floor control system based on multi agent paradigm could be developed by using commonly available computer hardware and software. This was achieved but there are still many opportunities for enhancing the developed system and for conducting research in the area of heterarchical manufacturing control systems in general, some of which are suggested in this chapter. The first and immediate progression is the measurement of the performance of the existing system under a variety of conditions. It is recognised that the network may prove to be a limitation in large manufacturing systems and it would be important to establish if this is so. The next development would seem to be the interfacing of the system with a real world workstation including one or more Computer Numerically Controlled (CNC) machines. The remainder of the network could be simulated workstations as in the model developed in this project. Following this a rewrite of the software into a better structure and using the extensive developments that have occurred in the various aspects of real time programme development systems, networking and simulation would be important. For simulated performance testing and measurement, random event generating capability and data recording facilities would need to be added. The model system would also provide a useful trial system to establish time lines and to plan the overall shop floor system including the appropriate level of redundancy to be provided etc. development. It would be possible to extend the agent based approach to other parts of the system such as AGV's tools etc. Another possible development would be the integration of an overall supervisory system to monitor the operation the shop floor on a real time basis.

14.1 Measuring the performance of agent based heterarchical control systems

An important first step for future work would be to measure the performance of the model system developed in this project under a variety of conditions. It is envisaged that the test bed application could be used to conduct a broad range of experiments to measure the performance of heterarchical shop floor control systems using the relatively simple agents developed in this project. Using a model of the manufacturing system and an integrated simulation approach, various agent based scheduling algorithms can be tested and evaluated. From the understanding of the behaviour of the basic algorithms under various operating conditions more complex algorithms could be developed and adopted.

The shop floor control software used in the model is, in the key points, nearly identical to the software that would be used in real system. Namely, each workstation in the modelled system interacts with the other workstations in the same distributed, heterarchical manner as it would in the real system. This allows the measurement of performance of the control system with considerable accuracy.

Measuring manufacturing systems performance should be undertaken under different loads and product mixes. In evaluating the performance of a manufacturing control system, several dimensions are important. The first relates to the way in which orders (jobs) are handled. Some of the parameters that can be measured are: the time a job takes to go through the system (referred to as a lead time or a flow time), the number of jobs in process, the delay/hurry in

completing an order (tardiness/earliness), the maximum rate at which orders can be accepted (the system throughput), or waiting times in workstation's buffers. Another dimension relates to the utilisation of resources (machine utilisation) and associated costs. However, any other parameters that might be of interest for measuring effectiveness and efficiency of a heterarchical shop floor control system may be included.

One of the ways to compare the global efficiency of the manufacturing system developed in this project with the other systems with different structures and control schemes (but with the essentially equivalent production capacities) would be by comparing the amount of queuing created during the manufacture of the same set of parts. Queues will develop in any manufacturing system because of limited resource (equipment and human) capacities. Different structures and different control policies result in different amounts of queuing. Having fixed mean but randomly distributed arrival time intervals of jobs, means that systems can be compared by determining either the average queuing times or the average number of jobs waiting in queues. Because the queuing measures are long-run averages, the time horizon of the simulations needs to be set over a long period; several tens of hours at least.

Also, part of the future work would be to test this system with real world parts manufacture schedules, randomly simulated machines failures and regular part changes in a highly variable sense. It is believed that the produced system would be able to cope with the frequent disturbances on a shop floor better than any other control systems in which control and scheduling functions are based on the traditional MRP/MRPII (Materials Requirements Planning/Manufacturing Resources Planning) principles used in job shop manufacturing systems nowadays and the system could be used to demonstrate the validity of this belief.

Also the system could be used to assess the adequacy of redundancy provisions and the likelihood of existing overall equipment meeting the requirements of manufacturing arrangements without serious bottlenecks or extensive delays. In this case the system could be used as a valuable facility planning tool.

14.2 Measurement of the limitations placed on the system due to the performance of the communication network

Measuring system performances from the communication point of view is another interesting set of experiments that should be conducted. The system developed here uses short preformatted messages and there is very little need for significant information transfer between workstations since each workstation maintains its own database. Therefore, in the experimental system the bandwidth of the LAN was not critical issue. However, in a case of large manufacturing systems that might be comprised of several hundreds of workstations and when a heterarchical SFC control system in such manufacturing systems would include other negotiation process (in addition to work-parts, negotiation processes could be conducted for shared resources such as cutting tools, fixtures, jigs, and AGV's or other transport means), then there is real threat that system performances may degrade due to possible saturations of the network. Some researchers believe that the amount of information that needs to be interchanged between distributed processing entities, which can vary significantly over time, would drown the system performance. For this reason, for large manufacturing systems, measuring communication performances is one of the very important tasks

Also, it would be appropriate to investigate the behaviour of the system when the Ethernet network, which is based on the Carrier Sense Multiple Access with Collision Detection

(CSMA/CD) method, would be replaced with the Token Ring network, which is based on the more "time deterministic" token passing method.

14.3 Possible project extensions

In the future, the project could be extended in several directions as suggested in this section.

14.3.1 Development of software modules as free-threaded COM components

It should remembered that the objective of this project was to demonstrate the feasibility of producing a heterarchical multi-agent shop floor control system using low costs computers and standard networks not to produce commercial software for such a system. Therefore, first and foremost, further work should be targeted towards improvements of the current version of the workstation agent software into a robust and well structured package.

If the system is to be used for simulation of heterarchical shop floor systems, one issue that requires early attention is the workstation's agent module responsible for simulation of workstation activities which should be modified to allow several simulation processes to run simultaneously as completely separate and autonomous computation processes. To accommodate this change, as it was discussed in Section 12.1.10 "Design and development of the workstation agent", each process has to be programmed as a separate COM (Component Object Model) object, and the operating system has to be changed to Windows 2000 Professional or XP Professional editions. In this way the workstation agent application would be able to use the full power of multitasking program execution and would be able to support a real time simulation of several time-consuming "independent" simulation processes that are performed simultaneously inside the individual workstations. Because the workstation agents cooperate with each other while running on independent computers linked into a LAN, and because their operation is event driven, this modification is of crucial importance for obtaining accurate results in future simulations.

Further optimisation of the source code and employment of more appropriate programming "techniques" used in designing commercial applications (including the above mentioned COM technology and an "n-tiered architecture" approach) would increase functionality, quality, and reliability of the workstation agents' software.

14.3.2 Procedures for generating random events

To assist measurements of the system performance the simulation module of a workstation agent should be extended with the procedures for generating random events. During simulation, these events would cause random equipment failures, down times for machine maintenance, or would force a machine to operate under degraded conditions (as in the case of using alternative cutting tools that have worse cutting characteristics than a tool that was originally planned to be used), and so forth. Currently, the application does not have built in procedures for generating random events automatically. However, a user can simulate machine or robot failures manually (randomly at any time during the simulation) by pressing certain control buttons on the user interface (see Figure 10.6). Also, the failure of a whole workstation agent can be simulated by turning the corresponding computer off!

Also, a module for the simulation of the random arrival of work-parts into the system would need to be developed. The module would use a stochastic algorithm to determine which product, from a set of predefined products, would be manufactured next. The inter-arrival time between

orders (products) could be modelled by using one of the statistical probability distributions, for example: Poisson, Normal, Exponential, or Uniform distribution. This conforms to situations observed in practice and with experiments reported, for instance, in [Huq 1994, Pedgen et al, 1990, Wein & Chevalier 1992]. In the simulation setup reported recently by [Wang & Usher, 2002] the order/job arrival follows the Poisson distribution with each job consisting of five to nine different operations. The batch size for each job was uniformly distributed between twenty and ninety units. All the setup and processing times were estimates from experienced engineers. These values were used as the mean values of the Normal distribution to generate the values actually used in the simulation runs.

14.3.3 Module for exporting data

The existing workstation agent's decision-making module records some production data either into a local workstation database (refer to Section 10.2.1.4 "The workstation control module"), which contains data about all work-parts that have been fabricated by one workstation, or into the Processing Route Table (as indicated in Section 11.2.2.5. "Dispatching of a work-part"), which is a suitable for recording data that relate to all workstations that a work-part has visited during its manufacture. Both repositories represent valuable sources of information that are important for system performance analyses both in the actual system if it were to be installed as a real world shop floor control system or it were used for simulation. The agent decision-making module can be modified to record any data that might be of interest. For example, in addition to times when a work-part enters or leaves the workstation, enters or leaves the input or output buffers, and so forth, the module can record data that are of importance for determining production costs (for example, consumption of electric power, or how long a machine has been out of order) or data related to the machine utilisation (such as machine operational and idle times), and so on. To assist in conducting these analyses, the application should be extended with a module for exporting data into suitable formats. Such formatted data then could be easily imported into some of the statistical software packages specifically designed for that purpose. Some of the popular packages for statistical analysis (going from very simple to more complex) include: ***Microsoft Excel*** (An excellent tool for data capture, charting and conducting very basic statistical analysis. Excel can be extended with statistical "add-in"-s such as the *Analyse-it* creating a powerful yet easy to use statistical analysis tool), *MINITAB* (a general purpose easy to use package that covers a wide range of statistical analyses), *SAS* (a general purpose but very large and comprehensive statistics package), *GAUSS* (a mathematical and statistical system, a rather flexible and powerful programming language, that provides a wide variety of statistical, mathematical and matrix handling routines) and *MATLAB* (A powerful matrix manipulation and mathematical package available in the Institute of Technology and Engineering).

14.3.4 Extending the negotiation mechanism to shared resources: AGV's, tools, jigs, and fixtures

Another possible extension of the application involves expanding the heterarchical control system to encompass shop floor transport and a tool management system. The negotiation mechanism can be readily extended to include bidding for AGVs and shared tools, jigs, and fixtures among workstations. The process of selecting of an AGV for transferring parts from a source workstation to a destination workstation and the process of gathering shared tools, jigs, and fixtures can be organised in the same manner as used in this project for selecting processing machinery. A secondary negotiation process (for AGVs and shared tools, jigs, and fixtures) could be initiated as a part of the "work-part bidding process" by broadcasting (not specifically addressing) task announcements for AGVs, tools, jigs, and fixtures. In this way, in the process of forming overall bids for given work-parts, transport time for AGV and times for tool, jig, and

fixture procurements would be more "dynamic" and more accurately determined at the time when an overall bid for a work-part needs to be created. By using data obtained from the AGV controllers, and tool, jig, and fixture store controllers, which have complete insight information and control over these shared resources, these times would be more precisely determined than would be the case when they are estimated and empirically determined in advance (as they are in the current application) and stored as static values in the workstations local databases. Essentially an agent would be created for each AGV or for major tool repositories as discussed in the next section.

14.3.5 Extending the system by introducing new agents of different types

One of dilemmas during the design stage of the project was to make decision as to which system elements should be mapped as agents. In the first phase of the project, as stated in Section 7.2 "A retrospective discussion on relevant background material", the minimalist approach was adopted (to develop a system consisted of workstation agents only). Since the development of the proposed heterarchical SFC system (using commonly available computer hardware, software and programming developing tools) was proved to be feasible and practical, in the second stage of the project the possibility of designing and developing other agents should be considered. As pointed out in Section 10.1 "The proposed control architecture of a heterarchical shop floor control system", some possibilities include development of:

- ***A management-control agent.*** This agent would have the role of a shop floor monitoring agent where data on progress of all jobs could be recorded and displayed. With networked computers providing the selection of workstations and local schedules of work-parts it is possible to monitor what is occurring in the system in real time and to "adapt" to changes in machines' states throughout the factory floor. This means that the workstation agents can be interrogated at any time by the management-control agent (a central monitoring system) for information on progress and the current status of the whole job shop. If required, the management-control agent would be able to enforce some corrective actions in the system affecting the behaviour of workstation agents (for example, in the case of emergency, or when the production of a specific work-part needs to be directly controlled[1], or when the production policy needs to be changed[2]). The management-control agent should reside on a separate computer dedicated for this purpose.
- ***A tool agent.*** While the need for developing a management-control agent is apparent, the development of separate tool agents should be carefully considered. If the number of shared tools in the system is high and if they are frequently shared among workstations, then there is good reason to consider the development of separate tool agents for each shared tool in the system. In this case, tool agents should be developed as mobile agents that would reside on different workstation computers (following the physical location of shared tools in the system, one tool agent would be located on one workstation at a time). Alternatively, if the number and requirement for sharing tools is low then the existing workstation agent software could be extended to negotiate shared tools (in addition to

[1] For example, by changing the value of a work-part's "total priority index" it is possible to control the process of work-part fabrication in the system.

[2] The change of a system operation mode (for example, switching from maximum productivity to maximum economy mode) can be achieved by choosing another negotiation scheme for selecting the workstations and/or by picking another criterion for local scheduling of work-parts inside workstations.

negotiation of work-parts, as discussed in Section 14.3.4 "Extending the negotiation mechanism to shared resources: AGVs, tools, jigs, and fixtures").

- ***A fixture/jig agent.*** All considerations about shared tools, also apply to shared jigs and fixtures in the system.
- ***A work-part agent.*** Each time when a new work-part enters the system, instead of creating a PRT, a new work-part agent associated with the work-part could be created. This approach is adopted by many researchers. Work-parts were modelled either as agents (called part, job or product agents), for example in [Dewan & Joshi, 2001, Lin & Solberg, 1992, Ottaway and Burns, 2000, Saad et al., 1997, Wang & Usher, 2002] or as holons, examples include [Brussel et al., 1999, Kanchanasevee et al., 1997]. Each work-part agent would be in charge of determining the work-part's unique processing route by negotiating with the workstation agents that have the potential to accomplish a given processing task. Work-part agents should be developed as mobile agents. They would have their own database, communication module for sending and receiving messages, a decision-making module, etc.
- ***A system advisory agent.*** To achieve higher level of global coherence of the system, an advisory agent could be introduced, see Section 14.4.4 "Achieving global coherence" for more details.

14.3.6 Development of virtual workstations

By using modern, discrete-event simulation tools, such as QUEST[1], it is possible to build a virtual workstation on each of the workstation computers on which corresponding workstation agents can reside.

Using QUEST, for example, for constructing very detailed three dimensional (3D) workstation models in which all manufacturing activities, such as robot, machine, and parts movements, are simulated by using highly visualised real time 3D animation, it would be possible to conduct a set of extremely high quality and realistic experiments. These very sophisticated simulations would offer a realistic view of workstation activities. QUEST's tools could be also used for conducting analysis and evaluation of workstation performance.

Integration of the virtual workstation (developed using QUEST) with the workstation agent should be performed in the same way as with the real workstation, that is, through the virtual manufacturing devices (VMDs). From the workstation agent point of view there would not be significant differences in dealing with the real or virtual workstation. In both cases, the part of the VMD, relating to the workstation agent, would be exactly the same. What would be different is the part of the VMD connecting to the equipment inside workstation depending on what the VMD interacts with, namely a real or virtual workstation equipment.

14.3.7 Development of a real workstation

After performing a set of experiments related to the measurement of system performance, to demonstrate the performance of the application on real "hardware", but under laboratory conditions, a further step in developing the heterarchical shop floor control system could be to implement a production workstation agent on a real workstation system.

---

[1] The Queuing Event Simulation Tool (QUEST®/Deneb) - a tool for modelling, simulation and analysis of manufacturing systems.

It is believed that the incorporation of a real world manufacturing workstation using either a Flexible Manufacturing System or a few Numerically Controlled machine tools is a relatively straightforward process. This simplicity is considered one of the key advantages of the multi-agent system developed for this project. Therefore, it would be appropriate to demonstrate this. The Institute of Technology and Engineering at Massey University had a numerically controlled lathe and mill and if the project had continued the incorporation of these as part of the network would have been a useful exercise. Installing the agent software on the teaching PC computers and the connection via the Institute's Ethernet system to these machines would have been a practical proposition[1]. However, both of these machines were relatively old and it would be more appropriate to make the tests on more modern machining systems.

Integration between the agent control module and existing software used for controlling equipment in the workstation would be a challenge of significant importance. This integration would address some practical issues regarding implementation of the methodology developed herein into a real, fully automated, manufacturing environment. One of the challenges would be to develop virtual manufacturing devices (VMDs) as interfaces (drivers) between the workstation agent's control module and real equipment controllers in the workstation. Therefore, at this stage, the advances achieved in the research area of virtual manufacturing devices would be beneficial [Mo et. al., 1996].

The real workstation could be integrated into a simulated model of heterarchical shop floor control system. Cooperation and information exchange between a workstation agent of the real workstation and other workstation agents in the manufacturing system model, whose workstation production activities are simulated, would be performed in exactly the same manner. From the heterarchical shop floor control point of view there would be no differences between these workstation agents. In this way the whole heterarchical shop floor control system would be brought to a higher level of "integrated simulation". Implementation of a workstation agent on a real workstation and its integration with the rest of simulated manufacturing system model would boost the overall confidence in the practicality of the heterarchical shop floor control system. Production workstation agents would deal with more detailed, real life information. Since all workstation agents use essentially the same software, these upgrades would be included in all other simulated workstations which are part of the manufacturing system model.

14.3.8 Incorporation of a shop floor management supervisory system

It was noted that one of the concerns of factory managers contemplating a heterarchical shop floor control system is that it could get completely out of control and generate into chaos if some form of supervision and monitoring were not available. As stated in Section 14.3.5 "Extending the system by introducing new agents of different types" it is possible to envisage a factory with a manned central operating station where by the progress of jobs and problems occurring on the shop floor could be displayed in a control panel. The incorporation of a management agent would assist the control room operators in interpreting the data from the shop floor and in some cases taking automatic action to correct certain problem situations. However, the management agent would be complex and the control room operators would be able to undertake the agent's

[1] The networked PCs in the Management Laboratory of the Institute were not used for development as this would have interrupted both their use as teaching machines and the development process of the agent software. However, since separate facilities were provided for software development, it would be possible to install only operational software on the Management Laboratory network, and to schedule tests providing at least six machines for performance testing.

task anyway. It should be noted that the control room is a monitoring and problem resolving system as is not in a master slave relationship with the shop floor agents. The concept can be extended to a shop floor consisting almost entirely of autonomous computer controlled workstations and transport systems with personnel dedicated to critical problem solving and maintenance as directed by the control room. With the modern CAD/CAM systems the control software for the part production on the workstation machines could be developed, simulated and tested off line and downloaded through the network prior to the arrival of the first part order. Thus the introduction of new parts and new machines would be a highly automated process.

14.3.9 Extension to external companies via the Internet

One of the more successful global manufacturing arrangements is the "clustering" of specialised facilities for the production of particular groups of products such as furniture, footwear, clothing machine tools etc. The negotiation mechanism between the agents in this system can be extended to embrace manufacturing resources located in the other, nearby manufacturing facilities. Integration of agents-based companies into larger virtual manufacturing facilities would be immediate and transparent. The system could dynamically create a "virtual" factory of machines that are placed in different locations, belonging to different owners each specialising in a particular area of production. For example, if some processing operations require services (such as high speed machining or automated textile cutting operations etc.) that are provided in another company in the cluster that is located reasonably close to the manufacturing facility then, because the interaction between workstation agents is the same within a factory as that between one factory and another, much larger "virtual" manufacturing facilities can be easily created (in this case the Internet should be used instead of a LAN). This concept is in essence an automation of the tendering process used between specialised manufacturing companies within a cluster. The ability to shift quickly between manufacturing parts in-house and "purchasing them from a supplier" or using a specialist production unit becomes an increasingly important feature that helps companies to remain competitive in increasingly global markets and to meet difficult customer requirements. For the specialist company the additional throughput generated by being part of a cluster can justify the expense of purchasing more advanced, automated machinery and make the operation for all participants more cost effective and less time consuming.

14.4 Investigation of different controlling strategies

One of the important future tasks would be to conduct a set of experiments with different controlling scenarios. For example, changing the point of time when the workstation agent starts the negotiation process in the case when an urgent work-part is to be processed. Another interesting area for research would be to investigate how different local scheduling criteria applied on individual workstations would affect the global performance of the system. Namely, each workstation, as an autonomous production unit, can be set to use a different local scheduling criterion at any one time. A workstation can be set to dynamically change its local scheduling criterion depending upon the conditions in the workstation or the particular part production requirements at the time. For example, if there are more then 5 work-parts in the input buffer then select the criterion 2, otherwise use the criterion 1.

Other experiments could be conducted with different ways of determining part priorities. For example, as mentioned in Section 10.3.2 "Local scheduling - scheduling methods for each individual production workstation agent" the lateness portion of a part priority index could be determined according to the due dates that would be recorded in Processing Route Tables (PRT) for each part, instead to be increased each time when a work-part in the input buffer of a workstation is skipped by a work-part that has a higher overall priority index. The lateness

portion of priority index would be progressively increased as the difference between "due time minus current time" and "estimated remaining processing time" decreased.

14.4.1 Resolving conflicting negotiation situations

The system could be extended to include processes for resolving conflicting negotiation situations. For example, consider a conflicting situation when two work-parts compete for the same workstation at the same time. Because there is no direct way for two source workstation agents (workstations that possess work-parts) to know that the other is making a competitive bid while they negotiate for their work-parts with a destination workstation, there is a danger that one of the work-parts will experience an unexpectedly longer waiting time than was initially determined during a negotiation process. In other words, in the current realisation of workstation agents, if two negotiation processes are running simultaneously on a destination workstation then, in the case that this destination workstation is selected as a winner workstation in both negotiation processes, information about "completion time" in a "bid message" sent to one of the source workstations would be incorrect. This problem can be solved by terminating the negotiation process which is being conducted for the work-part that has a lower priority.

14.4.2 Improving schedule performance

The process of scheduling work-parts among workstations can be modified to include a higher degree of forecasting, making the global scheduling process more sophisticated. For example, for a work-part whose processing task can be performed on only one workstation perceived as a "potential bottleneck workstation" (the only available workstation in the Processing Route Table (PRT) for that particular processing task) the auctioning process could be activated one processing task in advance. Work-parts with high priorities that need to be produced on the potential bottleneck workstations need to reserve these workstations some time in advance. By reserving a workstation the workstation would have information about the potential arrival of a "critical work-part" and would be able to consider the time required to process this part in assembling its bids.

The approach described above could be used to achieve better global performances of the proposed control system, particularly for high priority work-parts, by applying it to all workstations and not only to the ones that are perceived as potential bottlenecks.

14.4.3 Managing bottlenecks in a heterarchical multi-agent system by prior simulation

In a highly dynamic system with multiple redundancy and versatile product mix, (such as the system presented in this project) in which process requirements change frequently over time, capacity bottlenecks will not be constant, either in time or location. The problem is to identify those bottlenecks that cause a high degree of imbalance of workload between workstations. Once the bottleneck workstations are known, then the system performances can be improved by ensuring that the work-parts with unique requirements for the bottleneck workstations are served in preference to other work-parts that have alternative routes. Bottlenecks in our model of manufacturing system can be classified into two groups: global and specific bottlenecks.

- ***Global bottlenecks*** are those workstations that limit the capacity of a system as a whole. To identify this type of bottlenecks the system should be observed over a long time. These bottlenecks can be identified by "asking" the workstations (that is, by broadcasting messages to the all workstations) to report periodically their utilisation and a number of "task announcement" messages that have not been responded to over a selected period of

observation (for example several weeks). Those workstations that had a high level of utilisation and a large number of refused "task announcement" messages over this period are bottleneck workstations. It is then up to the system managers to decide whether they wish to relieve the bottleneck by introducing an additional workstation in the system or to relax production plans. Note that this process could be simulated by the system developed in this project. During simulation the manufacturing processes inside workstations could be run faster than they are performed in real systems to establish the potential impact of global bottlenecks and hence the likelihood for the requirement to introduce additional workstations. This allows this planning to be undertaken before major problems occur and ahead of the lead-time for delivery of the additional workstations if they are required.

Note: When "speeding up" simulation of manufacturing processes inside workstation there is real danger of "clogging up" the network. The increased frequency of transferring messages among workstations can lead to the collapse of the system – by quickly reaching the saturation point on the LAN. Simulation in the heterarchically controlled model of the job shop manufacturing system can be carried out either in time scaled or real time manner. A ***time-scaled simulation*** is characterised by: 1) much faster simulation than an event-based technique (simulation can be limited only by the communication resources or theoretically by computing resources); 2) the existence of synchronisation mechanism that is required to ensure right sequences of events in distributed system (all internal clocks of the workstations have to be synchronised and all workstations have to have the same accelerating factor – speed of executing manufacturing operations); and 3) an ability to perform time scaling. The main characteristics of the ***real time simulation*** technique are as follows: 1) the simulation of workstation's manufacturing processes takes the same time as the real processes would be in a real manufacturing system. The simulation is very slow because it is based on triggering events in real time; 2) the sequence of message transmission and reception does not have to be synchronised and preserved; and 3) CPU times and network bandwidth may not be fully utilised because there may be no events to be simulated for long periods. Which methodology is to be implemented is a design issue that involves tradeoffs between complexity of application and maximising the utilisation of the computer's CPUs and the network bandwidth. Originally it was envisaged that the application would support both modes, however, because time-scaled simulation could lead to the congestion on the LAN (causing problems that would not exist in the real–life systems) the current realisation of application supports only real time simulation.

- ***Specific bottlenecks*** apply more severely to a particular group of work-parts (work-parts that have processing tasks that can be performed on only one workstation), and could be more troublesome as the demand for those work-parts increases. Workstations that are indicated as the only one in the processing route tables (PRTs) for performing specific tasks are all potential "specific bottlenecks".

Our attention in the future study should be to cope with these "specific bottlenecks". Emerging bottlenecks of this type in the system can be avoided to a certain extent if the work-parts that have several alternatives are directed to other workstations instead of to the workstation that is perceived as the potential "specific bottleneck". To reduce possibility of occurring "specific bottlenecks" and, therefore, to avoid delaying of a particular group of work-parts, the following rules might be applied:

- ***From the perspective of work-parts:*** For a work-part whose processing task can be performed on only one workstation the auctioning process could be activated one

processing task in advance to allow this workstation to reserve its resources. This rule is described in more detail in Section 14.4.2"Improving scheduling performances".

- ***From the perspective of workstations:*** Workstations may have a rule which, for example, says: "if utilisation of the workstation is above 80%, confine the workstation's bidding to jobs that have been rejected by the other workstations more than 2 times in the last 5 minutes". In this way, workstations would be able to "intelligently" pick up "orphan" work-parts, and ignore jobs that can find alternative workstations, regardless of how effective or efficient are these workstations in producing those work-parts.

An important part of the workstation reporting to the factory management would be to warn management that there are work-parts in the system that cannot be manufactured on a certain workstation[1]. If this stoppage in processing work-parts was caused because a workstation was overloaded and if this issue is repeated frequently on the same workstation, then an option for acquiring additional processing capacity for the operations that a specific bottleneck workstation performs should be considered.

14.4.4 Achieving global coherence

Achieving global coherence is recognised as a potential problem in heterarchical agent based systems. There are, however, several ways to increase the global coherence of heterarchical control systems. Three of them are proposed by Prabhu [Prabhu, 2000] and could be incorporated in the system described here in the future. These proposals are briefly introduced in the following subsections.

14.4.4.1 By using a global scheduling criterion

One way to solve the global coherence problem is to use a global scheduling criterion that would be common to all workstations in the system for a given time period. In this approach, the global goal of the system can be viewed as an aggregation of all of the local goals of its elements. For example, if the global goal is to minimise total lateness then local goal for each workstation can be lateness minimisation (all workstations would use minimum slack criteria as a local scheduling criteria).

14.4.4.2 By using global scheduling agent

The second way to solve the global coherence problem is by using a global scheduling agent. In this approach, the global scheduling agent would have an overview of entire system by receiving information from all workstations. Having global "picture" of the system and necessary global data, the agent would be able to coordinate activities among autonomous workstations in the system. For example, the global scheduling agent would be able to set different local goals for individual workstation agents in order to achieve a desired global system goal. If before executing its plans, a workstation agent realises that its plan would not achieve a newly set local goal then a poor performance of the overall system can be avoided by not executing this plan; a workstation can try different permutations or modifications of its plans until an acceptable solution is found. However, the global scheduling agent would only play the "advisor's role", that is, it would advise (not control) each workstation from time to time to undertake some corrective actions with the aim of improving the overall system performance. This "advice" can

[1] If a work-part cannot be processed on a specific bottleneck workstation for some time, for example, if the destination workstation does not respond to several task announcements, this situation should be reported. The ID number of the specific bottleneck workstation is known from the processing route table (PRT) for that work-part.

either be immediately accepted by workstations or postponed to a later stage for any particular reason. For example, the global agent can advise a workstation to change the local scheduling criterion, but because of the presence of an urgent part in the input buffer, the workstation may decide not to change the scheduling criterion until it finishes the processing of the urgent part. This means that in the "absence" of the global scheduling agent, for any reason (e.g. due to failure of the computer system on which global scheduling agent resides or communication problems), the system will still be operative but the overall system performance would presumably deteriorate over time. It should be emphasised that this form of cooperation and global coherence can be achieved with minimal global data, and without introducing master-slave relationships among workstations. In other words, cooperation can be achieved without compromising the principles of heterarchical systems.

14.4.4.3 By using cooperative scheduling

Yet another approach for resolving the global coherence problem is suggested by Prabhu [Prabhu, 2000]. He suggests using a cooperative scheduling method that is based on simultaneous execution of simulated and real-life control processes. In his approach each entity is decomposed into two new entities: 1) a *real entity* comprised of the real controller that interacts with the physical system; and 2) a *virtual entity* comprised of two parts: the simulated controller, called a *virtual controller,* and a scheduler module that "interacts" with the simulated system. The virtual entities continually generate the new plans (tentative local schedules for the real system) using local information and heuristics, which then are simulated together in order to evaluate their local and global performance. The set of local schedules that results in the best global performance is used as the schedule for the shop-floor even if a particular local schedule is locally suboptimal. This improves global performance and results in implicit cooperation among entities.

In the suggested approach, the performance of the system would be evaluated by using a deterministic, time-scaled distributed simulation of the entire manufacturing system and an *evaluation entity* that would be introduced in the system to synchronise and compute the merits of each simulation. Each time when a new plan for a virtual entity in the system is completed, the virtual entity can request the "evaluator" for an appraisal of its new plan. The simulation duration can also be specified by the requesting entity. If another virtual entity has completed a new plan, the evaluator can evaluate it in the same simulation. Virtual entities that have not completed their plans simulate their previous plans. All virtual entities participate in simulations. This is an extension of the simulation process discussed above for the identification of bottlenecks but with the purpose of optimising the overall performance of the system through improved global coherence.

Simulation uses a replica of the real system software which is executed several orders of magnitude faster than real-time. In simulation each virtual entity uses a *virtual controller*, which "executes" different plans generated by a local scheduler. In such an "execution", virtual entities interact between each other and send messages exactly as real entities would be in the real system. However, the simulation activities would be performed much faster than in the real system for two reasons. Firstly, all physical actions are simulated by the virtual controllers, and secondly, since the virtual controllers and the schedulers are tightly coupled, the rate of information flow between them would be significantly higher than the rate of flow of information between the real controller and the scheduler. Accurate performance evaluation and schedule effectiveness can be ensured by initialising each simulation to the state of the shop-floor and periodically re-evaluating the performance of the best schedule.

14.5 Integration of a distributed heterarchical shop floor control system with the other modules of a traditional Production Planning and Control system

Nowadays, production planning and control in the systems that are characterised with relatively stable production programs, with predictable customer orders, is typically performed using a set of integrated software packages. These software packages, which are usually based on the principles of Manufacturing Resource Planning (MRPII) approach, tend to cover most, if not all aspects of a Production Planning and Control (PPC) system. It is considered that the heterarchical shop floor control system developed in this project could be integrated with these software packages without difficulties. As input data, the heterarchical shop floor control system would receive job orders and accompanying processing route table files. On the other hand, from the heterarchical shop floor control system data about the state of the shop floor (such as information regarding monitoring quality and reporting operating statistics) would be sent back to the other modules of the PPC system as output data.

The main benefit of this integration would be a significant reduction in the material and capacity planning functions, which would otherwise need to generate very detailed short-term plans on an hourly basis. The other benefit of this integration would be in reduction of the amount of the information exchanged between the shop floor control module and the other modules of the PPC system. In the proposed heterarchical system there is no need for detailed work-in-process tracking.

14.6 Summary

In summary the possibilities for evaluating and extending the system are considerable now that the practical operation of a system model has been demonstrated. Perhaps the most important opportunity is that the project has demonstrated that a low cost agent based system can be developed and this system would appear to have commercial potential. To achieve this development the software would need to be refined and restructured. On the other hand the model system produced could provide the basis for an ongoing research programme to investigate the capabilities and limitations of heterarchical multi-agent based systems using commonly available hardware. Even more universality of its application and development could possibly be obtained in the future using open operating systems such as Linux as the basis for the software development. It is with regret that because of the changes in personnel and the restructuring of the Department of Production Technology and the Institute of Technology and Engineering at Massey University that this project is unlikely to proceed further. It is hoped that a sponsor for the further extensions can be found in the future.

15 Literature

[Adams J. Balas E., and Zawack D., 1988]
"The Shifting Bottleneck Procedure for Job Shop Scheduling"
Management Science, Vol.34 No.3, pp.391-401.

[Agility-forum, web]
URL: http://www.agility-forum.de/index.html

[Anstiss, P., 1988]
"The Implementation and Control of Advanced Manufacturing Systems"
Control and programming in Advanced Manufacturing, IFS Publications Ltd. UK, Springer-Verlag, Berlin, 1988.

[Armentano V.A. and Scrich C.R., 2000]
"Tabu search for minimizing total tardiness in a job shop"
International journal of production economics, Vol.63, pp.13-140

[Arzi Y., and Roll Y., 1993]
"Real-time Production Control of a FMS in a Produced-to-Order environment"
International Journal of Production Research, Vol.31, No.9, 1993, pp.2195-2214.

[Arzi Y., 1995]
"On-line scheduling in a multi-cell flexible manufacturing system"
International Journal of Production Research, Vol.33, No.12, 1995, pp.3283-3300.

[Ashby W.R., 1956]
"*An introduction to cybernetics"*
Chapman and Hall, London, 1956.

[Baker A.D., 1991]
"Manufacturing Control with a Market-Driven Contract Net"
PhD Dissertation, Department of Electrical Engineering, Rensselaer Polytechnic Institute, Troy, NY.

[Baker, A. D., and Merchant, M. E., 1993]
"Automatic factories: How will they be controlled",
IEEE Potentials, Vol.12, No.4, pp.15-20.

[Baker A.D., 1995]
"Agility and Factory Performance from an Autonomous-Agent Architecture"
Proceedings to the 4th Annual Agility Forum Conference, Atlanta, GA, March 7-9, 1995, pp. 354-367.

[Baker A.D., 1996]
"Metaphor or Reality: A Case Study where Agents Bid with Actual Costs to Schedule a Factory"
In "Market Based Control: A Paradigm for Distributed Resource Allocation", edited by Clearwater S.H., Singapore: World Scientific Publishing Co. Pte.Ltd., 1996, pp. 184-223.

[Baker A.D., Parunak H.V.D. and Erol K., 1997]
"Manufacturing over the Internet and into Your Living Room: Perspectives from the AARIA Project".
Working Paper, Department of Electrical & Computer Engineering and Computer Science, University of Cincinnati.
URL: http://www.ececs.uc.edu/~abaker

[Baker A.D., 1998]
"A survey of factory Control Algorithms that can be implemented in a Multi-Agent Heterarchy: Dispatching, Scheduling, and Pull"
Journal of Manufacturing Systems, Voll 7, No.4, pp.297-320, 1998

[Balasubramanian S., and Norrie D.H., 1995]
"A Multi-Agent Intelligent Design System Integrating Manufacturing and Shop-Floor Control"
Proceedings of the First International Conference on Multi-agent Systems (ICMAS '95), San Francisco, California, June 12-14, 1995, pp.3-9.

[Barbuceanu M., and Fox M.S., 1995]
"COOL: A Language for Describing Coordination in Multi-Agent Systems",
in V. Lesser (ed), Proceedings of the First Intl. Conference on Multi-Agent Systems, AAA Press/The MIT Press, june 1995, pp 17-25.

[Barbuceanu M., and Fox M.S., 1996]
"The Agent Building Shell: A Tool for Building Enterprise Multi-Agent Systems",
Canadian Artificial Intelligence 40, Autumn 1996, pp 9-11.

[Ben-Daya M., 1995]
"FMS Short Term Planning Problems: A Review"
In: "Flexible Manufacturing Systems: Recent Developments" edited by Raouf A. and.Ben-Daya M., Elevier Science B.V., 1995, pp.113-139.

[Benjaafar S., 1994]
"Models for Performance Evaluation of Flexibility"
International Journal of Production Research, Vol.32, No.6, 1994, pp.1383-1402.

[Benjaafar S.B., Talavage J., and Ramakrishnan R., 1995]
"The Effect of Routing and Machine Flexibility on the Performance of Manufacturing Systems"
International Journal of Computer Integrated Manufacturing, Vol.8, 1995, pp.265-276.

[Benjaafar S., and Ramakrishnan R., 1996]
"Modelling, Measurement and Evaluation of Sequencing Flexibility in Manufacturing System"
International Journal of Production Research, Vol.34, No.5, 1996, pp.1195-1220.

[Berry W.L., and Hill T.J., 1989]
"Linking systems to Strategy"
Working Paper, Chapel Hill: University of North Carolina, Kenan-Flager School of Business, 1989.

[Bertrand J.W.M., Wortmann J.C.& Wijngaard J., 1990]:
"Production Control: A Structural and Design Oriented Approach"
Eindhoven University of Technology, Elsevier Science Publishers, the Netherlands

[Biegel J.E., and Davern J.J., 1990]
"Genetic Algorithms and Job Shop Scheduling"
Computers and Industrial Engineering, Vol.19, No.1, pp.81.

[Biemans F.P., and Vissers C.A., 1989]
"Reference Model for Manufacturing Planning and Control Systems"
Journal of Manufacturing Systems, Vol.8, No.1, pp.35-45.

[Bilberg A., and Alting L., 1991]
"When Simulation Takes Control"
Journal of Manufacturing Systems, Vol.10, No.3, pp.179-193.

[Bjork B. and Wix J., 1991]
"An Introduction to STEP. Technical Report"
VTT Technical Research Centre of Finland and Wix McLelland Ltd., England.
URL: http://www.nist.gov/sc4/www/stepdocs.htm

[Blackstone J.H., Phillips D.T., and Hogg G.L., 1982]
"A State-of-the-art Survey of Dispatching Rules for Manufacturing Job Shop Operations"
International Journal of Production Research, Vol.20, No.1, pp.27-45.

[Blayzewicz J., Domschke W., and Pesch E., 1996]
"The job shop scheduling problem: Conventional and new solution techniques"
European Journal of Operational Research, No.93, pp.1–33, 1996.

[Blexrud C., et al, 2000]
"Professional Windows DNA - Building Distributed Web Applications with VB, COM+ MSMQ, SOAP and ASP"
Wrox Press Ltd., 2000.

[Bona B., Brandimarte P., Greco C., and Menga G., 1990]
"Hybrid Hierarchical Scheduling and Control Systems in Manufacturing"
IEEE Transactions on Robotics and Automation, Vol.6, No.6, pp.673-686.

[Bongaerts L., Indrayadi Y., Van Brussel H., and Valckenaers P, 1999]
"Predictability of Hierarchical, Heterarchical, and Holonic Control",
Proc. of the 2nd International Workshop on Intelligent Manufacturing Systems, Leuven-Belgium, September, 1999, pp.167-176.

[Brandimarte P., 1993]
"Routing and Scheduling in Flexible Job Shop by Tabu Search"
Annals of Operations Research, Vol.41, pp.157-183.

[Brandin, B.A., 1996]
"The Real Time Supervisory Control of an Experimental Manufacturing Cell"
IEEE Transactions on Robotics and Automation, Vol.12, No.1, pp.1-14.

[Bratman M., 1988]
"Plans and resource bounded practical reasoning"
Computer intelligence, Vol.4, No.4, November 1988

[Brazier, F., Jonker, C., Treur, J. and Wijngaards, N., 1998]
"Compositional Design of Generic Design Agent"
In Proceedings of AI & Manufacturing Research Planning Workshop, Albuquerque, NM, The AAAI Press, pp. 30-39.

[Brennan R.W., Balasubramanian S., and Norrie D.H., 1997]
"Dynamic Control Architecture for Advanced Manufacturing Systems"
In Proceedings of International Conference on Intelligent Systems for Advanced Manufacturing, Pittsburgh, PA.
URL: http://imsg.enme.ucalgary.ca

[Brennan R., 2000]
"Performance comparison and analysis of reactive and planning-based control architectures for manufacturing"
Robotics and Computer Integrated Manufacturing Vol.16, No.2-3, pp.191-200.

[Brennan R.W. and William O, 2000]
"A simulation test-bed to evaluate multi-agent control of manufacturing systems"
Proceedings of the 2000 Winter Simulation Conference, J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, Eds.

[Brennan R., Zhang X., Xu Y., and Norrie D., 2002]
"A reconfigurable concurrent function block model and its implementation in real-time Java"
Journal of Integrated Computer-Aided Engineering Vol.9, pp.263–279.

[Brill M., and Gramm U., 1991]
"MMS : MAP application services for the manufacturing industry"
Computer Networks and ISDN Systems, Vol.21, pp.357-380.

[Brill P.H. and Mandelbum M., 1990]
"Measurement of Adaptivity and Flexibility in Production Systems"
European Journal of Operational Research, Vol.49, No.3, pp.325-340.

[Browne J., Dubois D., Rathmill K., Sethi S.P., and Stecke K.E., 1984]
"Classification of Flexible Manufacturing Systems"
The FMS magazine, Vol.4, April 1984, pp.114-117.

[Bruckner S., Wyns J., Peeters P., and Kollingbaum M., 1998]
"Designing Agents for Manufacturing Control"
URL: http://citeseer.nj.nec.com/539132.html

[Brussel Van H., 1994]
"Holonic Manufacturing Systems, the vision matching the problem"
In Proceedings of the First European Conference on Holonic Manufacturing Systems, Hannover, Germany, December, 1994, pp.

[Brussel Van H., Wyns J., Valckanaers P., Bongaerts L., Peeters P., 1998]
"Reference Architecture for Holonic Manufacturing Systems: PROSA"
PROSA. Computers in industry, Vol.37, No.3, pp.255-276.

[Brussel H., Bongaerts L., Wyns J., Valckenaers P., Ginderachter T., 1999]
"A conceptual framework for holonic manufacturing: Identification of manufacturing holons"
Journal of manufacturing systems, Vol.18, No.1, pp.35-52.

[Buffa E. S., Armour G. C., and Vollmann T. E., 1964]
"Allocating facilities with CRAFT"
Harvard Business Review, Vol. 42, No. 2, pp.136-158.

[Burbidge J.L., 1978]
"The Principles of Production Control"
4th edition McDonald & Evans, UK, 1978.

[Burbidge J.L, 1989]
"Production Flow Analysis"
Oxford science publication, Oxford, UK, 1989.

[Burke P., and Prosser P., 1994]
"The Distributed Asynchronous Scheduler"
In: "Intelligent Scheduling", edited by Morgan M. B., Morgan Kaufman Publishers, Inc., San Francisco, 1994, pp.309-339.

[Bussmann S., 1998a]
"Agent-Oriented Programming of Manufacturing Control Tasks"
In Proceedings of Third International Conference on Multi-Agent Systems (ICMAS'98), pp.57-63, IEEE Computer Society, 1998.

[Bussmann S., 1998b]
"An Agent-Oriented Architecture for Holonic Manufacturing Control"
In Proceedings of First International Workshop IMS Europe, Lausanne, Switzerland, pp. 1-12.

[Bussmann, S. McFarlane D.C., 1999]
"Rationales for Holonic Manufacturing Control"
In Proc. of Second Int. Workshop on Intelligent Manufacturing Systems, Leuven, Belgium, pp.177-184.

[Bussmann S., and Schild K., 2000]
"Self-Organizing Manufacturing Control: An Industrial Application of Agent Technology"
In Proc. of the Fourth Int. Conf. on Multi-Agent Systems, Boston, MA, USA, 2000, pp.87-94.

[Bussmann S. Jennings N., and Wooldridge M., 2000]
"On the Identification of Agents in the Design of Production Control Systems"
AOSE, pp.141-162.

[Bussmann S., Jennings N., Wooldridge M., 2002]
"Re-use of Interaction Protocols"
URL: http://citeseer.nj.nec.com/bussmann02reuse.html

[Butler J., and Ohtsubo H., 1992]
"ADDYMS: Architecture for Distributed Dynamic Manufacturing Scheduling"
In: "Artificial Intelligence Applications in Manufacturing", edited by Famili A., Nau D. S., and Kim S. H., AAAI Press/The MIT Press, Menlo Park, CA, 1992, pp.199-214.

[Buzacott J.A., 1982]
"The Fundamental Principles of Flexibility in Manufacturing Systems"
Proceedings of the First International Congress on Flexible Manufacturing Systems, Bedford, UK, 1982, pp.13-22.

[Camarinha-Matos L., Carelli R., Pellicer J., Martín M., 1997]
"Towards to the Virtual Enterprise Food Industry"
In Proceedings of International Conference on Integrated and Sustainable Industrial Production, Eds. Luis Camarinha-Matos, Chapman & Hall, Lisboa, 1997, pp 73-84.

[Camarinha-Matos L.M. and Afsarmanesh H.; 1997]
"Virtual Enterprises: Life cycle supporting tools and technologies"
in Handbook of Life Cycle Engineering: Concepts, Tools and Techniques,
A. Molina, J. Sanchez, A. Kusiak (Eds.), Chapman and Hall, 1997.

[Campbell H.G., Dudek R.A., and Smith M.L., 1970]
"A Heuristic Algorithm for the "n" Job "m" Machine Scheduling Problem"
Management Science, Vol.16, No.10, June 1970.

[Caprihan R. and Wadhwa S., 1997]
"Impact of Routing Flexibility on the Performance of an FMS: A Simulation Study"
International Journal of Manufacturing Systems, Vol.9, 1997, pp.273-298.

[Carlson R., 1992]
"Management of Flexible Manufacturing: An International Comparison"
OMEGA, Vol.20, No.1, 1992, pp.11-22.

[Carrie A, 1988]
"Simulation of Manufacturing Systems"
Willey, New York, 1988

[Casad J., 2001]
"Teach Yourself TCP/IP in 24 Hours, Second Edition"
Sams Publication, 2001.

[Chaib-draa, B.; 1995]
"Industrial Applications of Distributed AI"
Communications of the ACM, Vol.38, No.11, 1995, pp.49-50.

[Chandra J., and Talavage J., 1991]
"Intelligent Dispatching for Flexible Manufacturing"
International Journal of Production Research, Vol.28, No.11, 1991, pp.2259~2278.

[Chandra P. and Tombak M., 1992]
"Models for the Evaluation of Routing and Machine Flexibility"
European Journal of Operational Research, Vol.60, No.2, 1992, pp.156-165.

[Chang, Y. L, Sueyoshi, T. and Sullivan, R. S., 1996]
"Ranking Dispatching Rules by Data Envelopment Analysis in a Job-Shop Environment"
IIE Transactions, Vol. 28, No.8, pp.631-642.

[Chen I.J., and Chung C.H., 1991]
"Effects of Loading and Routing Decisions on Performance of Flexible Manufacturing Systems"
International Journal of Operational Research, Vol.29, 1991, pp.2209-2225.

[Cheng R., Gen M. and Tsujimura Y., 1999]
"A tutorial survey of job-shop scheduling problems using genetic algorithms, part I: representation, part II: Hybrid genetic search strategies",
Computer and Industrial Engineering, Vol. 36, pp 343-364

[Cheung J. Y., 1994]
"Scheduling"
in Dagli, C. H. (ed) Artificial Neural Networks for Intelligent Manufacturing,
Chapman and Hall, London, Chapter 8, pp. 159-193.

[Chiu C.; Yih Y., 1995]
"Learning based methodology for dynamic Scheduling in Distributed Manufacturing Systems"
International Journal of Production Research, Vol. 33, No. 11, 1995, pp. 3217-3232.

[Cho H., Derebail A., Hale T., and Wysk R.A., 1994]
"A Formal Approach to Integrated Computer Aided Process Planning and Shop Floor Control"
Journal of Engineering for Industry, 1994, January.

[Cho H., and Wysk R.A., 1995]
"Intelligent Workstation Controller for Computer Integrated Manufacturing: Problems and Models"
Journal of Manufacturing Systems, Vol.14, No.4, 1995, pp252-263.

[Choi K.H., Kim S.C., Yook S.H., 2000]
"Multi-agent hybrid shop floor control system"
International Journal of Production Research, Vol.38, No.17, pp.4193-4203.

[Chowdhury, D. D., 2000]
"High Speed LAN Technology Handbook"
Springer-Verlag Berlin and Heidelberg GmbH & Co. KG, ISBN: 3540665978

[Chryssolouris G., Chan S., and Cobb W., 1984]
"Decision Making on the Shop Floor: An Integrated Approach to Process Planning and Scheduling"
Robotics and Computer Integrated Manufacturing, Vol.1, 1984, pp.315-319.

[Chryssolouris G., Pierce J., and Dicke K., 1991]
"Madena: An Approach for Allocating Manufacturing Resources to Production Tasks"
Journal of Manufacturing Systems, Vol.10, 1991, pp.368-382.

[Chu-Carroll J.; and Carberry S.; 1995]
"Communication for Conflict Resolution in Multi-Agent Collaborative Planning"
Proceedings of the First International Conference on Multi-Agent Systems, 1995, pp.49-56.

[Comer E.D., 2000]
"Internetworking With TCP/IP Volume 1: Principles Protocols, and Architecture"
4th edition, 2000.

[Conry S. E., Kuwabara K., Lesser V. R., and Meyer R. A., 1991]
"Multistage negotiation for distributed constraint satisfaction"
IEEE Transactions *on System,* Man and Cybernetics, Vol.21, No.6, 1991, pp.1462-1477.

[Conway R.N, 1965]
"Priority Dispatching and Work-in-Process Inventory in a Job Shop"
Industrial Engineering, Vol.29, No.11, 1965, pp.123-130.

[Conway R.N., Maxwell W.L., and Miller L.W., 1967]
"Theory of Scheduling"
Addison Wesley, 1967.

[Costa A. and Garetti M., 1985]
"Design of a control system for a flexible manufacturing cell"
Journal of Manufacturing Systems, Vol.4, No.1, 1985, pp.65-84.

[Crowe T; and Stahlman E.J.; 1995]
"A proposed structure for distributed shop floor control"
Integrated Manufacturing Systems, Vol.6, No.6, 1995, pp.31-36.

[Cummins M. Philip M., 1999]
"LAN Technologies Explained"
Butterworth-Heinemann, ISBN: 1555581927

[Das S.K. and Nagendra P., 1993]
"Investigations into the Impact of Flexibility on Manufacturing Performance"
International Journal of Production Research, Vol.31, No.10, 1993, pp.2337-2354.

[Daouas T., Ghedira K. and Muller J.P., 1995]
"How to schedule a flow shop plant by agents"
Applications of Artificial Intelligence in Engineering, Computational Mechanics Inc, Billerica, MA, 1995, pp.73-80.

[Dautenhahn K., 1998]
"The Art of Designing Socially Intelligent Agents Science, Fiction and the Human in the Loop"
Available at: http://citeseer.nj.nec.com/dautenhahn98art.html

[Davis L., 1985]
"Job Shop Scheduling with Genetic Algorithm"
Proceedings on an International Conference on Genetic Algorithms and Their Applications, 1985, pp136-140.

[Davis R., and Smith R.G., 1983]
"Negotiation as a Metaphor for Distributed Problem Solving"
Artificial Intelligence, 20, pp.63-109

[DeGarmo E.P, Black J.T, and Kohser R.A., 1988]
"Materials and Processes in Manufacturing"
7th edition, Macmillan Publishing Company, 1988.

[Dell'Amico M., and Trubian M., 1993]
"Applying Tabu Search to the Job Shop Scheduling Problem"
Annals of Operations Research, Vol.41, 1993, pp.231-252.

[Denzier D. R., and Boe W. J., 1987]
"Experimental Investigation of Flexible Manufacturing System Scheduling Rules"
International Journal of Production Research Vol.25, No.7, 1987, pp.979-994.

[Dessouky M.I., Moray N., and Kijowski B., 1995]
"Strategic Behavior and Scheduling Theory"
Human Factors, Vol.37, No.3, 1995, pp.443-472.

[DeToni A., Nassimbeni G., and Tonchia S., 1996]
"An artifficial, intelligece-based production scheduler"
Integrated Manufacturing Systems, Vol.7, No.3, 1996, pp.17-25.

[Dewan P. and Joshi S., 2001]
"Implementation of An Auction-Based Distributed Scheduling Model for A Dynamic Job Shop Environment"
International Journal of Computer Integrated Manufacturing, Vol.14 No.5, pp.446-456.

[Dilts D.M., Boyd N.P., and Whorms H.H., 1991]
"The Evolution of Control Architectures for Automated Manufacturing Systems"
Journal of Manufacturing Systems, Vol.10, No.1, 1991, pp.79-93.

[Drake, G. R. and Smith, J. S.; 1996]
"Simulation System for Real-time Planning, Scheduling, and Control"
Proceedings of the 1996 Winter Simulation Conference, Coronado, CA.

[Duffie N.A., 1982]
"An Approach to the Design of Distributed Machinery Control Systems"
IEEE Transactions on Industry Applications, Vol.1A-18, No.4, July/August 1982, pp.435-442.

[Duffie N.A., and Piper R.S., 1986a]
"Non - Hierarchical Control of Manufacturing Systems"
Journal of Manufacturing Systems, Vol.5, No.2, 1986, pp.137-139.

[Duffie N.A., Piper R.S., Humphrey B.J., and Hartwick J.Jr., 1986b]
"Hierarchical and Non - Hierarchical Manufacturing Cell Control with Dynamic Part - Oriented Scheduling"
Proceedings of NAMRC-XIV, Minneapolis, Minnesota, May, 1986, pp.504-507.

[Duffie N.A., and Piper R.S., 1987]
"Non - Hierarchical Control of a Flexible Manufacturing Cell"
Robotics and Computer Integrated Manufacturing, Vol.3, No.2, pp.173-179.

[Duffie N.A., Chitturi R., and Mou J.I., 1988]
"Fault-tolerant Heterarchical Control of Heterogeneous Manufacturing Entities"
Journal of Manufacturing Systems, Vol.7, No.4, 1988, pp.315-328.

[Duffie N.A., 1990]
"Synthesis of Heterarchical Manufacturing Systems"
In: Computer Science in Industry, Special Issue: Josef Hatvany Memorial: Total Integration - Analysis and Synthesis, Elsevier Science Publishers, 1990, pp 167-174.

[Duffie N.A. and Prabhu V.V., 1994]
"Real Time Distributed Scheduling of Heterarchical Manufacturing Systems"
Journal of Manufacturing Systems, Vol.13, No.2, 1994, pp94-107.

[Durfee E.H., Lesser V.R. and Corkil D., 1989]
"Cooperative distributed Problem Solving"
In: "The Handbook of Artificial Intelligence", edited by Barr A, Cohen P.R., and Feigenbaun E.A, Vol.4, Addison-Wasley, 1989, pp.83-147.

[Dwyer J. and Ioannou A., 1988]
"MAP and TOP, Advanced Manufacturing Communications"
John Wiley & Sons, Inc., New York, 1988.

[Esprit, 1993]
"CIMOSA: Open Systems Architecture for CIM"
Edited by Esprit Consortium AMICE, Second Edition, Berlin: Springer

[Farber D.J., Feldman J.; Heinrich F.R.; Hopwood M.D.; Larson K.C.; Loomis D.C.; and Rowe, L.A; 1973]
"The Distributed Computing System"
IEEE COMPCON Spring, 1973, pp.31-34.

[Ferber, J., 1999]
"Multi-Agent Systems, An Introduction to Distributed Artificial Intelligence"
Addison-Wesley, 1999.

[Finin T. , Fritzson R., McKay D., and McEntire R., 1994]
"KQML as an Agent Communication Language"
Proceedings of the 3rd International Conference on Information and Knowledge Management CIKM'94, ACM Press, Gaithersburg, MD, USA, pp.456-463.

[Finke G., 1995]
"Planning and Scheduling"
In: "Lectures notes in computer science 973", edited by Goos G., Hartmainis J, and Van Leeuwen J., 1995.

[Finin T., Fritzson R., McKay D., and McEntire R., 1994]
"KQML as an Agent Communication Language"
Proceedings of the 3rd International Conference on Information and Knowledge Management (CIKM'94). Also available at: http://citeseer.nj.nec.com/finin94kqml.html

[Finin T., Labrou Y., and Mayfield J., 1997]
"KQML as an Agent Communication Language"
In Software Agents. AAAI/The MIT Press, Cambridge, MA.

[FIPA, 2003]
Foundation for Intelligent Physical Agents
URL: http://www.fipa.org

[Fisher G., 1991]
"Computer Software for Flexible Manufacturing Systems"
In: "Handbook of Flexible Manufacturing Systems", edited by Jha N.K., Academic Press Inc., 1991, pp.61-87.

[Fletcher M., Garcia-Herreros E., Christensen J.H., Deen S.M., Mittmann R., 2000]
"An Open Architecture for Holonic Cooperation and Autonomy"
11th International Workshop on Database and Expert Systems Applications (DEXA'00), September 06 - 08, 2000, Greenwich, London, U.K. pp-224-230

[Fordyce K., Dunki-Jacobs R., Gerard B., Sell R., and Sullivan G., 1992]
"Advance Decision Support System for the Fourth Decision Tier Dispatch or Short-Interval Scheduling"
Production and Operations Management Vol.1, No.1, pp.70-86.

[Fordyce K. and Sullivan G.G., 1994]
"Logistics Management System (LMS): Integrating Decision Technologies for Dispatch Scheduling in Semiconductor Manufacturing"
In: "Intelligent Scheduling", edited by Morgan M. B., Morgan Kaufman Publishers, Inc., San Francisco, 1994, pp. 473-516.

[Franklin S., and Graesser A., 1997]
"Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agent"
Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages, published as Intelligent Agents III, Springer-Verlag, 1997, 21-35.

[French S., 1982]
Sequencing and Scheduling - An Introduction to the Mathematics of the Job-Shop,
Ellis Horwood, John-Wiley & Sons, New York.

[Fussell P.S., Wright P.K., and Bourne D., 1984]
"A Design of a Controller as a Component of a Robotic Manufacturing System"
Journal of Manufacturing Systems, Vol.3, No.1, 1984, pp.1-10.

[Gaalman G., Slomp J. and Suresh N., 1999]
"Towards an Integration of Process Planning and Production Planning and Control for Flexible Manufacturing Systems"
The International Journal of Flexible Manufacturing Systems, No.11, 1999, pp.5-17.

[Gaines B.R.; Norrie D.H.; Lapsley A.Z.; 1995]
"Mediator: an Intelligent Information System Supporting the Virtual Manufacturing Enterprise"
Proceedings of the 1995 IEEE International Conference on Systems, Man and Cybernetics, Vancouver, B.C., Oct. 22-25, 1995, pp. 964-969.

[Garey, M. R., Johnson, D. S. and Sethi, R., 1976]
"The Complexity of Flow Shop and Job-Shop Scheduling"
Mathematics of Operations Research, May, Vol.1, No.2, pp.117-129.

[Gaines B.R., Norrie D.H., and Lapsley A.Z., 1995].
"Mediator: an Intelligent Information System Supporting the Virtual Manufacturing Enterprise"
In Proceedings of 1995 IEEE International Conference on Systems, Man and Cybernetics, New York, pp. 964-969.

[General Magic, 1997]
"Odyssey Information"
URL: http://www.genmagic.com/technology/odyssey.html

[Genesereth, M.R., Fikes, R.E., 1992]
"Knowledge Interchange Format, Version 3.0, Reference Manual"
Computer Science Department, Stanford University, Technical Report Logic-92-1.
URL: http://www.cs.umbc.edu/agents/kse/kif

[Gensym, 1997]
"Agent Development Environment (ADE): Overview. Project Report, NCMS"
URL: http://www.gensym.com

[Gershwin S.B., Hildebrant R.R., Suri R., and Mitter S.K., 1986]
"A Control Perspective on Recent Trends in Manufacturing Systems"
Control System Magazine, MCS, Vol.6, No.2, pp.3-15, 1986.

[Ghosh S. and Gaimon C., 1992]
"Routing Flexibility and Production Scheduling in a Flexible Manufacturing System"
European Journal of Operational Research, Vol.60, No.3, 1992, pp.344-364.

[Gillespie D.M., 1992]
"The Architecture of an Execution Systems"
Autofact'92, pp18-7 to 18-20.

[Glover F., 1989]
"Tabu Search-Part 1"
ORSA Journal on Computing, Vol.1, 1989, pp.190-206.

[Glover F., 1990]
"Tabu Search-Part 2"
ORSA Journal on Computing, Vol.2, 1990, pp.4-32.

[Goldberg D.E., 1989]
"Genetic Algorithms in Search, Optimisation and Machine Learning"
Wiley 1989.

[Goldman S., Nagel R. and Preiss K., 1995]
"Agile Competitors and Virtual Organizations"
Van Nostrand Reinhold, New York, 1995.

[Goldratt M. E, 1990]
"What is this thing called theory of constraints and how should it be implemented?"
Croton-on-Hudson, New York, North River Press, 1990.

[Gomes C.P., Tate A., and Thomas L., 1994]
"Distributed scheduling framework"
Proceedings of IEEE International Conference on Tools with Artificial Intelligence, IEEE, Piscataway, NJ, 1994, pp.147-152.

[Gou, L., Luh, P. B. and Kyoya, Y., 1998]
"Holonic manufacturing scheduling: architecture, cooperation mechanism, and implementation."
Computer in Industrial, Vol.37, pp. 213-231.

[Gowan J.A., and Mathieu, 1993]
"Critical success factors for information systems development in fully-automated FMS environment."
Proceedings of the 1993 Annual Meeting of the decision Sciences institute, pp.1387-1399.

[Grabot B. and Geneste L., 1994]
"Dispatching Rules in Scheduling: A Fuzzy Approach"
International Journal of Production Research, Vol.32, No4, 903-915.

[Groover, M.P., 1994]
"Automation"
In: "Handbook of design, manufacturing and automation", edited by Dorf R.C. and Kusiak A. John Wiley & Sons, Inc., 1994, pp.3-21.

[Gruber T., 1993]
"*A Translation Approach to Portable Ontologies*"
Knowledge Acquisition, Vol.5, No.2, pp.199-220.

[Gu P., Balasubramanian S., and Norrie D.H., 1995]
"Bidding-based autonomous process planning and scheduling"
Proceedings of SPIE - The International Society for Optical Engineering, Society of Photo-optical Instrumentation Engineers, Bellingham, WA pp.184-194.

[Gunasekaran A., Martikainen T., and Yli-Olli P., 1995]
"Flexible Manufacturing Systems: An Investigation for Research and Applications"
In "Flexible Manufacturing Systems: Recent Developments" edited by Raouf A., and.Ben-Daya M., Elevier Science B.V., 1995, pp.3-44.
Also in: European Journal of Operational Research, Vol.66, 1993, pp.1-26.

[Gupta D; Buzacott J.A.; 1989]
"A Framework for Understanding Flexibility of Manufacturing Systems"
Journal of Manufacturing Systems, Vol.8, No.2, 1989, pp.89-97.

[Gupta Y.P., Gupta M.C., and Bector C.R., 1989]
"A review of scheduling rules in flexible manufacturing systems"
International Journal of Computer Integrated Manufacturing, No.2, pp.356-377.

[Guun T., 1987]
"Manufacturing for Competitive Advantage"
Ballinger Publishing, Mass. USA, 1987

[Hadavi K., et al., 1990]
"ReDS - A Dynamic Planning, Scheduling, and Control System for Manufacturing"
Journal of Manufacturing Systems, Vol.9, No.4, 1990, pp.332-344.

[Halevi, G. and Weill, R., 1980]
"Development of Flexible Optimum Process Planning Procedures"
Annals of the CIRP, Vol.29, 1980, pp.313-317.

[Hammer H., 1987]
"Flexible Manufacturing Cells and Systems with Computer Intelligence"
Robotics and Computer Integrated Manufacturing, Vol.3, No.1, 1987, pp.39-54.

[Hanks S., Pollack M.E., and Cohen P.R., 1993]
"Benchmarks, Test Beds, Controlled Experimentation, and the Design of Agent Architecture"
Artificial Intelligence Magazine, 1993, pp.32-40.

[Hatvany J., 1985]
"Intelligence and Cooperation in Heterarchical Manufacturing System"
Robotics and Computer Integrated Manufacturing, Vol.2, No.2, 1985, pp101-104.

[Hayes R.H. and Wheelright S.C., 1986]
"Restoring our competitive edge"
John Wiley & Sons, Inc., New York, 1986.

[Hitomi K., 1994]
"Analysis and Design of Manufacturing Systems"
In: "Handbook of design, manufacturing and automation", edited by Dorf R. and Kusiak A., John Wiley & sons, Inc, 1994, pp.405-433.

[HMS, web]
URL: http://www.profactor.at/hms/index.html

[Hoitomi D.J., Luh P.B., and Pattipati K.R., 1993]
"A Practical Approach to Job-Shop Scheduling Problems"
IEEE Transactions on Robotics and Automation, Vol.9, No.1, 1993, pp.1-13.

[Holstein W.K., 1968]
"Production Planning and Control Integrated"
Harvard Bus. Rev., Vol.46, May-June, 1968, pp.121-140.

[Hornberger R., 2001]
"Adapting to Net Economy"
presentation on the Energy E-Business Reality website,
URL: http://www.hourgroup.com/e-energy/reg2001.html.

[Howard W. J., 1996]
"Fast Ethernet - Down of a New Network"
Prentice-Hall, Inc., 1996.

[Hug Z., 1994]
"A Process Oriented Manufacturing System Simulation to Measure the Effects of Shop Control Factors"
Journal of Simulation, Vol.62, No.4, pp.218-228.

[Huhns M., editor, 1987]
"Distributed Artificial Intelligence"
Los Altos, Morgan Kaufmann, 1987.

[Hutchinson J., 1991]
"Current and Future Issues Concerning FMS Scheduling"
OMEGA, Vol.19, No.6, pp. 1991, pp.529-537.

[Hutchinson J., Leong K., Sniyder D., and Ward, P., 1991]
"Scheduling Approaches for Random Job Shop Flexible Manufacturing Systems"
International Journal of Productions Research, Vol.29, No.5, 1991, pp.1053-1067.

[Hutchinson G.K., and Pflughoeft K.A., 1994]
"Flexible Process Plans: Their Value in Flexible Automation Systems"
International Journal of Production Research, Vol.32, 1994, pp.707-719.

[Hynynen J. E., 1989]
"BOSS: An Artificially Intelligent System for Distributed Factory Scheduling"
In: "Computer Applications in Production and Engineering", edited by Kimura F. and Rolstadas A., Elsevier, 1989, pp.667-677.

[Hyun J.H. and Ahn B.H., 1993]
"A Unifying Framework for Manufacturing Flexibility"
Manufacturing Perspective, Vol.5, No.4, 1993, pp.251-260.

[IBM, 1998]
"Aglets Software Development Kit. IBM"
URL: http://www.trl.ibm.com/aglets

[IMS international, web]
"Holonic Manufacturing Systems Introduction"
URL: http://hms.ifw.uni-hannover.de

[Interrante L., and Goldsmith S, 1998]
"Emergent Agent-Based Scheduling of Manufacturing Systems"
Proceedings of Workshop on Agent-Based Manufacturing, 1998, pp.47-56.

[Jacobs F.R., and Bragg D.J, 1989]
"Repetitive Lots: Job Flow Considerations in Production Sequencing and Batch Sizing"
Decision Science, Vol.19, No.2, 1989, pp.281-294

[Jain A.S., and Meeran S., 1998]
"A state-of-the-art review of job-shop scheduling techniques"
Technical report, Department of Applied Physics, Electronic and Mechanical Engineering, University of Dundee, Dundee, Scotland, 1998.

[Jain A. S. and Meeran S., 1999]
"Deterministic job-shop scheduling: Past, present and future"
European Journal of Operational Research, Vol.113, No.2, pp.390-434

[Jennings N.R., Mamdani E.H., Laresgoiti I., and Corera J., 1992]
"GRATE: A General Framework for Co-operative Problem Solving"
IEE-BCS Journal of Intelligent System Engineering 1:2 (Winter), 1992, pp.102.114.

[Jennings N., 1995]
"Controlling Cooperative Problem Solving in Industrial Multi-Agent Systems Using Joint Intentions"
Artificial Intelligence, Vol.75, No.2, pp.195-240, 1995.

[Jennings N., 1996]
"Coordination Techniques for Distributed Artificial Intelligence"
in Foundations of Distributed Artificial Intelligence, G. M. P. Eds.: Ohare and N. R. Jennings, John Wiley & Sons, 1996, pp. 429-447.

[Jennings, N.R. and Wooldridge, M.J., 1998]
"Applications of Intelligent Agents"
In Agent Technology: Foundations, Applications, and Markets. Jennings, N.R. and Wooldridge, M.J (Eds.), Springer, pp.3-28. Also available at: http://www.elec.qmw.ac.uk/dai

[Jennings, N., Sycara K., and Wooldridge M., 1998]
"A Roadmap of Agent Research and Development,"
Autonomous Agents and Multi-Agent Systems, Vol.1, No.1, pp.7-38.

[Johnson S. M., 1954]
"Optimal Two and Three-Stage Production Schedules with Setup Time Included"
Naval Research Logistics Quarterly 1, 1954, pp.61-68.

[Jones A.T., and McLean C.R., 1986]
"A Proposed Hierarchical Control Model for Automated Manufacturing Systems"
Journal of Manufacturing Systems, Vol.5, No.1, 1986, pp.15-25.

[Jones A., and Saleh A., 1990]
"A Multi-Level / Multi-Layer Architecture for Intelligent Shop Floor Control"
International Journal of Computer Integrated Manufacturing, Vol.3, No.1, 1990, pp.60-70.

[Jones, A. and Rabelo, J. C., 1998]
"Survey of Job Shop Scheduling Techniques"
NISTIR, National Institute of Standards and Technology, Gaithersburg, MD.1998

[Kaighobadi M. and Venkatesh K.; 1994]
"Flexible Manufacturing Systems: An overview"
International Journal of Operation & Production Management, Vol.14, No.4, pp.26-49.

[Kanchanasevee P., Biswas G., Kawamura K., and Tamura S. 1997]
"Contract-Net Based Scheduling for Holonic Manufacturing Systems",
Proceedings of SPIE, "Architectures, Networks, and Intelligent Systems for Manufacturing Integration", Pittsburgh, Pennsylvania, 15-16 October 1997, pp. 108-115.

[Keilmann K.P., 1995]
"Multi Agent Systems – a natural trend in CIM"
In: "Lectures notes in computer science", edited by Goos G., Hartmanis J. and Van Leeuwen J., pp548-562.

[Khoshnevis B., and Chen Q., 1990]
"Integration of Process Planning and Scheduling Function"
Journal of Intelligent Manufacturing, Vol.1, 1991, pp.165-176.

[Kidd P., 1994]
"Agile Manufacturing: Forging New Frontiers"
Addison Wesley, 1994.

[Kim M.H., and Kim Y.D., 1994]
"Simulation-Based Real-Time Scheduling in a Flexible Manufacturing Systems"
Journal of Manufacturing Systems, Vol.13, No.2, 1994, pp85-93

[Kirkpatrick S., Gelatt C.D., and Vecchi M.P., 1983]
"Optimisation by Simulated Annealing"
Science, Vol.220, No.4598, 1983, pp.671-680.

[Kochhar A.; and McGarrie B., 1992]
"Identification of requirements of manufacturing control system: key characteristic approach"
Integrated Manufacturing Systems, Vol. 3, No. 2, 1992, pp.4-14.

[Kochhar A.K., 1997]
"Shifting Manufacturing Paradigms - Implementing Lean and Agile Manufacturing systems"
World Manufacturing Congress, Manufacturing Systems, New Zealand, 1997, pp.18-24.

[Koestler A., 1967]
"The Ghost in the Machine"
Arkana Books, 1967.

[Kolonko M., 1999]
"Some new results on simulated annealing applied to job shop scheduling problem",
European Journal of Operational Research, Vol.113, pp.123-136

[Krogh C., 1996]
"The Rights of Agents"
In Intellgent Agents II: Agent Theories, Architectures and Languages, Editors: M. Wooldridge, J.P.Muller, and M. Tambe, printed by Springer, 1996, ISBN 3-540-60805-2

[Kusiak A. 1987]
"Artificial Intelligence and Operations Research in Flexible Manufacturing Systems"
INFOR Journal Vol.25, No.1, 1987, pp.-7-10.

[Kusiak A. 1990]
"Intelligent Manufacturing Systems"
Englewood Cliffs, New Jersey: Prentice-Hall 1990.

[Kwok A.d. and Norrie D.H.; 1994]
"A Development System for Intelligent Agent Manufacturing Software"
Integrated Manufacturing Systems, Vol.5, No.4/5, 1994, pp.64-76.

[Labrou Y, 2001]
"Standardizing Agent Communication"
Lecture Notes in Computer Science, Vol.2086, pp.74-98.

[Larhoveen P.J.M.V., Aarts E.H.L., and Lenstra J.K., 1992]
"Job Shop Scheduling by Simulated Annealing"
Operations Research, Vol.40, No.1, 1992, pp.113-125.

[Larsen N.E. and Alting L., 1992]
"Dynamic Planning Enriches Concurrent Process and Production Planning"
International Journal of Production Research, Vol.30, 1992, pp.1861-1876.

[Lathon R.D., Claassen A.F., Rochowiak D.M., and Interrante L.D., 1994]
"Negotiation among scheduling agents to achieve global production goals"
Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Vol.2, pp.1541-1546.

[Lee C.K., Mansfieled W.H.Jr., and Sheth A.P., 1993]
"A framework for Controlling Cooperative Agents"
IEEE Computer, 26:7, pp.8-16.

[Lee, R. C. and J. M. Moore, 1967]
"CORELAP - Computerised Relationship Layout Planning"
Journal of Industrial Engineering, Vol.18, No.3, pp.195-200.

[Leito P. and Restivo F., 2000]
"A framework for Distributed Manufacturing Applications",
In: Proceedings of ASI'2000 International Conference, Bordeaux, France, September 2000.

[Lenz, J.E., 1994]
"Management of Production Cells"
In: "Handbook of design, manufacturing and automation", edited by Dorf R.C. and Kusiak A., John Wiley & Sons, Inc., 1994, pp.509-528.

[Leung Y.T.; and Sheen G.J.; 1993]
"Resolving Deadlocks in Flexible Manufacturing Cells"
Journal of Manufacturing Systems, Vol.12, No.4, 1993, pp291-304.

[Levy R.; Erol K.; and Mitchell H.; 1996]
"A Study of Infra Structure Requirements and Software Platforms for Autonomous Agents"
In CSE'96, July 9th, 1996. Also available at: http://www.aaria.uc.edu/paper.html

[Lewis W.C. Jr, 1981]
"Data Flow Architectures for Distributed Control of Computer Oriented Manufacturing Systems: Structure and Simulated Application"
PhD dissertation, School of Industrial Engineering, Purdue University, West Lafayette, IN, May, 1981.

[Lewis W., Barash M.M., and Solberg J.J., 1987]
"Computer Integrated Manufacturing System Control: Data Flow Approach"
Journal of Manufacturing Systems, Vol.6, No.3, 1987, pp.177-191.

[Lin G.Y. and Solberg J.J., 1992]
"Integrated shop floor control using autonomous agents"
IIE Transactions, Vol.24, No.3, pp.57-71.

[Lin G.Y. and Solberg J.J., 1994]
"An Agent-Based Flexible Routing Manufacturing Control Simulation System"
Proc. of the 1994 Winter Simulation Conference, pp.970-977.

[Luck M. and d'Inverno M, 2001]
"A Conceptual Framework for Agent Definition and Development",
The Computer Journal, Vol.44, No.1, pp.1-20.

[Looveren A.J. Van, Gelders L.F., Wassenhove L.N. Van, 1986]
"A review of FMS planning models"
In: "Modelling and Design of Flexible Manufacturing Systems", edited by Kusiak A., Amsterdam: Elsevier 1986, pp. 3-31.

[Lyons D.M., and Hendrinks A.J., 1992]
"Planning, Reactive"
Encyclopedia of Artificial Intelligence (2nd ed.) John Wiley, pp.1171-1181.

[Macchiaroli R. and Riemma S., 2002]
"A negotiation scheme for autonomous agents in job shop scheduling"
International Journal of Computer Integrated Manufacturing, 2002, Vol.15, No.3, 222–232.

[Madureira A., Ramos C, and Silva S., 2001]
"A Genetic Approach for Dynamic Job-Shop Scheduling Problems"
MIC'2001 - 4th Meta-heuristics International Conference, Porto, Portugal, July 16-20, 2001

[Mahmoodi F., Mosier C.T., and Morgan J.R., 1999]
"The Effects of Scheduling Rules and Routing Flexibility on the Performance of Random Flexible Manufacturing Systems"
The International Journal of Manufacturing Systems, No.11, 1999, pp.271-289.

[Malakooti B., 1989]
"A Hierarchical, Multi-Objective approach to the Analysis, Design, and Selection of Computer Integrated Manufacturing Systems"
Robotics and Computer Integrated Manufacturing, Vol.6, No.1, 1989, pp.83-97.

[Maley J.G., 1988]
"Managing the Flow of Intelligent Parts"
Robotics and Computer Integrated Manufacturing, Vol.4, No.3/4, 1988, pp.525-530.

[Malmborg C.J., 1994]
"Facility Design: Block Layout Planning Process for Manufacturing Systems"
In "Handbook of design, manufacturing and automation", edited by Doorf R. and Kusiak A., John Wiley & Sons, Inc, 1994, pp.461-479.

[Mandelbrot B.B., 1982]
"The Fractal Geometry of Nature"
San Francisco. W. H. Freeman (1982)

[Maríк V, and Pechoucek M., 2000]
"HOLOMAS'00: Industrial Applications of Holonic and Multi-Agent Systems Applications"
Proceedings of the eleventh International workshop on database and expert system application (DEXA'00), 2000.

[Martin D.L., Cheyer A.J. and Moran D.B., 1998].
"Building Distributed Software Systems with the Open Agent Architecture"
In Proceedings of PAAM'98, London.
URL: http://www.ai.sri.com/~oaa

[Mather H., 1986]
"MRP-II Won't Schedule the Factory of the Future"
CIM Rev. Vol.3, No.1, pp.64-68, 1986

[Maturana F.P. and Norrie D.H., 1995]
"A Generic Mediator for Multi-Agent Coordination in a Distributed Manufacturing System"
Proceedings of the 1995 IEEE International Conference on Systems, Man and Cybernetics, Vancouver, B.C., Oct. 22-25, 1995, pp. 952-957.

[Maturana F.P. and Norrie D.H., 1996]
"Multi-agent Mediator architecture for distributed manufacturing"
Journal of Intelligent Manufacturing, Vol.7, 1996, pp.257-270.

[Maturana F., Shen W., Norrie D.H., 1999]
"MetaMorph: an adaptive agent-based architecture for intelligent manufacturing"
International Journal of Production Research, 1999, Vol.37, No.10, pp.2159- 2173.

[Mayfield J., Labrou Y., and Finin T., 1996]
"Evaluation of KQML as an Agent Communication Language"
In Intelligent Agents II: Agent Theories, Architectures and Languages, Lecture Notes in Artificial Intelligence. Springer-Verlag, Berlin, vol. 1037, pp.347–360.

[McCarthy I., 1995]
"Manufacturing Classification: Lessons from Organizational Systematics and Biological Taxonomy"
Integrated Manufacturing Systems, Vol.6, No.6, 1995, pp.37-48.

[McEleney, B., O'Hare, G.M.P., and Sampson, J., 1998].
"An Agent Based System for Reducing Changeover Delays in a Job-Shop Factory Environment"
In Proc. of PAAM'98, London. Also available at: http://www.co.umist.ac.uk

[McFarlane D., and Bussmann S, 2000]
"Developments in Holonic Production Planning and Control"
International Journal of Production Planning and Control, Vol.11, No.6, pp. 522–536.

[McKay K.N., Safayeni F.R., and Buzacott J.A., 1995]
"A Review of Hierarchical Production Planning and Its Applicability for Modern Manufacturing"
Production Planning and Control, Vol.6, No.5, 1995, pp.384-394.

[McKay K.N., and Wiers C.S., 1999]
"Unifying the Theory and Practice of Production Scheduling"
Journal of Manufacturing Systems, Vol.18, No.4, 1999, pp.241-255.

[Mehra, A. and Nissen, M., 1998]
"Intelligent Supply Chain Agents using ADE"
In Proceedings of AI & Manufacturing Research Planning Workshop, Albuquerque, NM, The AAAI Press, pp. 112-119.

[Miilar H.H. and Yang, T., 1996]
"Batch Sizes and Lead-Time Performance in Flexible Manufacturing Systems"
International Journal of Flexible Manufacturing Systems, Vol. 8, No.1, 1996, pp.5-21.

[Mo J.P.T., Wang Yamin; and Tang Chi Kang; 1996]
"Use of the Virtual Manufacturing Device in the Manufacturing Massage Specification Protocol for Robot Task Control"
Computers in Industry, Vol.28, No.2, 1996, pp.123-136.

[Mooney E.L., and Rardin R.L., 1993]
"Tabu Search for a Class of Scheduling Problems"
Annals of Operations Research, Vol.41, 1993, pp.253.

[Morley R.E., and Schelberg C., 1993]
"An Analysis of Plant-Specific Dynamic Scheduler"
Proceedings of the NSF Workshop on Dynamic Scheduling, Cocoa Beach, FL

[Mosier C. T.; Elvers D. A., and Kelly D., 1984]
"Analysis of Group Technology Scheduling Heuristics"
International Journal of Production Research Vol.22, No.5, 1984, pp.857-875.

[Muller, H.J., 1994]
"Negotiation Principles"
In "Foundations of Distributed Artificial Intelligence", edited by O'Hare and Jennings, 1994.

[Naugle G. M., 1994]
"Network Protocol Handbook"
McGraw-Hill, Inc. Series on Computer Communication.

[Nawoz M.A., Enscoci M., and Ham E.E., 1983]
"A Heuristic Algorithm for the M Machine-N Job Flow Shop Sequencing Problem"
Omega Vol.41, No.1, 1983.

[Naylor A.W., and Volz R.A., 1987]
"Design of Intelligent Manufacturing System Control Software"
IEEE Transactions on Systems, Man, And Cybernetics, Vol.17, No.6, 1987, pp.881-897.

[Niebel B. W., Drapper A.B., and Wysk R.A., 1989]
"Modern Manufacturing Process Engineering"
McGraw-Hill, New York, 1989.

[Nii H.P., 1986]
"Blackboard Systems"
AI Magazine, Vol.7, No.3, pp.40-53 and Vol.7, No.4, pp.82-107.

[ObjectSpace, 1997]
"Agent-Enhanced Manufacturing System Initiative: Project Brief. ObjectSpace"
URL: http://www.atp.nist.gov/www/comps/briefs/97050018.htm

[Odell J., Parunak H., and Bock C., 1999]
"Representing agent interaction protocols in UML"
In OMG Document ad/99-12-01. Intellicorp Inc., December 1999.

[O'Grady P.J., 1986]
"Controlling Automated Manufacturing Systems"
Kogan Page Ltd., London, 1986

[Okino N., 1989]
"Bionic Manufacturing Systems -- Modelon-Based Approach"
In Proceedings of the CAM-I (Computer Aided Manufacturing - International Inc) 18th Annual International Conference, New Orleans, Louisiana, pp.485-492.

[Okino N.,1992]
"A Prototyping of Bionic Manufacturing System"
In Proceedings of the International Conference on Object Oriented Manufacturing Systems, Calgary, Alberta, pp.297-302.

[Okino N., 1993]
"Bionic Manufacturing System"
In Flexible Manufacturing Systems Past - Present - Future, CIRP, J. Peklenik (ed.), pp.73-95.

[Orlicky J., 1975]
"Materials Requirements Planning"
McGraw-Hill Book Co., Inc., New York, 1975.

[Osman I. H. and Kelly, J. P., 1996]
"Meta-Heuristics: An Overview"
in Osman, I. H. and Kelly, J. P. *Meta-Heuristics: Theory and Applications*, Kluwer Academic Publishers, Norwell, MA, USA, Chapter 1, pp.1-21.

[Ottaway, T.A. and Burns, J.R., 2000]
"An Adaptive Production Control System Utilizing Agent Technology"
International Journal of Production Research, 38(4), 721-737.

[Ouelhadj D., Hanachi C., and Bouzouia B., 1998]
"A Multi-agent system for dynamic scheduling and control in manufacturing cells"
In the Proceedings of the IEEE International Conference on Robotics and Automation, ICRA'1998, May 1998, Louvain, Belgium, pp. 1256-1262.

[Ouelhadj D., Hanachi C., and Bouzouia B., 1999]
"A Multi-contract net protocol for dynamic scheduling in manufacturing systems"
In the Proceedings the IEEE International Conference on Robotics and Automation, ICRA'1999, Detroit, USA, pp. 1114-1120.

[Ovacik I.M., and Uzsoy R., 1994]
"Exploiting Shop Floor Status Information to Schedule Complex Job Shops"
Journal of Manufacturing Systems, Vol.13, No.2, 1994, pp73-84.

[Overmars A. and Toncich D., 1996]
"Hybrid FMS control architectures based on holonic principles"
International Journal of Flexible Manufacturing Systems, Vol.8, pp. 263-278.

[Panwalkar S.S., and Iskander W., 1977]
"A Survey of scheduling Rules"
Operations Research, Vol.25, No.1, 1987, pp.45-61.

[Park P.S., 1991]
"The Examination of Worker Cross-Training in a Dual Resource Constrained Job Shop"
European Journal of Operational Research, Vol.52, No.3, 1991, pp.291-299.

[Parunak, H.V.D., 1987]
"Manufacturing experience with the contract net"
Distributed Artificial Intelligence (London: Michael M. Huhns, Pitman) pp. 285 – 310.

[Parunak H.V.D., and White J.F., 1987a]
"A Synthesis of Factory Reference Models"
Proceedings of the IEEE Workshop on Languages for Automation, 1987, pp.109-112.

[Parunak H.V.D., 1987b]
"Manufacturing Experience with the Contract Net"
In: "Distributed Artificial Intelligence", edited by Huhns M.N., Pitman, London, 1987, pp.285-310.

[Parunak H.V.D., Kindrick J., and Irish B.W., 1987c]
"Material Handling: A conservative Domain for Neural Connectivity and Propagation"
Proceedings of the 6th National Conference on Artificial Intelligence, AAAI, 1987, pp.307-311.

[Parunak H.V.D., Kindrick J., and Irish B.W., 1988]
"A Connectionist Model for Material Handling"
Robotics and Computer Integrated Manufacturing, Vol.4, No.3/4, 1988, pp.643-654.

[Parunak H.V.D., 1990]
"Distributed AI and Manufacturing Control: Some Issues and Insights"
In: "Decentralised AI", edited by Demazeau Y. and Mueller J.P., North-Holland, 1990, pp.81-104.

[Parunak H.V.D., 1991]
"Characterising the Manufacturing Scheduling Problem"
Journal of Manufacturing Systems, Vol.10, No.3, 1991, pp.241-259.

[Parunak, H.V.D., 1994]
"Applications of Distributed Artificial Intelligence in Industry"
White Paper, available at: http://www.iti.org/~van/jennings.ps

[Parunak, H.V.D., 1995a]
"Autonomous Agent Architectures: A Non-Technical Introduction"
White Paper, available at: http://www.iti.org/~van/nontech.ps

[Parunak, H.V.D., 1995b]
"MASCOT: A Virtual Factory for Research and Development in Manufacturing Scheduling and Control"
White Paper, available at: http://www.iti.org/~van/testbed.ps

[Parunak, H.V.D., 1996a]
"Workshop Report: Implementing Manufacturing Agents"
White Paper, available at: http://www.iti.org/~van/paamncms.ps

[Parunak, H.V.D., 1996b]
"The Heartbeat of the Factory: Understanding the Dynamics of Agile Manufacturing Enterprises"
White Paper, available at: http://www.iti.org/~van/heartbt.ps

[Parunak H.V.D, Baker A.D., and Clark S.J., 1997a]
"The AARIA Agent Architecture: An Example of Requirements-Driven Agent-Based System Design"
Proceeding of the First International Conference on Autonomous Agents (ICAA'97), Marina del Rey, CA, February 6-8, 1997.
White Paper, available at http://www.iti.org/~van/aaria.ps

[Parunak H.V.D., Baker A., and Erol K., 1997b]
"Manufacturing over the Internet and into Your Living Room"
White Paper, available at: http://www.aaria.uc.edu/cybermfg.ps

[Parunak, H.V.D., 1997c]
"Go to the Ant: Engineering Principles from Natural Agent Systems"
White Paper, available at http://www.iti.org/~van/gotoant.ps

[Parunak, H.V.D, 1998a]
"Characterizing Multi-Agent Negotiation"
White Paper, available at: http://www.erim.org/~vparunak/IWMAS98_Characterize.pdf

[Parunak H.V.D., Baker A.D., and Clark S.J., 1998b]
"The AARIA Agent Architecture: From Manufacturing Requirements to Agent-Based System Design"
Workshop on Agent-Based Manufacturing, ICAA'98, Minneapolis, MN, 10 May 1998.

[Parunak, H.V.D, 1998c]
"What can Agents do in Industry, and Why? An Overview of Industrially-Oriented R&D at CEC"
Industrial Technology Institute; CIA'98
White paper, available at: http://www.erim.org/~vparunak/cia98.pdf

[Parunak, H.V.D., 2000a]
"Agents in Overalls: Experiences and Issues in the Development and Deployment of Industrial Agent-Based Systems"
International Journal of Cooperative Information Systems, Vol.9, No.3, 2000, pp 209-227.

[Parunak H.V.D., 2000b]
"Autonomous Agents and Multi-Agent Systems"
ERIM. 3:4 (2000) 389-407 8/14/00 10:41
URL: http://citeseer.nj.nec.com/541629.html

[Parunak H. V. D., 2000c]
"A Practitioners? Review of Industrial Agent Applications".
Autonomous Agents and Multi-Agent Systems Vol.3, No.4. pp.389-407

[Pattison T., 2000]
"Programming Distributed Application with COM+ and MS Visual Basic 6.0"
Second Edition, Microsoft Programming Press, 2000.

[Pedgen C.; Shannon R.; and Sadowski R. P., 1990]
"*Introduction to Simulation Using Siman*",
Hightstown, NJ: McGraw-Hill, Inc., 1990.

[Peng Y., Finin T., Labrou Y., Chu B., Long J., Tolone W. J. and Boughannam A., 1998]
"A Multi-Agent system for Enterprise Integration"
In proceedings of the 3rd International Conference on the Practical Applications of Agents and Multi-Agent Systems - PAAM-98, London, UK, pp.155-169.

[Peng Y., Finin T., Labrou Y., Cost R. Chu B., Long J., Tolone W. and Boughannam A., 1999]
"An agent-based approach for manufacturing integration - the CIIMPLEX experience"
International Journal of Applied Artificial Intelligence, Vol.13 No.1-2, pp.39-64.

[Philipoom P.R. and Fry T.D., 1993]
"Capacity-Based Order Review/Release Strategies to Improve Manufacturing Performance"
International Journal of Production Research, Vol.30, No.11, 1993, pp.2559-2572.

[Pickel J., 1988]
"Issues in the Design of Scheduling Systems"
In: "Expert Systems and Intelligent Manufacturing", edited by Oliff, M., New York, Elsevier Science, 1988, pp.70-89.

[Pierreval H. and Mebarki N., 1991]
"Dynamic Selection of Dispatching for Manufacturing System Scheduling"
International Journal of Production Research, Vol.35, No.6, 1997, pp.1575-1591.

[Pinson E., 1995]
"The Job-Shop Scheduling Problem: A Concise Survey and Some Recent Developments"
in: Scheduling Theory and its Applications, Chretienne P et al (eds), John Wiley & Sons Ltd, Ch.13, pp. 277-293.

[Pirlot M., 1996]
"General Local Search Methods"
European Journal of Operational Research, Vol 92, pp.493-511.

[Ponnambalam S.G., Aravindan P., and Sreenivasa P.R., 2001]
"Comparative evaluation of genetic algorithms for job-shop scheduling"
Production planning and control, 2001, Vol.12, No.6, pp.560-574

[Prabhu V, 2000]
"Distributed Control systems"
URL: http://www.ie.psu.edu/people/faculty/prabhu/dcon/distcon.htm

[Primentel J.R., 1990]
"Communication network for manufacturing"
Prentice Hall Inc. 1990

[Rabelo, R. and Camarinha-Matos L.M., 1994]
"Multi-Agent based Dynamic Scheduling"
Int. Journal on Robotics and Computer Integrated Manufacturing, Vol.2, No.4, pp.303-310.

[Rabiee M. M., 1999]
"Local Area Network (LAN) in Manufacturing"
Journal of Industrial Technology, Vol.15, No.2, February 1999 to April 1999

[Ramos C., 1994]
"An architecture and a negotiation protocol for the dynamic scheduling of manufacturing systems"
Proceedings of IEEE International Conference on Robotics and Automation, Part VI, pp.3161-3166.

[Rana S.P., and Taneja S.K., 1988]
"A distributed Architecture for Automated Manufacturing Systems"
International Journal of Advanced Manufacturing Technology, Vol.3, No.5, 1988, pp.81-98.

[Ranky P.; 1986]
"Computer integrated manufacturing"
Prentice Hall International, UK, Ltd, 1986.

[Rembold U., Blume, C., and Dillmann R., 1985]
"Computer Integrated Manufacturing - Technology and Systems"
Dekker, New York, 1985.

[Rosenschein J. S. and Zlotkin G., 1994]
"Rules of Encounter: Designing Conventions for Automated Negotiation among Computers"
Cambridge, MA, MIT Press, 1994.

[Roy D. and Anciaux D., 2001]
"Shop-floor control: a multi-agents approach"
International Journal of Computer integrated Manufacturing Vol.14, No.6, pp.535–544.

[Rumbaugh J., Blaha M., Premerlani W., Eddy F., and Lorensen W., 1991]
"Object-Oriented Modeling and Design"
Englewood Cliffs: Prentice Hall,, 1991.

[Russell J.S., and Norvig P., 1995]
"Artificial Intelligence: A modern Approach"
Englewood Cliffs, NJ: Prentice Hall, 1995.

[Saad A.; Kawamura K.; and Biswas G.; 1997]
"Performance evaluation of Contract Net Based Heterarchical Scheduling for Flexible Manufacturing Systems"
Intelligent Automation and Soft Computing, Vol.3, No.3, 1997, pp.229-248.

[Saad A.; Kawamura K.; Biswas G.; Johnson M.E.; Salama A.; 1995]
"Evaluating a contract net-based heterarchical scheduling approach for flexible manufacturing"
Proceedings of the IEEE International Symposium on Assembly and Task Planning Aug 10-11, 1995 Pittsburgh, PA, USA pp.147-152.

[Sabuncuoglu I. and Gurgun B., 1996]
"A Neural Network Model for Scheduling Problems",
European Journal of Operational Research, Vol.93, No.2, pp.288-299.

[Sandell N.R., Varaiya P., Athans M., and Safonov M.G., 1978]
"Survey of decentralized control methods for large-scale systems"
IEEE Transactions on Automatic Control, Vol. AC-23 No.2, 1978, pp.108-28.

[Sandholm T. W., 1996]
"Negotiation among Self-Interested Computationally Limited Agents"
University of Massachusetts, Computer Science, 1996.

[Sauter J. and Parunak H.V.D., 1999]
"Ants in the Supply Chain"
Workshop on Agent based Decision Support for Managing the Internet-Enabled Supply Chain, Agents 99, Settle, WA, 1 May 1999.

[Schlenoff C., Ivester R., Knutilla A., 1999]
"A robust process ontology for manufacturing system integration"
National Institute Of Standards and Technology Gaithersburg, MD 20899;
URL: http://www.ontology.org/main/papers/psl.html

[Searle J., 1969]
"Speech Acts"
Cambridge University Press, December 1969

[Sethi A.K., and Sethi S.P., 1990]
"Flexibility in Manufacturing - A Survey"
International Journal of flexible Manufacturing Systems, Vol.2, No.4, 1990, pp. 289-328.

[Shanker K., and Tzen Y.S., 1985]
"A Loading and Dispatching Problem in a Random Flexible Manufacturing System"
International Journal of Production Research, Vol.23, No.3, 1985, pp.579-595.

[Shaw M.J., and Whinston A.B., 1985a]
"Automatic Planning and Flexible Scheduling: A knowledge based approach"
Proceedings of the IEEE Conference on Robotics and Automation, 1985, pp.890-894.

[Shaw M.J., and Whinston A.B., 1985b]
"Application of Artificial Intelligence to Planning and Scheduling in Flexible Manufacturing"
In: "Flexible Manufacturing Systems: Methods and Studies", edited by Kusiak, A., Elsevier Science Publisher 1986, pp.223-242.

[Shaw M.J., and Whinston A.B., 1985c]
"Task Bidding and Distributed Planning in Flexible Manufacturing"
IEEE Proceedings of the Second International Conference on Artificial Intelligence Applications, Florida, Miami Beach, 1985, pp.184-189.

[Shaw M.J., 1987a]
"A Distributed Scheduling Method for Computer Integrated Manufacturing: The Use of Local Area Networks in Cellular Systems"
International Journal of Production Research, Vol.25, No.9, 1987, pp.1285-1303.

[Shaw M.J., 1987b]
"Distributed Planning in Cellular Flexible Manufacturing Systems"
INFOR Vol.25, No.1, 1987, pp13-25.

[Shaw M.J., 1988]
"A Dynamic Scheduling in Cellular Manufacturing Systems: A Framework for Networked Decision Making"
Journal of Manufacturing Systems, Vol.7, No.2, 1988, pp.83-93.

[Shaw M. J., 1989]
"FMS Scheduling as Cooperative Problem Solving"
Annals of Operations Research, Vol.17, 1989. pp.323-346.

[Shen W. and Norrie D.H., 1998]
"A Hybrid Agent-Oriented Infrastructure for Modelling Manufacturing Enterprises"
URL: http://ksi.cpsc.ucalgary.ca/KAW/KAW98/shen/index.html

[Shen W., Xue D., and Norrie D.H., 1998].
"An Agent-Based Manufacturing Enterprise Infrastructure for Distributed Integrated Intelligent Manufacturing Systems"
In Proceedings of the Third International Conference on the Practical Application of Intelligent Agents and Multi-Agents, PAAM'98, London, UK. March, 23-25, 1998.
Also available at: http://imsg.enme.ucalgary.ca

[Shen W. and Norrie D.H., 1999]
"Agent-Based Systems for Intelligent Manufacturing: A State-of-the-Art Survey"
An international Journal of Knowledge and information systems: Vol.1, No.2, pp. 129-156.

[Shoham Y., 1993]
"Agent-oriented programming"
Artificial Intelligence, Vol.60, No.1, 1993, pp.51-92.

[Sihn W., 1997]
"The Fractal Factory: A Practical Approach to Agility in Manufacturing"
Proceedings of The Second World Congress On Intelligent Manufacturing Processes & Systems, pp. 617-621, Budapest, Hungary, June 10-13, 1997.

[Sikora R. and Shaw M. J., 1997]
"Coordination Mechanisms for Multi-Agent Manufacturing Systems: Applications to Integrated Manufacturing Scheduling"
IEEE Transactions on Engineering Management, Vol.44, No.2, 1997, pp.175-187.

[Silva N. and Ramos C., 1999]
"Proposal for Inter-Enterprises Negotiation Infra-Structures using an Holonic approach"
Proceedings of 1st International IFAC Workshop on Multi-Agent Systems in Production; Wien, Austria; December 2-4 1999

[Silva N. and Rocha J., 2000]
"Requirements for Co-operation and Knowledge Sharing in Virtual Organizations"
Proceedings of the International ICSC Symposium on Multi-Agents and Mobile Agents in Virtual Organizations and E-Commerce - MAMA'2000; Wollongong, Australia;11-13 December 2000

[Simon, H.A., 1962]
"The Architecture of Complexity"
Proc. Am. Philos. Soc., Vol.106, No.6, 1962.

[Simoneau P., 1997]
"Hands-On TCP/IP"
McGraw-Hill, Inc. Series on Computer Communication.

[Simpson J.A., Hocken R.J., and Albus J.S., 1982]
"The Automated Manufacturing, Research Facility of the National Bureau of Standards"
Journal of Manufacturing Systems, Vol.1, No.1, 1982, pp.17-32.

[Skinner W., 1985]
"Manufacturing: The Formidable Competitive Weapon"
John Wiley & Sons, Inc., New York, 1985.

[Smith R.G., 1980]
"The Contract Net Protocol: High Level Communication and Control in a Distributed Problem Solver"
IEEE Transactions on Computers, Vol.C-29, No.12, 1980, pp.1104-1113.

[Smith R.G., and Davis R., 1981]
"Frameworks for Cooperation in Distributed Problem Solving"
IEEE Transactions on Systems, Man, and Cybernetics, Vol.11, No.1, 1981, pp.61-69.

[Smith S.F., Fox M.S., and Ow P.S., 1986]
"Constructing and Maintaining Detailed Production Plans: Investigations Into the Development of Knowledge-Based Factory Scheduling Systems"
AI Magazine Vol.7, No.4, 1986, pp.45-61.

[Smith S.F., and Hynynen J.E., 1987]
"Integrated Decentralisation of Production Management: An Approach for Factory Scheduling"
In: "Intelligent and Integrated Manufacturing Analysis and Synthesis", edited by Liu C.R., Requicha A., and Chandraseker S., New York: ASME, pp.427-439.

[Smith J.S.; 1996]
"Design and Implementation of FMS Control Software"
Proceedings of the 1996 FAIM Conference, Atlanta, GA, pp. 859-867.

[Smith J.S.; Hoberecht W.C., and Joshi S.B.; 1996]
"A Shop Floor Control Architecture for Computer Integrated Manufacturing"
IIE Transactions, Vol.28, No.10, October 1996, pp. 783-794.

[Smith J.S. and Joshi S.B.; 1995]
"A Shop Floor Controller Class for Computer Integrated Manufacturing"
International Journal of Computer Integrated Manufacturing, Vol.8, No.5, 1995.

[Smith S. F. and Becker M.A., 1997]
"An Ontology for Constructing Scheduling Systems"
Proceedings of AAAI Spring Symposium on Ontological Engineering, 1997. Also available at: http://www.cs.cmu.edu/afs/cs/project/ozone/www/AAAI_Symp_On_Ontol_97/abstract.html.

[Somendra P.; Laurie R.; and Cheng H.; 1994]
"Manufacturing information integration using a reference model"
International Journal of Operation and Production Management, Vol.14, No.11, 1994, pp.52-72.

[Sousa P. and Ramos C., 1998]
"A dynamic scheduling holon for manufacturing orders"
Journal of Intelligent Manufacturing, Vol.9, No.2, 1998, pp.107-112.

[Sousa P., Silva N., Heikkila T., Kollingbaum M., Valckenaers P., 1999]
"Aspects of Co-operation in Distributed Manufacturing Systems"
Proceedings of the 2nd International Workshop on Intelligent Manufacturing Systems - IMS-Europe'99; Leuven, Belgium, 22-24 September 1999.

[Spur G., Seliger G., and Viehweger B., 1986]
"Cell Concepts for Flexible Automated Manufacturing"
Journal of Manufacturing Systems, Vol.5, No.3, 1986, pp.171-179.

[Stamper A. D., 1994]
"Local Area Networks"
The Benjamin/Cummings Publishing Company, Inc.

[Stanford, 1997]
JATLite. CDR, Stanford University,
URL: http://java.stanford.edu

[Stecke K.E., and Solberg J.J., 1981]
"Loading and Control Polices for a Flexible Manufacturing System"
International Journal of Production Research, Vol.19, No.5, 1981, pp.481-490.

[Steinhofel K., Albrecht A., Wong C. K., 1999]
"Two simulated annealing-based heuristics for the job shop scheduling problem"
European Journal of Operational Research, Vol.118, pp.524-548

[Stern K.P., and Marney-Petix V.C., 1996]
"Mastering LAN Enabling Technologies"
Numidia Press, Fremont, CA 94536.

[Steventon, A., 1998]
FIPA Agent Communication Technologies and Services,
ACTS Project Ac317, June, 19, 1998
URL: http://news.cert.fr/francais/deri/cros/Cros/Texte/FactsOverview.ps

[Stuart R.J., and Norvig P., 1995]
"Artificial Intelligence: A modern Approach"
Englewood Cliffs, NJ: Prentice Hall

[Subhash C.S., and Sanchoy K.D., 1994]
"Computer Integrated Production Planning and Control"
In: "Handbook of Design, Manufacturing and Automation", edited by Dorf R. and Kusiak A.,
Wiley and Sons, Inc, New York, 1994, pp.483-508.

[Subramanyam S., and Askin R.G., 1986]
"An Expert Systems Approach to Scheduling in Flexible Manufacturing Systems"
In: "Flexible Manufacturing Systems: Methods and Studies", edited by Kusiak A.,
Elsevier Science Publisher 1986, pp.243-256.

[Suda, 1989]
"Future Factory System Formulated in Japan (1)"
In: Techno Japan, Vol.22, No.10, pp.15-25.

[Suda, 1990]
"Future Factory System Formulated in Japan (2)"
In: Techno Japan, Vol.23, No.3, pp.51-61.

[Sugimori Y., Kusunoki K., Cho F., and Uchikawa S., 1977]
"Toyota Production System and KANBAN System: Materialisation of Just-In-Time and Respect for Human System"
International Journal of Production Research, Vol.15, No.6, 1977, pp.553-564.

[Suri R., and Hildebrant R.R., 1984]
"Modelling Flexible Manufacturing Systems Using Mean-Value Analysis"
Journal of Manufacturing Systems, Vol.3, No.1, 1984, pp.27-38.

[Sycara K.P., Roth S.F., Sadeh N. and Fox M., 1991a]
"Distributed constrained heuristic search"
IEEE Transactions on Systems, Man and Cybernetics, Vol.21, 1991, pp.1446-1461.

[Sycara K.P.; Roth S.F.; Sadeh N.; and Fox M.; 1991b]
"Resource Allocation in Distributed Factory Scheduling"
IEEE Expert, Vol.6, No.1, 1991, pp.29-40.

[Tanenbaum A.S., 1996]
"Computer Networks"
3rd Ed., Pearson Higher Education, ISBN: 0133499456

[Tawegoum, R.; Castelain, E.; Gentina, J.C., 1994]
"Real-time piloting of flexible manufacturing systems"
In: European Journal of Operational Research, Vol. 78, No. 2, pages 252-261. 1994.

[Tharumarajah A., Wells A., and Nemes L., 1996]
"Comparison of the Bionic, Fractal and Holonic manufacturing systems concepts"
in International Journal of Computer Integrated Manufacturing, 1996, Vol.9, No.3, pp 217-226.

[Tharumarajah, A.; Bemelman, R.; 1997]
"Approaches and Issues in Scheduling a Distributed Shop-Floor Environment"
Computers in Industry, Vol.34, No.1, 1997, pp.95-109.

[Tilley K. J., and Williams D. J., 1992]
"Modelling of Communications and Control in an Auctionbased Manufacturing Control System"
Proceedings of IEEE International Conference on Robotics and Automation, IEEE, 1992, pp 962-967.

[Tonshoff, H.-K., I. Seilonen, G. Teunis and P. Leitão, 2000]
"A Mediator-based approach for decentralized production planning, scheduling and monitoring"
In Proc. Of Int. Seminar on Intelligent Computation in Manufacturing Engineering, Capri, Italy, 2000, pp.113-118.

[Tsubone H., and Horikawa M., 1999]
"A Comparison Between Machine Flexibility and Routing Flexibility"
The International Journal of Flexible Manufacturing Systems, No.11, 1999, pp.83-101.

[Tsukada T.K., and Shin K. G., 1994]
"Polite rescheduling: Responding to local schedule disruptions in distributed manufacturing systems",
In Proceeding of the IEEE International Conference on Robotics ans Automation, pp. 1986-1991.

[Tsukada, T.K., and Shin, K.G., 1996]
"PRIAM: Polite Rescheduler for Intelligent Automated Manufacturing"
IEEE Transactions on Robotics and Automation, Vol.12, No.2, 1996, pp.235-245.

[Ueda, N., 1993]
"A Genetic Approach toward Future Manufacturing Systems"
In Flexible Manufacturing Systems Past – Present - Future, CIRP, J. Peklenik (ed), pp.211-228.

[Ulieru M., Stefanoiu D., and Norrie D., 2000]
"Holonic Metamorphic Architectures for Manufacturing: Identifying Holonic Structures in Multi-Agent Systems by Fuzzy Modelling"
Invited Chapter 3 in Handbook of Computational Intelligence in Design and Manufacturing (Jun Wang & Andrew Kussiak – Editors), CRC Press 2000, ISBN No 0-8493-0592-6, pp. 3-1 – 3-36.

[Ulieru, M., 2001]
"Web-Centric Diagnosis and Prediction System for Global Manufacturing"
IFSA-NAFIPS Joint Conference on Fuzzy Systems 2001, Vancouver, Canada, July 24-27, 2001.

[Ulieru M., 2002]
"Emergence of Holonic Enterprises from Multi-Agent Systems: A Fuzzy Evolutionary Approach"
Soft Computing Agents V. Loia (Ed.) IOS Press, 2002, Chapter 8, pp.187-215.

[Upton D. M., Barash M. M., and Matheson A. M., 1991]
"Architectures and auctions in manufacturing"
International Journal of Computer Integrated Manufacturing, Vol.4, No.1, 1991, pp.23-33.

[Upton D.M., 1997]
"A Flexible Structure for Computer-Controlled Manufacturing System"
URL: http://www.people.hbs.edu/dupton/papers/organic/workingPaper.htm

[Usher j., and Wang Y.C., 2000]
"Negotiation Between Intelligent Agents For Manufacturing Control"
URL. http://citeseer.nj.nec.com/usher00negotiation.html

[Vaario J. and Ueda K., 1998]
"An emergent modeling method for dynamic scheduling"
Journal of Intelligent Manufacturing, Vol.9, No.2 (April), 1998, pp.129-140.

[Valckenaers, P., Brussel H, Bongaerts L., and Bonneville F., 1995]
"Programming, scheduling and control of flexible assembly systems"
Computers in Industry, vol. 26, pp. 209-218, 1995.

[Vakharia A.J., and Selim H., 1994]
"Group technology"
In "Handbook of design, manufacturing and automation", edited by Dorf R. and Kusiak A., John Wiley & Sons, Inc, 1994, pp.435-460.

[Vandaele, N. De Boeck L., 2003]
"Advanced Resource Planning"
Robotics and Computer Integrated Manufacturing No.19, pp.211–218

[Veeramani D., 1992]
"Task and Resource Allocation via Auctioning"
Proceedings of the 1992 Winter Simulation Conference, 1992.

[Veeramani D., Bhargava B., and Barash M.M., 1993]
"Information System Architecture for Heterarchical Control of Large FMSs"
Computer Integrated Manufacturing Systems, Vol.6, No.2, 1993, pp.76-92.

[Veeramani, D., 1994]
"Control of Manufacturing Systems"
In: "Handbook of design, manufacturing and automation", edited by Dorf R. and Kusiak A., John Wiley & Sons, Inc, 1994, pp.553-566.

[Veeramani, D. and Wang K.J.; 1997]
"Performance Analysis of Auction Based Distributed Shop-Floor Control Schemes from the Perspective of the Communication System"
International Journal of Flexible Manufacturing Systems, Vol. 9, No. 2, 1997, pp.121-143.

[Vollmann, E.T., Berry L.W., and Whybark D.C., 1992]
"Manufacturing Planning and Control Systems"
Business One Irwin, Third edition, Homewood, Illinois, 1992.

[Wang H., 1986]
"An Experimental Analysis of the Flexible Manufacturing System (FMS)"
In: "Flexible Manufacturing Systems: Methods and Studies", edited by Kusiak A., Elsevier Science Publisher, 1986, pp.319-338.

[Wang L., 2000]
"Collaborative Design Approach For Holonic Manufacturing Systems"
URL: http://citeseer.nj.nec.com

[Wang Y.C. and Usher J., 2002]
"An Agent-Based Approach for Flexible Routing in Dynamic Job Shop Scheduling"
URL: http://citeseer.nj.nec.com

[Warnecke H.J., and Kuehnle H., 1992]
"Integration of CAPP and PPC – the PPC Part"
Annals of the CIRP, Vol.41, No.1, 1992, pp.15-18.

[Warnecke H. J., 1993]
"The Fractal Company: A Revolution in Corporate Culture"
Berlin, New York: Springer-Verlag, 1993

[Waters, G.A. 1991]
"Computer Communication Networks"
McGraw-Hill Education – Europe, ISBN: 0077073258

[Watson J.P., Beck J.C., Howe A.E., and Whitley L.D., 2003]
"Problem difficulty for tabu search in job-shop scheduling"
Artificial Intelligence, Vol.143, pp.187–217.

[Weatherall A., 1988]
"Computer Integrated Manufacturing: from fundamentals to implementation"
Butterworth & Co., 1988

[Weiderhold G., 1992]
"Mediators in the Architecture of Future Information Systems"
IEEE Transaction on Computer, 1992, pp.38-48.

[Wein L. M. and Chevalier P. B., 1992]
"A broader view of the job-shop scheduling algorithm",
Management Science, Vol.38, No.7, pp.1018-1033.

[Weiss, G., 1999]
"Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence"
The MIT Press, Cambridge, Massachusetts, 1999.

[Welgama, P. S. and P. R. Gibson, 1996]
"An integrated methodology for determination of layout and materials handling system"
International Journal of Production Research, Vol.34, No.8, pp.2247-2264.

[Wemmerlov U. 1987]
"Production Planning and Control Procedures for Cellular Manufacturing Systems: Concepts and Practice"
Falls Church, Va.: American Production and Inventory Control Society, 1987.

[Wemmerlov U., and Vakharia A.J., 1991]
"Job and Family Scheduling a Flow Line Manufacturing Cell: A Simulation Study"
IIE Transaction, December 1991.

[Widmer M., 1991]
"Job Sop Scheduling with Tooling Constraints: a Tabu Search Approach"
Journal of the Operational Society, Vol.42, 1991, pp.75-82.

[Wilhelm W.F., and Shin H.M., 1985]
"Effectiveness of Alternative Operations in a Flexible Manufacturing Systems"
International Journal of Production Research, Vol.23, 1985, pp.65-79.

[Womack, J.P., and Jones, D.T., 1996]
"Lean Thinking"
Simon and Schuster, New York, 1996.

[Wong, H. C., and Sycara, K, 1999]
"Adding Security and Trust to Multi-Agent Systems"
In Proceedings of Autonomous Agents '99 (Workshop on Deception, Fraud and Trust in Agent Societies). May 1999, Seattle, Washington, pp. 149-161.

[Wooldridge M., and Ciancarini P., 2000]
"Agent-Oriented Software Engineering: The State of the Art"
First Int. Workshop on Agent-Oriented Software Engineering, Springer-Verlag, Berlin, pp.1-28.

[Wright, O.W., 1984]
"MRP-II: Unlocking America's Productivity"
Wight Publications, Williston, 1984.

[Wu S.Y.D., and Wysk R.A., 1989]
"An Application of Discrete Event Simulation to Online Control and Scheduling in Flexible Manufacturing Systems"
International Journal of Production Research, Vol.27, 1989, pp.1603-1623.

[Wu D.J., and Sun Y, 2002]
"Cooperation in multi-agent bidding"
Decision Support Systems No.33 pp.335–347

[Wyns J, 1999]
"Reference architecture for Holonic Manufacturing Systems - the key to support evolution and reconfiguration"
K.U.Leuven, Belgium.
URL: http://www.mech.kuleuven.ac.be/~jwyns/phd/abstr.html

[Wysk R. A., et al, 1995]
"A Formal Process Planning Schema for Shop Floor Control"
Engineering Design & Automation, Vol.1, No.1, 1995.

[Yao D.D. and Pei F., 1990]
"Flexible Parts Routing in Manufacturing Systems"
IIE Transactions, Vol.22, No.1, 1990, pp.48-55.

[Young R.E., and Mayer R., 1984]
"The information dilemma: To conceptualise manufacturing as information process"
Industrial Engineering, Vol.16, No.9, 1984, pp.28-34.

[Zeestraten, M.J., 1990]
"The Look Ahead Dispatching Procedure"
International Journal of Production Research, Vol.28, No.2, 1990, pp.369-384.

[Zelenovic M.D., 1982]
"Flexibility - A Condition foe Effective Production Systems"
International Journal of Production Research, Vol.20, No.3, 1982, pp.319-337.

[Zhang, H.C. and Huang, S. H., 1995]
"Applications of Neural Networks in Manufacturing: A State-of-the-Art Survey,
International Journal of Production Research, Vol.33, No.3, 705-728.

[Zhang, X., Norrie, D.H., 1999]
"Holonic Control at the Production and Controller Levels"
IMS 99, Leuven, Belgium, Sept. 22-24, 1999, pp. 215-224.

Web sources:
http://www.cs.ucl.ac.uk/staff/S.Bhatti/D51-notes/notes.html
http://hms.ifw.uni-hannover.de
http://msdn.microsoft.com/vbasic
http://www.msdn.microsoft.com
http://www.informit.com/free_library

Appendix 1. Multi Threading

1.1 Introduction

The software developed in this thesis would need to be restructured if it was to be used in a commercial application, either for a large job shop with many work stations, or for reliable, faster than real time simulation of the system for evaluation of its performance. This appendix provides information on recent additions to Visual Basic which could be used to enhance its performance for the situations above.

Material presented here relates to some techniques (applicable in Visual Basic 6) that can be used to overcome problems that were encountered during development when Visual Basic versions 4 and 5 were used. Note that the latest Microsoft's Dot Net development platform provides yet another and entirely different development environment that offers even greater possibilities (for example, VB.Net supports free-threading model). The appendix explains why the structural changes in software are required and what important points should be considered in the future developments. The proposals are independent of the language or platform to be used in future developments.

1.2 The current design of a workstation agent - a monolithic application

The current workstation agent's application architecture (regardless of the modular design) follows the principles of a monolithic application. All the logic, resources, and the user interfaces needed to accomplish the entire programming task are found in one program. The user interface for defining data (forms which are part of an MS Access application) and the data themselves (data stored in the database tables) are separated from the other parts of the workstation agent but the program design still remains monolithic. Even in object oriented programming, where the parts of application (functions and features that are implemented, tested, and encapsulated in separate entities - objects) are cooperative, a program built from objects is still monolithic (refer to the Figure 1-1). In the current design of the workstation agent, the objects are an intrinsic part of the application. This means that all objects are internal to the one process (in serial manner) and they are not visible to the outside world (other applications). This is an important feature if some parts of the program need to be interconnected with the other applications, for example, with the software packages for modelling and simulation of workstation manufacturing activities such as QUEST for instance.

Within the workstation application, the user interface code is intermingled with program logic and data access code. This makes the workstation agents incapable of executing several parallel tasks (simulated processes) simultaneously, while at the same time being responsive to events which are triggered either from inside the system, (internal workstation events), or outside the system (events triggered by other agents or users). Events that occur in the meantime, during the simulation of these long processes, would create a "queue of events" and the program would respond to them later (after the simulation of the long processes is finished) in the serial manner. This makes simulation of long manufacturing processes (maintaining their states) unnecessarily

complicated, in addition to creating an effect of "unsmooth" simulation, which is unacceptable in this kind of applications. The most obvious example of this is visible when two long processes are running concurrently. Each process has a progress bar that indicates the stage of the each process. Suppose that the process one is started and after that process 2. When the first process starts the current version of the program will start moving small bars inside a progress bar of that process. At the time when the second process starts, advances of the first progress bar will stop, and the small bars inside the second progress bar will start moving. In the meantime the simulation of the first process will be frozen. When the second process is finished the program control will be return to the first process. However, because the time for its completion may be already past, the indication of the process progress on the progress bar will jump from the stage where it was frozen to the very end straight away. It means that the event that indicates that the first process is finished will be late and is triggered only after the second process is completed instead of at the time when the first process was actually finished.

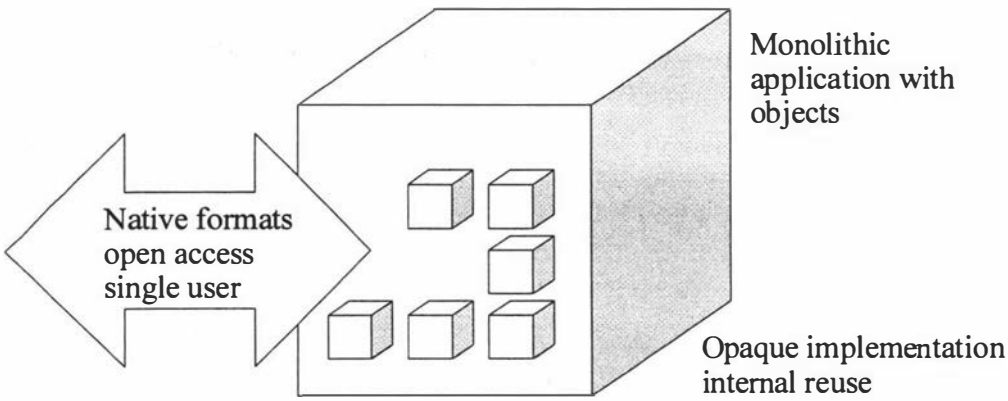


Figure 1-1. A monolithic application with objects [Blexrud et al, 2000]

1.3 A proposed design of the workstation agent - an "n-tiered" application

To increase the responsiveness and scalability[1] of the workstation agents, they need to be designed and developed in a manner similar to the design and development of a distributed network application. This design is also known as "3-tier", "n-tier", or "multi-tier" design[2].

In the n-tier model architecture (see Figure 1-2) the overall application design is divided into several parts (sub-applications). The Client applications remain focused on presenting information and receiving input from users (***presentation services tier*** or **presentation tier),** data are hosted on one or more data servers in the ***data services tier*** or **data tier,** while program logic (business rules) is moved to the ***application logic tier***, also referred as a ***business services tier*** or **middle tier** [Blexrud et al, 2000]. In the data tier some processing is still implemented but it is

[1] Generally speaking **scalability** is an application's ability to serve a growing number of users without experiencing a drop-off in response times and overall system throughput. A process is not considered scalable if its responsiveness and overall throughput decrease in a linear fashion as the number of requests per second increases.

[2] Before the appearance of n-tiered applications, ***the client-server model*** was prevalent model in business applications. Data has been centralised and stored on a relational database - server, and is accessed by remote users - clients - over the network. ***The N-tier architecture model*** resulted from the need to separate the business rules out from the relational databases and to put them in a separate layer. Originally, the business rules were implemented to enforce data integrity in the relational databases (using triggers and stored procedures), however, over time these business rules begun to look like small programs on their own right. This was particularly evident in the large business applications.

restricted only to the processing that is required to access data and maintain data integrity (e.g. query engines, transaction managers, as well as event triggers and stored procedures[1]).

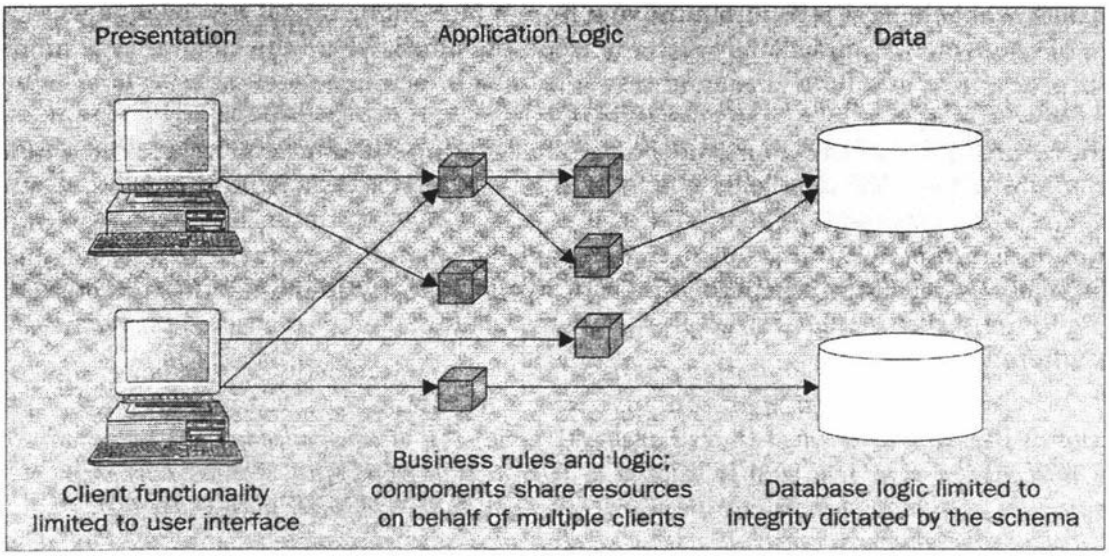


Figure 1-2. Multi tier architecture [Blexrud et al, 2000]

In multi-tier development the number of tiers does not necessarily indicate how many computers are involved. An N-tier model is just logical model. Figure 1-3 shows two possible scenarios. In a small deployment, the business code and the database server might run on the same computer. In a larger system, the data can be kept on one or more dedicated computers while the business objects run on a separate host.

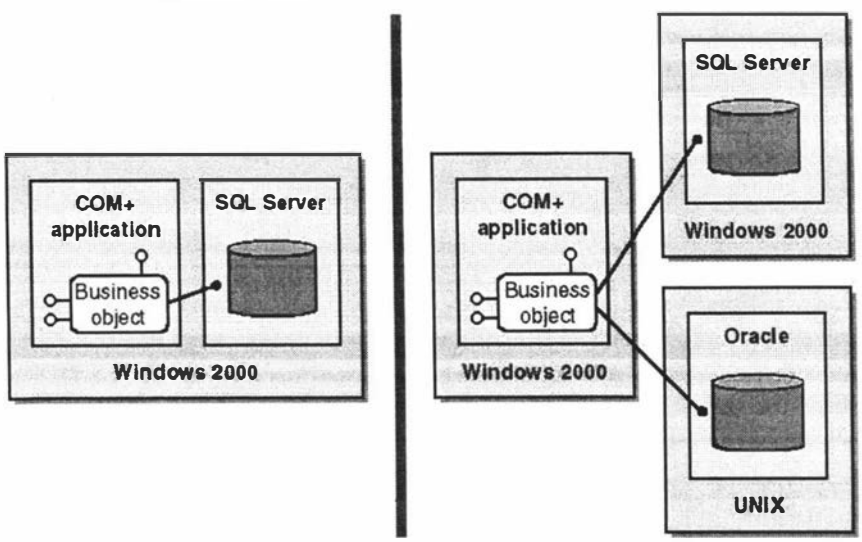


Figure 1-3. The mapping between business objects and the data access layer [Pattison, 2000]

In the new proposed design (refer to Figure 1-4) of the workstation agent architecture, elements and code of the user interface (presentation tier) are separated from the other parts of workstation agents.

[1] Unlike client-server model the triggers and stored procedures are limited in scope to managing the integrity of the data residing on this data tier.

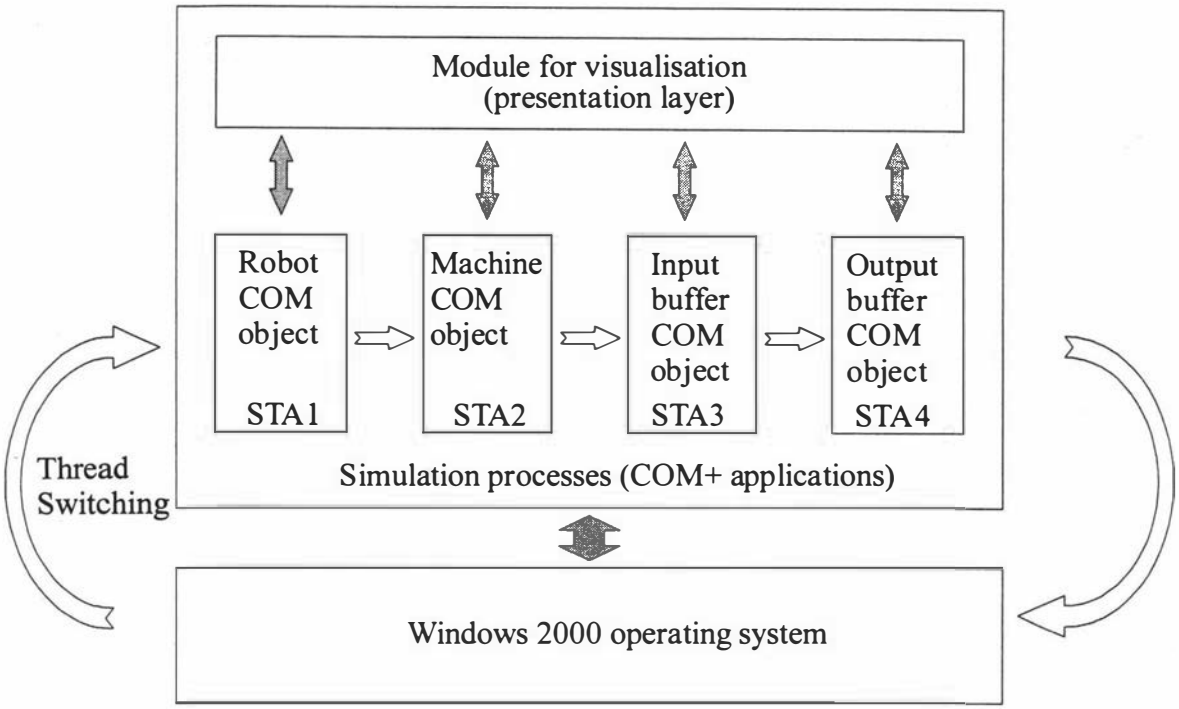


Figure 1-4. Redesign of the workstation agent: Presentation and logic layers are separated while the logic for simulating machine processing tasks and robot handling tasks is encapsulated inside the COM components

Most importantly, each module, or a sub-module of crucial importance for a workstation agent functionality (in particular, the machine and robot logic), should be configured as a separate application - COM component. The COM component is essentially a small application on its own without visual representation. The term COM stands for the Microsoft technology called Component Object Model but it is often used to refer to the piece of software – component - that is developed according to the specification defined by Component Object Model.

By introducing COM components a workstation agent would be actually consisted of a set of a few processes running on one computer. Each component would have its own execution thread making the agents capable of executing parallel processes (simulation activities) in a simultaneous and asynchronous[1] manner. Also, this design would make agents scalable. They would be able to handle each new request, (for example adding a new component in the workstation) with an acceptable level of responsiveness.

1.4 Threading - technical background

Threading is an important issue for applications that experience a moderate to heavy computation load. When a task takes too long to be processed in one sequence, asynchronous processing is used. In that case, the task is broken apart and the part which takes a long time to process is moved into another thread or even another process space (see below). There are two mechanisms for partitioning tasks: queuing and event processing. Event processing is used within the Windows environment while queue processing is more applicable under the web

[1] In this context *asynchronous* has a meaning of non-chronological or non-linear sequence of events. *Asynchronous processing* is an application design pattern that allows non-linear execution of code, whereas synchronous processing is linear-based, and can only complete a single task at a time. In a Windows-based system tasks within a process are carried out by one or more threads.

paradigm. Further discussion on events is out of the scope of this appendix. However, this matter is comprehensively covered in [Blexrud et al., 2000].

The Win32 programming model is based on two high-level abstractions called processes and threads. These terms are all low-level abstractions and are not physical elements.

- A ***process*** is a combination of data and code. It is a running instance of an application that owns an address space of virtual memory as well as various other resources.
- A ***thread*** is the schedulable entity to which the operating system allocates CPU time, and is responsible for executing the application code (instructions). As an entity, a thread is owned by one and only one process. The operating system recognises each thread in every process and is responsible for scheduling threads for time in the system's processors, as shown in Figure 1-5. Every Win32 process owns one or more threads. Each thread has an associated call stack and can store data in a private area known as thread-local storage (TLS)). Unless asynchronous processing is utilised the thread that is assigned to a process will be busy either processing code or waiting for other processes (which are called from the code) to finish. All threads within a given process share the same virtual address space, global variables, and operating-system resources.

Every thread has its own call stack. A scheduler allocates processing cycles by giving each thread a time slice. When the time slice is over, the running thread is pre-empted and another thread is given a turn. A pre-empted thread can keep enough information on its call stack to remember what it was doing and how far it got before being switched out of the processor. When it gets another time slice, it can pick up where it left off. The combination of threads and the scheduler is powerful because it allows a single-processor computer to appear to be doing more than one thing at a time.

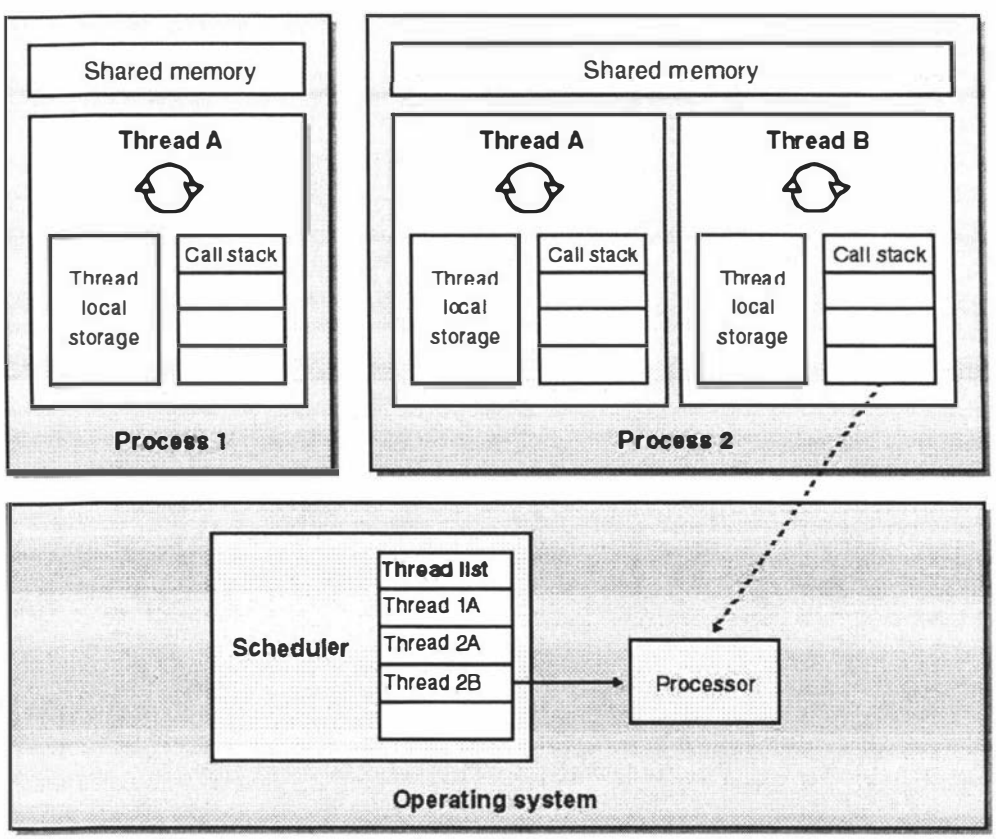


Figure 1-5. Win32 processes and threads

Some caution should be exercised when dealing with multithreaded applications. In particular, ***shared memory*** that is accessed by multiple threads is vulnerable to inconsistency and corruption because of a phenomenon known as ***concurrency***. In a pre-emptive multithreading environment such as Windows 2000, the scheduler switches threads out of the processor arbitrarily. There is no way to guarantee that a thread has completed its work. When another thread is switched into the processor, it might see shared memory left in an invalid state by some other thread that was preempted in the middle of a series of changes.

The problems associated with concurrency arise only when two or more threads access the same data items in shared memory. These problems can be avoided by using ***local variables on the call stack*** instead of using shared memory. These local variables are private to a particular thread. The Win32 API also allows a thread to store persistent data in a private memory area known as ***thread-local storage (TLS)***. Using TLS solves many concurrency problems because there is no need for synchronisation. However, it creates a few problems of its own. Data stored in TLS can only be accessed by the thread that owns it. Objects that store data in the TLS generate a dependency on the thread that created them. Visual Basic objects are heavy users of TLS, so every Visual Basic object has a dependency on the thread that created it. ***A Visual Basic object can never be accessed by any other thread.*** This condition is known as *thread affinity*.

Each process begins its life with a system-created thread – a ***primary thread*** which serves as the entry point into the application. In a typical Windows-based application with a user interface, the primary thread is used to create the application's main window and to set up a message pump to monitor incoming messages sent by the operating system. Once an application is up and running, the Win32 API permits the spawning additional threads using a Win32 API function named *CreateThread*. In most cases, the secondary thread does not monitor a message queue and therefore does not require as much overhead. This type of thread is often referred to as a ***worker thread***. The obvious benefit of the second thread is that a background task can be run without blocking the responsiveness of the user interface.

1.5 Threads and scalability issues

A process that is based on a single thread does not allow for concurrency. On the other hand, a process that spawns a new thread for each client application has problems with respect to scalability. Namely, the operating system must spend a larger percentage of its time switching threads in and out of its processors. As this administrative overhead increases, the process's overall throughput decreases. To attain the highest levels of scalability, some type of thread-pooling scheme must be provided. The goal of such a scheme is to create an optimised balance between higher levels of concurrency and more efficient resource usage. Managing a pool of threads is not a trivial task but fortunately, COM+ provides a built-in scheme for thread pooling, and it does so in a way that is transparent to programmer (COM+ runtime spread out objects across a set of threads at runtime).

While the COM+ threading scheme is largely transparent, several important threading concepts must be taken into consideration when designing Visual Basic components. Coverage of these concepts is given in [Pattison, 2000]

1.6 Threading and apartments

There are two different apartment threaded models that are supported by Visual Basic 6, these are:

- Single threaded
- Apartment threaded

An ***apartment*** is where objects (instances of classes) are created.

1.6.1 A Single-Threaded Apartment (STA) model

The single-threaded apartment has only one primary thread that takes care of all the needs of its processes (refer to Figure 1-6). It is the only thread that has access to the apartment.

Since there is only one thread, calls made to the object all execute in "single-file". This is not a particularly useful scenario when this model is considered with a large number of clients.

Using the single threaded model will make an object serialised, which means that all calls on the object will be dealt with one at a time, no matter which instance of the object called it.

When an object is associated with a single-threaded apartment, only that particular thread may execute calls on the object. When another thread wishes to execute a call on the single-threaded object, the call is marshalled to the apartment that contains the object and that thread executes the call. If a result is required, it is then marshalled back to the original caller.

Marshalling in COM+ requires that two additional objects be created, the ***proxy*** and the ***stub***. These objects are responsible for getting data back and forth between apartments. This affects performance as two objects lie between the caller and the server code. Figure 1-7 shows the proxy/stub pair and how it fits between apartments for marshalling.

Marshalling does not only occur across apartment boundaries, but also across process boundaries and machine boundaries.

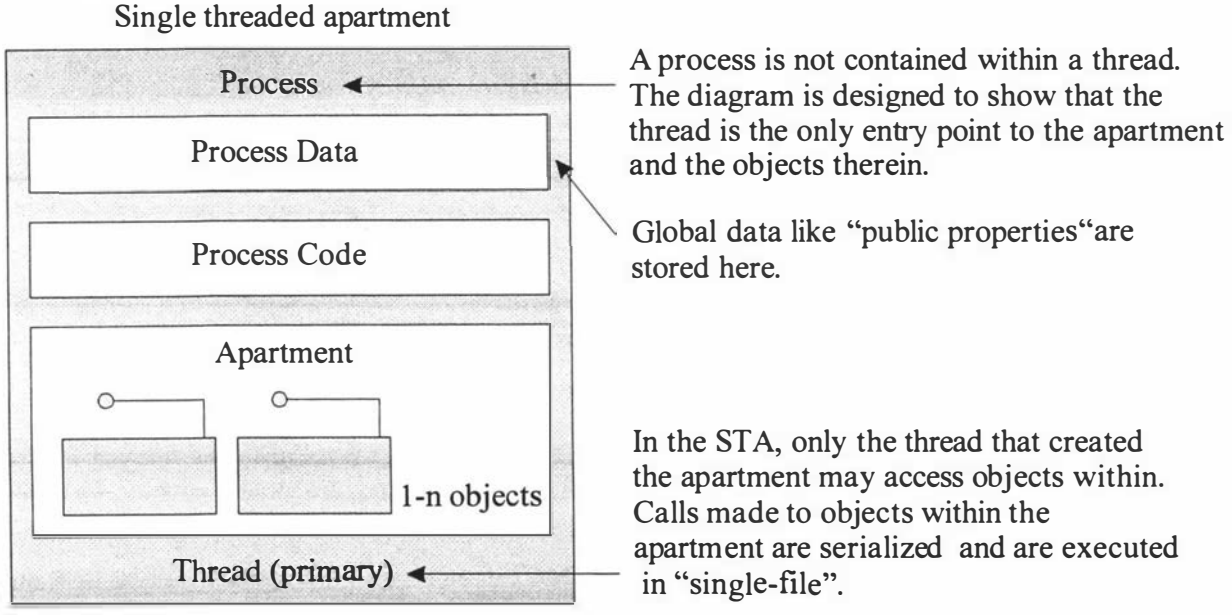


Figure 1-6. Single threaded apartment model

1.6.2 A multiple-threaded apartment model

In the apartment-threaded option (see Figure 1-8) an apartment can have many threads of executable code, which can all execute simultaneously. This makes it a better option for a server component, as clients do not have to wait in a queue for calls to complete as is the case in the single threaded option.

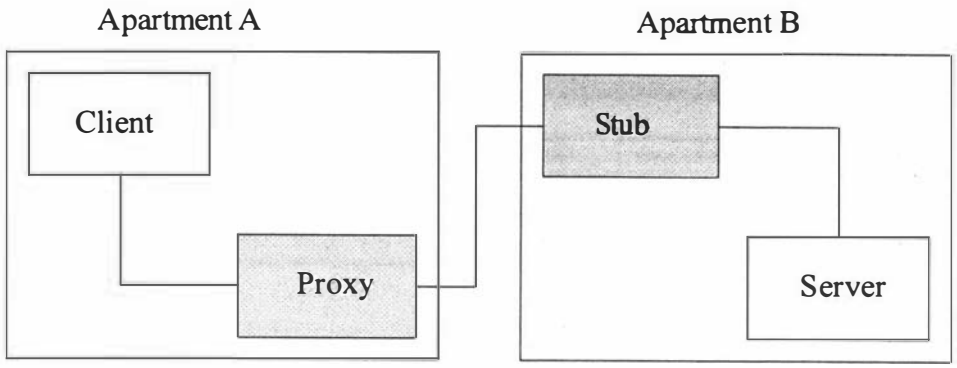


Figure 1-7. Marshalling calls across apartment boundaries by using the proxy/stub combination

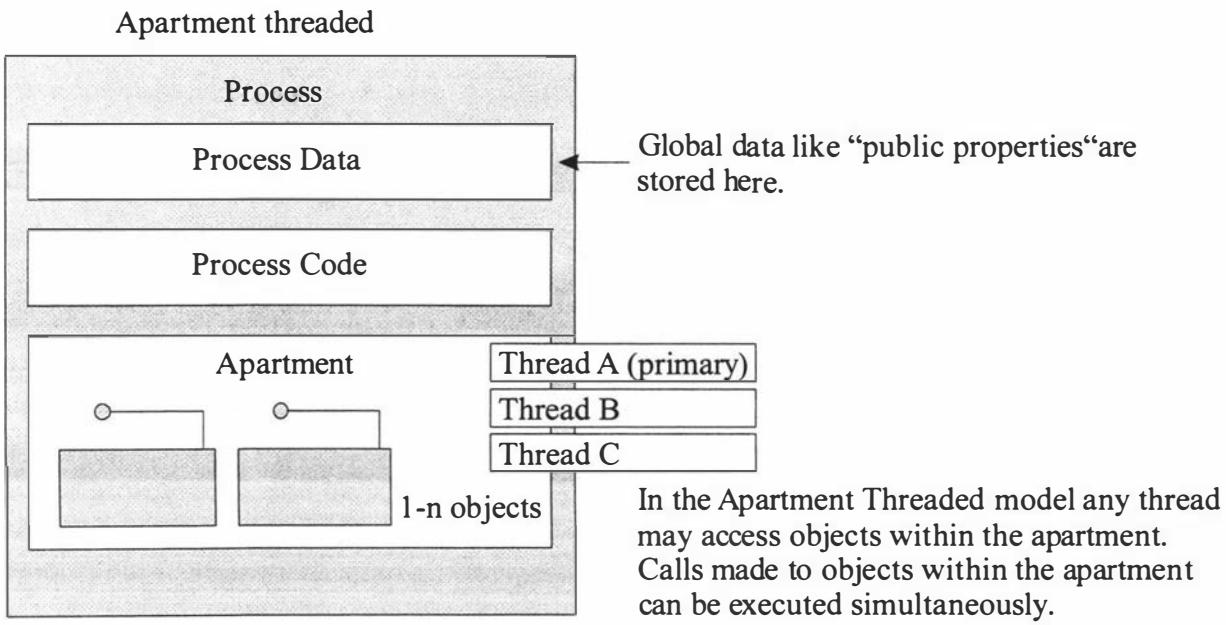


Figure 1-8. Apartment treaded model

1.7 The proposed platform for the future developments

COM+ and Microsoft Windows 2000 provide a robust development platform. This platform is made up of several core technologies that provide the basic building blocks for constructing multi-tier business applications, and since the underlying platform provides more features, the the user has to write and debug less code.

VB and C++ programming languages remain valuable tools for building presentation elements. They are the best performing solutions in terms of speed. Both languages are tightly tied to the Windows platform.

1.8 Reusability of code by applying COM components

Until recently, if we had, for example, the code for a Customer object in one program, and we wanted to use that object in another application, we had to COPY the code into our application. In this way we could reuse the code, but, if anything was changed in the original code, we would need to change each copy of the customer object in all the different applications where it could be in use. The problem with code reuse in this way is to keep all the different copies synchronised.

With binary components, we can achieve a whole new level of reuse in our applications. Going back to our Customer object, we can put the code into one component rather than into each individual application. The applications can then use the object directly from the component: We get the same functionality as if we had copied the source code into each application, but now we only need to maintain that code in a single place - the component.

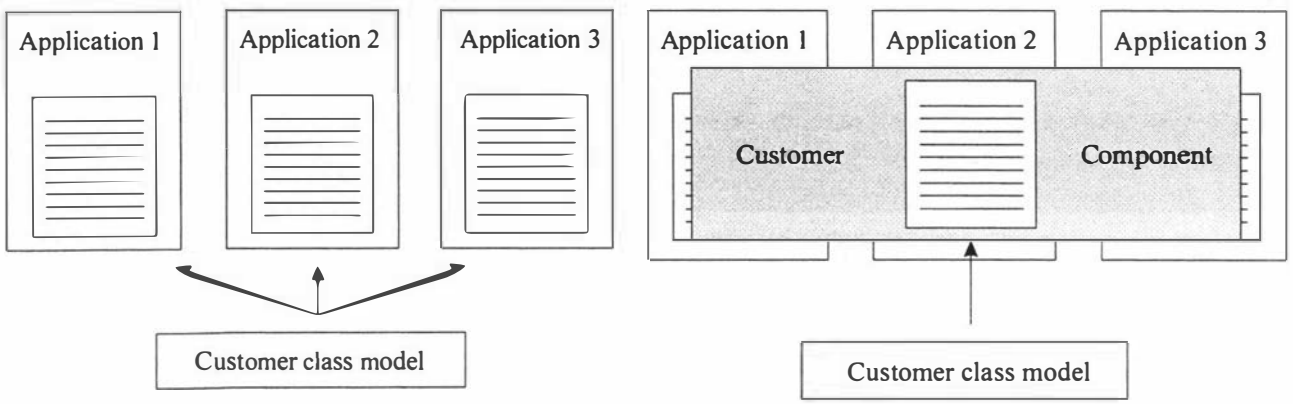


Figure 1-9. Reusability of code

1.9 Summary

The present software for demonstrating the ability to create a heterarchical agent based shop floor control system from commonly available programming systems, suffers from being a monolithic application. Although this worked appropriately in standard real time applications if the system was to be used for faster than real time simulation to enable a wide range of statistics to be collected about its performance, for example, then its structure would need to be modified. It is recommended that a multiple threaded structure be developed (with COM components as described above). This would considerably enhance the reliability and the capability to perform at faster than real time as with a large number of workstations. This should be the subject of future development of the system.

Appendix 2. A copy of the Letter from the Massey University New Technology Developments Operations Group

Leading University Technologies

NEW TECH DEVELOPMENTS

MASSEY UNIVERSITY

Goran Colak
3 D, Crew Crescent
Palmerston North

12 November, 1999

Research Services
Massey University
Private Bag 11222
Palmerston North
New Zealand
Telephone +64-6-350 5341
Facsimile +64-6-350 5690

Dear Goran,

re: DISTRIBUTED SHOP FLOOR CONTROL SYSTEM

After the demonstration of the "distributed shop floor control system" software at your place, the New Technology Developments Operations Group had a discussion in relation to your needs for taking your project into a commercial reality. It became clear that your project is based in a very interesting concept and that the software you developed is a vehicle for the implementation of this novel concept.

Through the discussion it also became apparent that at this point of time, this project might be beyond Massey's scope and it is doubtful whether Massey can add any value.

The New Technology Development Operations Group decided not to go to the next step in further evaluating the technology.

The recommendation from the group was to ensure that your supervisors at Massey are aware of the commercial potential of your research and of the need to protect your intellectual property.

The NewTech group also recommended that any approach by you to third parties should be done under confidentiality agreements. I would be happy to provide you with appropriate templates.

In any case you should write your PhD thesis first with full disclosure of the novel concept/algorithm behind the software development, for the purpose of being fully assessed by the examiners under strict confidentiality.

I know that this might not be the outcome you wished for, but in all fairness to you, we felt that you should keep the entire intellectual property because we do not have the resource to add value to your project and take it to the next phase for a successful commercial exploitation.

Please, do not hesitate to contact me for any assistance you may require in the future, particularly in relation to the provision of templates for your Confidentiality Agreements.

We wish you success in your endeavours.

Best regards

Josephine Serrallach
New Technology Manager

copy: Prof. Dan Barnes