DEVELOPMENT OF A WEB-BASED DYNAMIC SCHEDULING METHODOLOGY FOR A FLEXIBLE MANUFACTURING CELL USING AGENT BASED DISTRIBUTED INTERNET APPLICATIONS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

BORAN ALATAŞ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

THE DEPARTMENT OF MECHANICAL ENGINEERING

JANUARY 2004

Approval of the Graduate School of Natural and Applied Sciences.

_____

Prof. Dr. Canan ÖZGEN
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

_____

Prof. Dr. Kemal İDER
Head of the Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

_____

Prof. Dr. Ömer ANLAĞAN
Supervisor

Examining Committee Members

Prof. Dr. S. Engin KILIÇ (Chairman) _____

Prof. Dr. Ömer ANLAĞAN _____

Prof. Dr. Sahir ARIKAN _____

Prof. Dr. Mustafa İ. GÖKLER _____

Assoc. Prof. Dr. Tayyar ŞEN _____

ABSTRACT

DEVELOPMENT OF A WEB-BASED DYNAMIC SCHEDULING METHODOLOGY FOR A FLEXIBLE MANUFACTURING CELL USING AGENT BASED DISTRIBUTED INTERNET APPLICATIONS

Alataş, Boran

M. Sc., Department of Mechanical Engineering

Supervisor: Prof. Dr. Ömer Anlağan

January 2004, 134 pages

The increasing importance of computer leads to develop new manufacturing methods. One of the most important example; "unmanned shop floor" model aims, the mankind can work in jobs that they can be more efficient and more comfortable. As the base of this model, in Middle East Technical University Computer Integrated Manufacturing Laboratory (METUCIM) "Agent Version 1.1" system is developed. Windows Distributed Internet Applications (DNA) modeling technique is used for the software development. In the developed system, by using web pages, one can give work orders to the flexible manufacturing cell in METUCIM. The manufacturing capabilities of the cell are limited by the capabilities of CNC Lathe and CNC Milling machine that exist in the system.

By the developed agent based dynamic scheduling method, it is prevented to be only an experimental system for the manufacturing cell. The real manufacturing environment is adapted to the cell that it is possible to give

unlimited number of work orders. The work orders can be queued and manufactured according to their “priorities”. By the “web-cam” application the given work orders can be watched from the web site so the system reliability is increased for the engineer. In the real manufacturing environment it is very frequent that the “urgent part” is needed to manufacture. In this system it is possible to give “urgent orders” for these situations.

Keywords: Shop Floor Control, Agent, Distributed Internet Applications, DNA for Manufacturing, Priority, Urgent Order, Manufacturing Execution Systems.

ÖZ

AJAN TEMELLİ ÇOK MERKEZLİ İNTERNET UYGULAMALARI KULLANILARAK ESNEK ÜRETİM HÜCRESİ İÇİN WEB TABANLI DİNAMİK PLANLAMA METODU GELİŞTİRİLMESİ

Alataş, Boran
Yüksek Lisans, Makina Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Ömer Anlağan

Ocak 2004, 134 sayfa

İmalatta bilgisayarın öneminin giderek artması yeni üretim modellerinin geliştirilmesine yol açmıştır. Bunun önemli bir örneğini olan "insansız atölye" tasarımı, insanların daha verimli olabilecekleri alanlarda ve daha konforlu koşullarda çalışabilmesini sağlamayı amaçlamaktadır. Bu tasarımın temeli olarak Windows Çok Merkezli İnernet Uygulaması kullanılmak suretiyle Orta Doğu Teknik Üniversitesi Bilgisayar Tümleşik İmalat Laboratuvarı'nda "Agent Version 1.1" sistemi geliştirilmiştir. Bu sistemde internet web sayfaları üzerinden laboratuvarımızdaki esnek üretim hücresine iş emri verilebilmekte, sistem içerisindeki torna ve frezenin yetilerinin sınırları içerisinde "insansız" üretim yapılabilmektedir.

Yeni geliştirilen ajan temelli dinamik planlama metodu sayesinde mevcut üretim hücresinin sadece deneysel bir sisem olmasının önene geçilmiş, gerçek imalat koşulları sisteme adapte edilmiştir. Bu adaptasyonla web üzerinden sınırsız iş emri verilebilmekte ve bu iş emirleri kendi içerisinde "önem sırası" na göre sıralanıp, üretim yapılabilmektedir. Web kamerası uygulaması ile web

sayfasından, verilen iş emirleri canlı olarak izlenebilmekte, böylece kullanıcı için sistemin güvenirliği artmaktadır. Bu sistemde, gerçek imalat endüstrisinde sık sık karşımıza çıkan "acil parça" gereksinimi, "acil iş emri" uygulaması ile karşılanabilmektedir.

Anahtar kelimeler: Atölye denetimi, Ajan teknolojisi, Dağıtık İnternet Uygulamaları, Dağıtık İmalat Uygulamaları, Önem Sırası Uygulaması, Acil İş Emri, İmalat Sistemleri Uygulamaları.

To My Family

ACKNOWLEDGMENTS

I would like to express my gratefulness and appreciation to my supervisor Prof. Dr. Ömer Anlağan for his guidance throughout the completion of this thesis. Also thanks go to Prof. Dr. S. Engin Kılıç for his encouragement about the study.

I would like to thank Tolga Cangar and Özgür Ünver for the design of the system model, on which this development process is based.

I am also indebted to my colleagues Yusuf Başıbüyük, Burak Sarı, Erhan Özsüer, Fatih Sarı in Integrated Manufacturing Technologies Research Group (IMTRG) for their endless support all through this hard work.

Finally, my greatest thanks go to my parents who shaped me with their never ending patience.

TABLE OF CONTENTS

LIST OF TABLES

TABLE

LIST OF FIGURES

FIGURE

CHAPTER 1

INTRODUCTION

1.1 CIM Concepts

The manufacturing industries have become the most important contributors to prosperity for the industrialized nations. However, it becomes increasingly difficult to meet customers' demands and compete on the international market. Thus, manufacturing industries must be able to react quickly to prevailing market conditions and to maximize the utilization of resources.

The industry is going through a period of rapid change, accompanied by record growth. To meet the challenge of selling products into a competitive global economy, while continuously reducing costs, manufacturing companies have to increase the efficiency of existing plants. Integration is the key to the success of deploying a modern Computer Integrated Manufacturing (CIM) system, but wiring the components together to produce such a system requires skills in full system model design and Information Technology [1].

The manufacturing systems of the future have to be flexible, and for this reason they must be reprogrammable. But any increase in flexibility will entail higher installation costs. Thus, it is necessary to provide a streamlined and uninterrupted production process, which is highly efficient and reliable. Information processing plays a major role in obtaining these goals. Information is considered an important resource whose true value is often difficult and impossible to estimate.

CIM conveys the concept of a semi- or totally automated factory in which all processes leading to the manufacture of a product are integrated and controlled by computer. It includes Computer-Aided Design (CAD), Computer-Aided Process planning (CAP), Production Planning and Control (PP&C), Computer-Aided Quality control (CAQ) and Computer-Aided Manufacturing (CAM). A summary of concepts through the integration of CIM is given in Figure 1.1

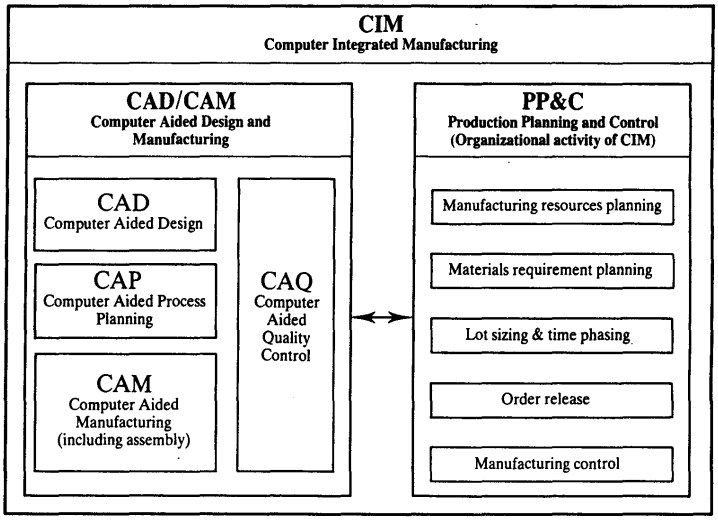

Figure 1.1 CIM Integration

Thus, the integrated definition of CIM includes both design and manufacturing data to be processed uniquely to obtain the optimum solution:

- Design human user interfaces which make complex reconfiguration more manageable,
- Write code in an object oriented language that is modular, re-configurable and fast,
- Use standard and well accepted communication protocols,
- Distribute the processes to as many workstations as possible.

Throughout the last few years, the design of distributed systems of autonomous agents, so called multi-agent systems (MASs) for use in

manufacturing gained attention in the robotics and automation research community. Due to their distributed nature, MASs promise, at least theoretically, some advantages that make them attractive structures for control and execution of manufacturing processes. Agents are modular system elements having robustness and fault tolerance and are easily maintainable and extendible. These features of MASs hold the potential of building manufacturing systems with greater flexibility then the currently used monolithic ones [2].

The concepts of CIM include a broad range of definitions from manufacturing to control and computer technology. It cannot be interpreted without a basic knowledge of software terms and philosophies. Section 1.2 discusses the technology of CIM from the programmer's perspective.

1.2 Software Technology

Constant innovation in computing hardware and software have made a multitude of powerful and sophisticated applications available to users at their desktops and across their networks. Yet, with such sophistication have come many problems for developers, software vendors, and users. For one, such large and complex software is difficult and time-consuming to develop, maintain, and revise. Revision is a major problem for monolithic applications, even operating systems, in which features are so intertwined that they cannot be individually and independently updated or replaced. Furthermore, software is not easily integrated when written using different programming languages and when running in separate processes or on separate machines.

Object-oriented programming has long been advanced as a solution to the problems at hand. However, while object-oriented programming is powerful, it has yet to reach its full potential because, in part, no standard framework exists through which software created by different vendors can interact within the same address space and across network and machine architecture boundaries.

The Microsoft's solution to the object-oriented paradigm is represented by Object Linking and Embedding (OLE) components. OLE offers a solution and a future extensible standards and mechanisms to enable software developers to package their functionality, and content, into reusable components, like an integrated circuit. Instead of worrying about how to build functions, developers can simply acquire or purchase that function without having to care about its internal implementation

Client/server computing is moving into the mainstream of corporate information systems. With this move comes the need for client/server applications that can access enterprise-wide data. Much of this data is stored in databases, which are accessible for the clients, which will then process the information and perform individually. The common interface architecture is Open Database Connectivity (ODBC), a gateway to manage the communication with multiple back-end databases. Single or multiple servers act as file or information storage, and interface suppliers responding to client drivers on client's Application Program Interfaces (API's) [3]. Figure 1.2 depicts a typical client server relationship between client drivers and server interfaces.

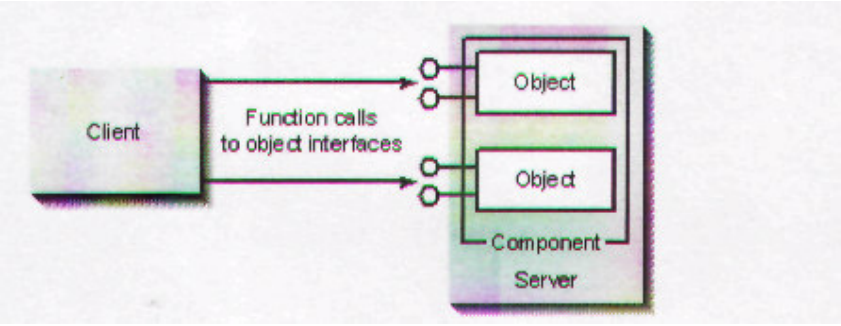

Figure 1.2 Client Server Relationship [3]

DNA architecture maps out the framework for building scaleable, three-tier distributed applications that can run over any network, including the web. The

term “three-tiered” defines the computers on which the application/service is running, which are:

- **Client tier**: a local computer on which either a Web browser displays a Web page that can display and manipulate data from a remote data source, or (in non–Web-based applications) a stand-alone compiled front-end application.
- **Middle tier**: a server computer that hosts components that encapsulate an organization's business rules. Middle-tier components can be either Active Server Page scripts executed on Internet Information Server, or (in non–Web-based applications) compiled executables.
- **Data source tier**: a computer hosting a database management system (DBMS), such as a SQL Server database. (In a two-tier application, the middle tier and data source tier are combined.)

Many technologies under the DNA umbrella can help developers to create applications for all three tiers: navigation and user interface, business processes, and data storage. The user interface might be a DNA client application that runs in a Web browser. The business process might be a DNA application that runs on a Web server. Data storage could be handled by a DNA application running on almost any type of computer, from PC server to mainframe.

1.3 Scope

The first scope of this study is to develop a scheduling methodology which includes “priority” in job dispatching with an “urgent order” application, and realize it in the test-bed of Middle East Technical University Mechanical Engineering Department Computer Integrated Manufacturing Laboratory (METUCIM). The second scope is to develop a new control method to make the production in batches for ordered parts. The shop-floor is to be modeled as a distributed multi-agent system with typical agents of machines such as CNC, Robot, Pneumatic Linear Robot Drive (PLRD), Buffer, and AGV; and parts with

all the operational information located on the main database. The whole system is controlled by the end user (client-tier), which has the necessary access password to the web site. While Internet Information Server (IIS) and Microsoft Transaction Server (MTS) are running on the main server computer (business-tier), SQL server is working on the back up computer (data source-tier).

The developed methodology is based on an object-oriented modeling approach where particular agents and their inheritances are assigned to classes. The developed software is generally written in Visual Basic 6.0, the web site is designed with Visual InterDev 6.0, and the database is constructed on SQL Server 7.0. This multi language, n-tier structure also reveals the advantages of DNA and Component Object Model (COM) technology.

1.4 Outline

Chapter 2 consists of a survey related to flexible manufacturing systems, their control, scheduling methods, modeling techniques and related software technology. Chapter 3 describes the system and data model of the developed method "Agent Version 1.1". Chapter 4 gives the description about development of three-tiered system model. Chapter 5 describes how the "Agent v1.1" system works by the test runs. Chapter 6 includes the concluding remarks and possible future work plans. Appendices A, B, C, D and E gives the detailed information on User's Manual, G-Codes of test runs, Key and Ids and sample code respectively.

CHAPTER 2

LITERATURE SURVEY

2.1Flexible Manufacturing System (FMS):

An FMS can be defined as a computer-controlled configuration of semi-dependent workstations and material-handling systems designed to efficiently manufacture various part types with low to medium volume. It combines high levels of flexibility with high productivity and low level of work-in-process inventory. The need for flexibility, efficiency, and quality has imposed a major change in manufacturing industries. An FMS can be considered flexible if it is able to process parts as and when they arrive into the system. Use of flexible manufacturing systems lead to:

- Increased product variety to satisfy customer needs.
- Shorter product development cycle.
- Flexibility to adapt to changes in the market.
- Improved capital/equipment utilization.
- Increased productivity and decreased costs of goods and services to maintain the market share.
- Reduced set up time and work-in-process (WIP).
- Quick cell creation for a new product family by simply re-programming the FMS.

One of the objectives of an FMS is to achieve the flexibility of low volume production while retaining the efficiency of high-volume mass production. To achieve this efficiency, various decisions must be made. Some of these decisions

are the selection of manufacturing control type, selection of scheduling type and choosing the right software tools. [4]

2.1.1 Requirements for Next Generation Manufacturing Systems

The manufacturing enterprises of the 21st century will be in an environment where markets are frequently shifting, new technologies are continuously emerging, and competitors are multiplying globally. Manufacturing strategies should therefore 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. Such manufacturing systems will need to satisfy the following fundamental requirements:

- ***Enterprise Integration:*** In order to support global competitiveness and rapid market responsiveness, an individual or collective manufacturing enterprise will have to be integrated with its related management systems (e.g., purchasing, orders, design, production, planning & scheduling, control, transport, resources, personnel, materials, quality, etc.) and its partners via networks.
- ***Distributed Organization:*** For effective enterprise integration across distributed organizations, 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 representation 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 reinitializing the working environment. This will require an open and dynamic system architecture.
- ***Cooperation:*** Manufacturing enterprises will have to fully cooperate with their suppliers, partners, and customers for material supply, parts fabrication, final product commercialization, and so on. Such cooperation should be in an efficient and quick-response 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, 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.
- ***Scalability:*** Scalability means that additional resources can be incorporated into the organization as required. This capability should be available at any working node in the system and at any level within the

nodes. Expansion of resources should be possible without disrupting organizational links previously established.

- ***Fault Tolerance:*** 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 minimize their impacts on the working environment.

Global competition and rapidly changing customer requirements are forcing major changes in the production styles and configuration of manufacturing organizations. Increasingly, traditional centralized and sequential manufacturing planning, scheduling, and control mechanisms are being found insufficiently flexible to respond to changing production styles and highly dynamic variations in product requirements. The traditional approaches limit the expandability and reconfiguration capabilities of the manufacturing systems. The traditional centralized hierarchical organization may also result in much of the system being shut down by a single point of failure, as well as plan fragility and increased response overheads. Agent technology provides a natural way to overcome such problems, and to design and implement distributed intelligent manufacturing environments. [5]

2.1.2 Manufacturing Cell Control

There are three principal types of shop-floor control architecture; the centralized, hierarchical and heterarchical. Okuba et al. (2000) compared the performance of a distributed control system against that of a centralized scheme using software modules (analogous to autonomous agents) to represent each job, transporter, and work cell. Unlike the system in this study, the communication between modules (agents) was not specific between entities of the system, but instead messages were broadcast to all entities. The decision as to what resource to use in processing was made using the completion time estimates provided by each work cell. The findings showed that the performance of the distributed system in terms of system lead times was better than a centralized system.

2.1.2.1 Traditional (centralized) approach

The traditional Manufacturing Cell Controller, developed and implemented for the Flexible Manufacturing Cell, uses a modified hierarchical architecture approach [6].

The Cell Controller architecture is a set of several modules, whose "brain" is the Manager Module, which is responsible for the control and the supervision of the production process of the manufacturing cell and also for the management of cell resources. Each physical device has an module, designated by Device Controller, which is customized to the industrial machine, such as production or handling equipment, and it has the responsibility for the local control of the machine, and for the execution of the jobs requested by the high level module.

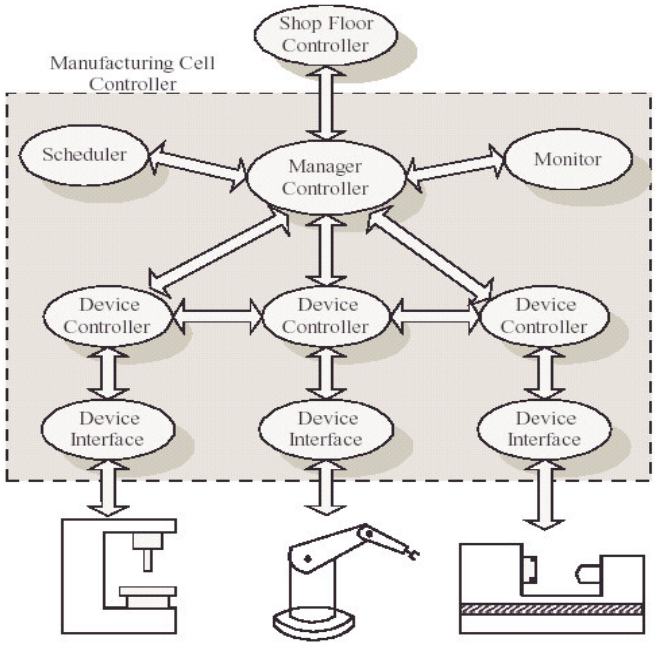

Figure 2.1 Manufacturing Cell Controlling using a Traditional Approach [7]

The interface between the Cell Controller and each of the industrial machines is implemented using the MMS (Manufacturing Message Specification) communication protocol. MMS define a standardized message system for exchanging real-time data and supervisory control information between networked devices and/or computer applications in such a manner that it is independent from the application function to be performed and from the developer of the device or application. [7]

The traditional approach presents the following main problems:

- **Reconfiguration**: It fits very well for applications that present a rigid organizational structure. However, it falls down when it is necessary to change. (for example, new shop floor layout, new strategies for the hierarchy, etc.).
- **Learning and disturbance management**: It is hard and complex to introduce intelligence in the application, in order to optimize its execution and to manage the disturbances and warnings.
- **Distribution and decentralization**: Doesn't support efficiently the distribution and decentralization of functions and entities.
- **Code re-usability**: The development of this type of applications based on this traditional approach has the advantage of its simplicity, when compared with other advanced approaches, but the code developed cannot be re-used.

2.1.2.2 The agent based approach

The multi-agent systems are defined as sets of agents, which represent the objects of the systems and through cooperation mechanisms perform complex tasks [8, 9]. In the automation and manufacturing domain, an agent is a software object that represents automation and manufacturing system objects, such as tasks, CNC machines, robots, AGVs, buffers, PLC devices and sensors.

The multi-agent technology is suitable for the distributed manufacturing environment. The automation and manufacturing applications characteristics like modular, decentralized, changeable, ill-structured and complex, are best suited for agents to solve [10].

Analyzing the benefits of multi-agent technology it is possible to conclude that they overcome problems presented by traditional approaches:

- **Autonomy** : An agent can operate without the direct intervention of external entities, and has some kind of control over their behavior
- **Cooperation:** The agents interact with other agents, in order to achieve a common goal.
- **Reactivity:** The agents perceive their environment and response quickly to changes that occur on it.
- **Proactivity:** The agents do not simply act in response to their environment, but are able to taking the initiative, controlling its behavior.
- **Adaptation and Decentralization**: The agents can be organized in a decentralized structure, and easily be reorganized into different organizational structures.

Using agent-based cell controllers have some more advantages. These are:

- **Platform independency**: The use of Object Oriented programming language and distributed communications platforms, such as CORBA, to develop control applications, allows the use of the same application in different operating systems environments (such as Windows, Linux and Unix), being platform independent.
- **Application development**: Using the agent-based approach, the software necessary to develop the application is shorter and simpler to write, to debug and to maintain.

- **Code re-usability**: The multi-agent technology concept allows an easy and modular development of control applications. Additionally, some components of the developed control application can be re-used for other applications.
- **Distribution and Autonomy**: Each agent has autonomy, has control about its behavior and has local and community knowledge. By this way, it is possible to build distinct and independent agents that can be placed transparently in a distributed environment.
- **Plugging Intelligence**: The addition of intelligence to an agent, for example to take decisions, manage disturbances or learning, is a transparent process for the agent and can be viewed as a plug-in of an intelligence module, which takes easier the development of control applications.

2.1.3 Agent Based Task Allocation and Dispatching

The allocation of operations that belongs to a task, the dispatching and their execution, is a crucial aspect in the control application. As a scenario the operations are announced individually, being the task agent responsible for the analysis and allocation of the operations. An important issue to be considered is the precedence between operations, which affects mainly the start date for each operation.

2.1.3.1 Hierarchical Task Allocation and Dispatching Structure

In the hierarchical structure, there is a supervisor agent, which takes the name of cell controller, and which is responsible for the coordination of the operational agents that represent the cell resources. Those operational agents are not visible from the exterior, and the task agents can only interact with the supervisor agents.

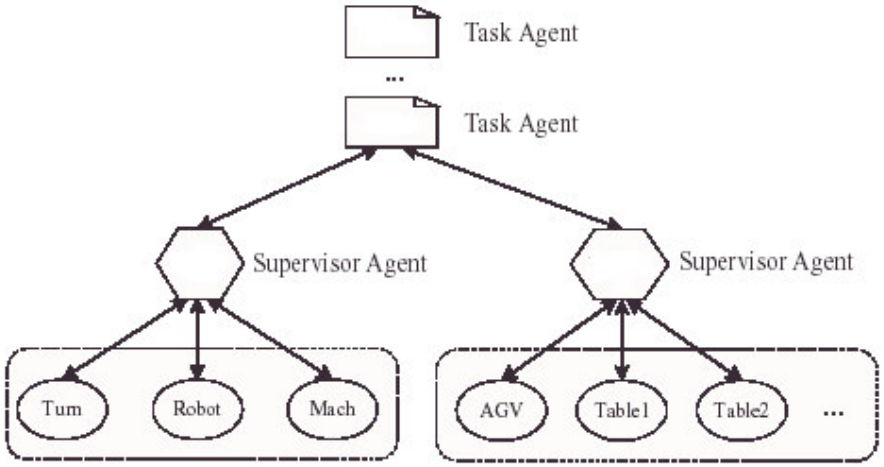

Figure 2.2 Hierarchical Task Allocation and Dispatching Structure [7]

In this structure, the task agent decomposes the task in operations and announces them to the supervisor agents available in the system. Each supervisor agent (cell controller) verifies the availability to execute the operation and elaborates a proposal to the task agent. The supervisor agent can ask for additional information to the operational agents that coordinate, in case of don't have enough information to elaborate a proposal.

After the expiration time, the task agent takes a decision and allocates the operation to the supervisor agent that had presented the best proposal. The supervisor agent should manage the execution of the operation, through the dispatch of the operation to the operational agent that represents the resource that will execute the operation. When the operation is finished, the supervisor agent should notify the task agent.

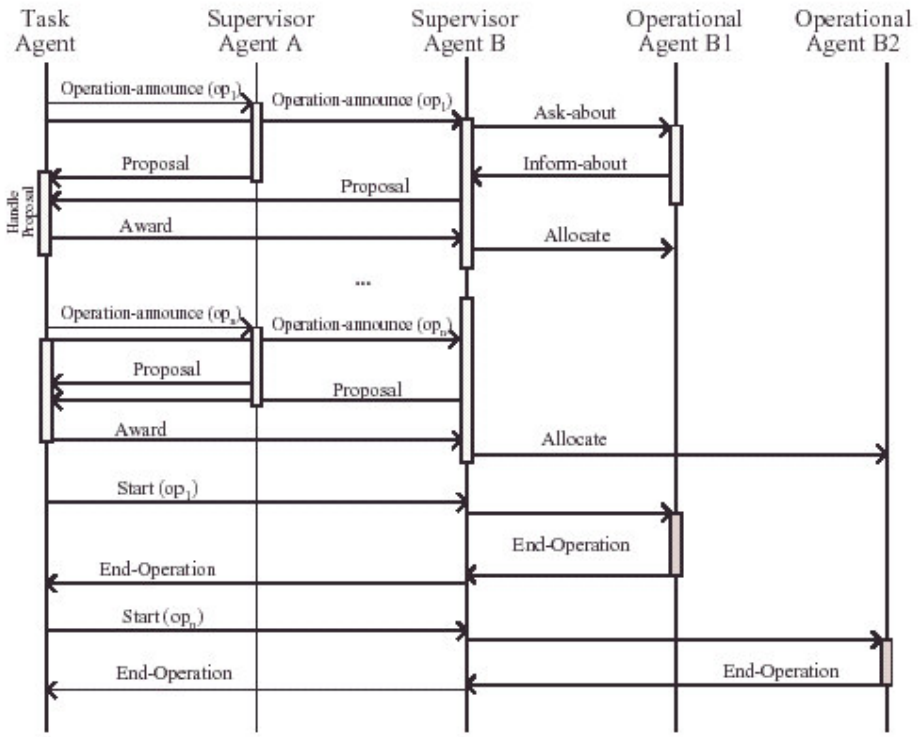

Figure 2.3 Communication Between Agents in the Hierarchical Structure[7]

2.1.3.2 Heterarchical Task Allocation and Dispatching Structure

In the heterarchical structure, there isn't a supervisor agent that represents the cell controller, being the cell controller replaced by the several operational agents that represents the cell resources.

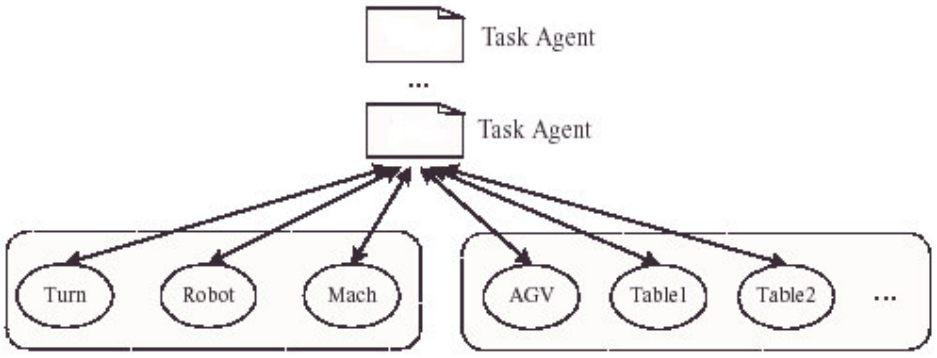

Figure 2.4 Heterarchical Organizational Structure [7]

Initially, the task agent announces the first operation to all operational agents available in the system. After the compilation of all proposals, the task agent evaluates and allocates the operation to the best proposal.

The next step is the announcement of the second operation, using the same procedure and indicating a start date based in the end date of the previous operation. This procedure is repeated until all operations are allocated.

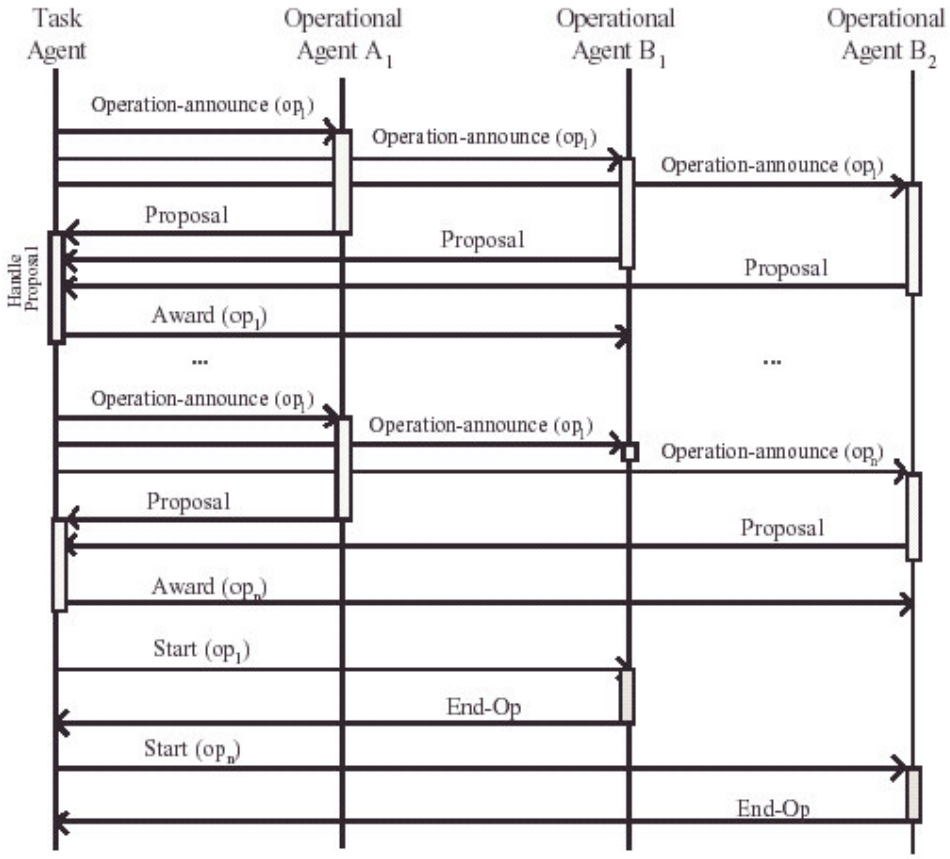

Figure 2.5 Communication Between Agents in the Heterachical Structure [7]

The operational agents can start the execution of the operation after the reception of the *Inform-about* message, indicating the availability to start the operation and the position of the material. The task agent gives this indication after the allocation and execution of the handling operations.

The application of multi-agent systems based on the concept of distributed artificial intelligence is believed to be one of the most promising control architectures for next-generation manufacturing [11]. Such systems are composed of distributed heterogeneous agents and make use of flexible control mechanisms for creating and coordinating the resulting society of agents. This society of agents provides the foundation for the creation of an architecture that possesses the

capability to benefit manufacturing by enhancing a system's reliability, maintainability, flexibility, fault recovery, and stability, as well as providing a means for real-time planning and scheduling. [12]

It is viewed control systems as "arrangements of decision-making and decision-execution entities" whose job is to accomplish the goals of the system. [13]. This view fits well with the idea of employing intelligent agents in these roles. In such roles an agent can be defined as "A software program that can perform specific tasks for a user and possesses a degree of intelligence that permits it to perform parts of its tasks autonomously and to interact with its environment in a useful manner" [14]. This definition implies that an agent must be capable of interacting with its environment in a flexible goal-directed way. This interaction would involve gathering information on its environment (machines, orders, etc. and including other agents), recognizing important states of the environment, making decisions based on this information, and then affecting the environment by executing specific actions as a result of the decisions it has made[15].

Manufacturing control involves the coordination of the flow of both physical items, as well as information. Therefore, the agents within agent-based systems applied to this area are found to represent either the physical or informational entities that are required by the system [16]. The agents of the system are said to behave intelligently as noted by their ability to process and react to information they receive from their environment in the form of communication from other system agents or possibly directly from sensory input. An agent operates by using its knowledge about the world in concert with information received from external sources to reason about what actions to take in order to satisfy local and global objectives.

Hatvany (1985) [17] was one of the first to propose a heterarchical control system. Duffie and Piper (1987) [18] then made use of agents to represent the parts and workstations in such a production system with the part agent negotiating

with workstations for its processing needs. Shaw (1988) [19] implemented the contract net protocol for negotiation to support scheduling in his cellular manufacturing systems. In his case, the best bidder was selected based on the earliest finishing time. Lin and Solberg (1992) [16] made use of a market-like model that combined objective and price mechanisms. The job orders are given currency based on criteria such as priority and then use this currency as the basis of negotiation with resources who are each responsible for determining their own pricing based on such factors as utilization and queue size. Saad *et al.* (1997) [20] also made use of a contract-net approach for heterarchical scheduling of a flexible manufacturing system. Their system employed what they refer to as a production reservation (PR) approach where the job agent schedules all the operations prior to its release to the shop. A problem with the PR approach is that it doesn't handle the need to reschedule jobs when machine breakdowns occur or there is a need to modify an order. Saad *et al.* (1997) also proposed a single step production reservation (SSPR) approach that schedules one operation at a time as the job moves through the system. In terms of average tardiness, they found that SSPR outperformed PR.

Sousa and Ramos (1999) [21] proposed a negotiation protocol for scheduling that is based on the contract-net but extends the concept to permit the system to handle temporal constraints. However, it appears that the system uses a PR approach sacrificing some of the advantage offered by the use of the SSPR method. The bids submitted by the resources offer the time windows during which they are free. Bid selection is then based on finding a resource that is able to finish the part before the due date and has more free time. Recently, Lui and Yih (2001) [11] explored applying an agent-based scheme to heterarchical control for a make-to-order production system where each of five products make use of a dedicated line with a predefined process plan offering no alternatives. They made use of collaborative agents to make possible the determination of the release timing of jobs to the system and the priority ordering of the jobs in machine cell buffers.

2.2 Scheduling

Scheduling of job shops and FMSs has received immense attention over the last three decades and there is an extensive body of literature on job shop and FMS scheduling research. In a production system, the scheduling problem is to synchronize resources, which are connected by a material transport system such as automatic guided vehicle (AGV), and parts in order to produce a variety of products in a predetermined period of time. Scheduling rules are used to select the next part to be processed from a set of parts awaiting service. These rules can also be used to introduce work pieces into the system, to route parts in the system, and to assign parts to facilities like workstations and AGVs. [22]

Because of the complexity of an FMS, it is not very useful to find the optimal solution in a scheduling problem since changes often occur with the system status (e.g., arrival of urgent parts, machine breakdown, and so on). Therefore, it is not desirable to design an optimal scheduler and spend lots of computer time, but rather to develop a flexible scheduling tool to monitor the system and make decisions in order to achieve the best effect by taking all performance measures into consideration. [23] The developed tool has to be easy to use and to react to changes in real time. It has to be expressed in terms of parameters that have to be chosen in accordance with the system objectives, which depend on the production situation.

In short term scheduling, dispatching rules are usually used for dynamically control part movement. However, no single dispatching rule has been shown to consistently produce better results than other rules under a variety of shop configurations and operating controls [24]. Many attempts have been made to combine simple dispatching rules to improve their efficiency. It has been recognized that a combination of simple dispatching rules or a combination of heuristics including simple dispatching rules, in most cases, could lead to better results than using individual dispatching rules.

Wen *et al.* (1996) [25] proposed a dynamic routing method using a fuzzy part-family formation approach, which was combined with a certainty factor procedure, to suggest the favorable route in a multi-cell FMS. A simulation model was constructed to compare the performance of the proposed dynamic routing method with the performance of the fixed routing method. The only dispatching rule used in the model was FCFS (first come first serve). In summary, the fuzzy clustering algorithm provided extra information that was not available in conventional algorithms. Yu *et al.* (1999) [26] proposed a fuzzy inference-based scheduling decision approach for FMS with multiple objectives, which consisted of different and dynamic preference levels. The preference levels were dynamic because the priority given to different objectives might change depending on the conditions of the production environment, such as urgent part orders. A multiple criteria scheduling decision was then made, using the partitioned combination of the preference levels.

2.2.1 Dynamic scheduling

Approaches to production scheduling and rescheduling in a dynamic environment can be classified into three main categories [27]:

- Completely reactive approaches
- Predictive-reactive approaches
- Robust scheduling.

In completely reactive approaches, no firm schedule is generated in advance and decisions are made locally in real-time. The dynamic scheduling problem is viewed as a queuing system by considering each machine as a server. In the queuing system the scheduling decisions are made as events occur, thus the system cannot be used to create predictive schedules, and so cannot benefit from advances in optimization technologies. In this situation, simulation has been found to be a desirable technique.

In predictive-reactive scheduling, a predictive schedule is generated in advance of execution using available information in the shop floor. When disruptions occur during execution, the predictive schedule needs to be modified in order to take into account the new events. These scheduling/ rescheduling methods implicitly treat a dynamic scheduling problem as a series of static problems, which are resolved on a rolling horizon basis. Predictive and reactive scheduling may thus be seen as complementary activities. An important issue in which this complementary relationship between predictive and reactive scheduling is highlighted is that of schedule robustness. [28]

In the robust scheduling approach, the predictive schedule is built using available information on the disruptions that are likely to occur during execution of the schedule to minimize deviation between the performance measure values of the realized and predictive schedules. Robustness is a desirable attribute of a predictive schedule as it focuses on minimizing the effects of disruptions on the performance measures.

The scheduling objective is to maximize shop efficiency (minimizing makespan), and at the same time minimize system impact caused by schedule changes. Cowling and Johansson (2001) [28] proposed two measures, utility and stability, to decide whether to repair a schedule or reschedule from scratch, and surveyed rescheduling and schedule-repair techniques. Utility is the improvement of the objective function resulting from repair, and stability measures the deviation from the original schedule.

In today's business environment, due to highly competitive and dynamic market conditions, it has become necessary for manufacturing systems to have quick response times and high flexibility. Flexible manufacturing systems (FMS's) have gained attention in response to this challenge. In FMS's to improve delivery performance, companies often resort to rescheduling jobs on the shop floor, selectively expediting the more urgent ones while de-expediting others. The intention to obtain better delivery performance is not the only reason for

rescheduling. This policy is sometimes inevitable because of the rush jobs, lack of raw material, the unavailability of components for assembly, staffing problems, etc.

During the last few years, successful results have been achieved in using multi-agents to solve complex dynamic scheduling problems. Multi-agents are distributed and autonomous systems that support reactivity, and are robust against failures locally and globally. Agents can locally react to local changes faster than a centralized system could in an ever-changing environment, and have the ability to cooperate to define a global feasible schedule. The application of multi-agents leads to dynamic scheduling systems that are emergent rather than planned, and concurrent rather than sequential. [29]

The fundamental objective of multi-agent based-scheduling systems is to provide robustness to disturbances, adaptability and flexibility to rapid changes, and an efficient use of resources.

2.3 FMS Control Software

Monitoring and controlling an FMS is a complex and important task. In fact, modern computerized manufacturing systems have lagged far behind what could be achieved with existing technology [30]. The cost of such systems is often too high and it should be justified by a return on investment, which can be gained only if the required system flexibility is assured. Flexibility is the capability to react to the continuous market changes through easy production adaptability.

A number of authors have stated many crucial needs for the success of FMS control software. Joshi and Smith (1994) [30] define, as key factors for the success are, reduction in costs of the software systems, increase in flexibility and the ability of using it, seamless integration, and reusability. Zhang et al. (1999) [31] assert that the minimum desirable characteristics for flexible manufacturing systems are reusability, reconfigurability, and scalability. Venkatesh and Zhou

(1998) [32] consider that the functional objective of FMS control software is to maintain high system utilization and throughput. In addition, reusable, modifiable, and extendible control software need to be designed. Aguirre, Weston, Martin, and Ajuria (1999) [33] believe that major obstacles arise because of difficulties associated with generating manufacturing control systems. Finally, it is agreed that software architecture for FMS should embed standard usage, object-orientation, and inherent support for flexibility.

Actually, the modeling phase plays a crucial role during Flexible Manufacturing Control System design. In the literature several modeling methodologies can be found. Among them, the most relevant are queuing networks (Buzacott and Shanthikumar, 1980) [34], IDEF0 (Gong and Lin, 1994) [35], Petri nets (D'Souza and Khator, 1994) [36], discrete event simulation (Haddock, 1995) [37], and object-oriented techniques (Booth, 1998;Venkatesh and Zhou, 1998) [38].

In fact, in the last ten years the supremacy of object-oriented (OO) approaches with respect to reusability, extendibility, and modifiability in control software design has been largely proven.

2.3.1 Object-oriented modeling, design, and programming

One main activity in software development is the achievement of a conceptual model of it. Furthermore, for the maintenance and modification of such software, the comprehension of its global organization, the relationships among its components and their response to external changes are essential phases. For these reasons, in the object-oriented paradigm the design, modeling, and analysis phases of the software system development play a crucial role. Object-modeling techniques (OMT) are widely recognized as powerful instruments in preliminary phases of software design. This is also because of the existing commercial tools, which aid the designer through visual modeling, code generation, and reverse engineering.

The OO paradigm for software development allows the designer to achieve three main advantages. The first concerns software maintenance; the program results are simpler and easier to understand. The programmer can view only the detail degree he retains to be important. The second is relative to code modifications; often it is sufficient to insert a new class into the system without otherwise changing anything. In fact, a new class inherits characteristics of its parent class and the designer must only add the new characteristics of the class. The third advantage is regarding class reutilizing; once a class has been defined, it can be reused in other programs with no relevant code changes [39].

2.3.1.1 Transactions

A *transaction* is a unit of work that is done as a single, atomic operation, in which the operation succeeds or fails as a whole. For example, consider transferring money from one bank account to another, which involves two steps: withdrawing the money from the first account and depositing it in the second. It is important that both steps succeed; it is not acceptable for one step to succeed and the other to fail. A database that supports transactions is able to guarantee this. [40]

Either being committed or being rolled back can complete transactions. When a transaction is committed, the changes made in that transaction are made permanent. When a transaction is rolled back, the affected rows are returned to the state they were in before the transaction was started. To extend the account transfer example, an application executes one SQL statement to debit the first account and a different SQL statement to credit the second account. If both statements succeed, the application then commits the transaction. But if either statement fails for any reason, the application rolls back the transaction. In either case, the application guarantees a consistent state at the end of the transaction [41].

Microsoft Transaction Server (MTS) is a component-based transaction processing system for developing, deploying, and managing high performance, scalable, and robust enterprise, Internet, and Intranet server applications. Transaction Server defines an application-programming model for developing distributed component-based applications.

Using MTS in database transactions provides additional safety in mission-critical modules. A distributed system with a number of COM objects running in several PCs requires an error-free execution, the effort in searching for the error may be great and time consuming. Transaction Server provides safe data communication between business objects.

2.4 Modeling Techniques

As the object oriented programming languages and HTML/ASP based web browsers tend to become the common platform in software development and internet, the need of modeling manufacturing components to the standards has arisen. Also the complexity of solutions has so increased that without a modeled start it is almost impossible to create a well functioning application. There are a number of software modeling techniques such as IDEF0, IDEF1X, and UML (Unified Modeling Language) available in the literature. The following sub-sections are intended to give a brief description of these. [40]

2.4.1 IDEF0 and IDEF1X

Integration Definition Function Modeling (IDEF0) is announced as a Federal Information Processing Standard of United States on 1981, and is based on the Air Force Wright Aeronautical Laboratories Integrated Computer-Aided Manufacturing Architecture [42]. It describes the IDEF0 modeling language (semantics and syntax), and associated rules and techniques, for developing structured graphical representations of a system or enterprise. Use of this standard permits the construction of models comprising system functions (activities,

actions, processes, operations), functional relationships, and data (information or objects) that support systems integration. IDEF0 models are composed of three types of information: graphic diagrams, text, and glossary. These diagram types are cross-referenced to each other. The graphic diagram is the major component of an IDEF0 model, containing boxes, arrows, box/arrow interconnections and associated relationships. Boxes represent each major function of a subject.

Integration Definition for Function Modeling standard is used to model a wide variety of automated and non-automated systems. For new systems, IDEF0 may be used first to define the requirements and specify the functions, and then to design an implementation that meets the requirements and performs the functions. For existing systems, IDEF0 can be used to analyze the functions the system that performs and to record the mechanisms (means) by which these are done.

Function box representation is the core of IDEF0 model. Each side of the function box has a standard meaning in terms of box/arrow relationships. The side of the box with which an arrow interfaces reflects the arrow's role. Arrows entering the left side of the box are inputs. Inputs are transformed or consumed by the function to produce outputs. Arrows entering the box on the top are controls. Controls specify the conditions required for the function to produce correct outputs. Arrows leaving a box on the right side are outputs. Outputs are the data or objects produced by the function. Figure 2.6 shows a typical function representation in IDEF0 [43].

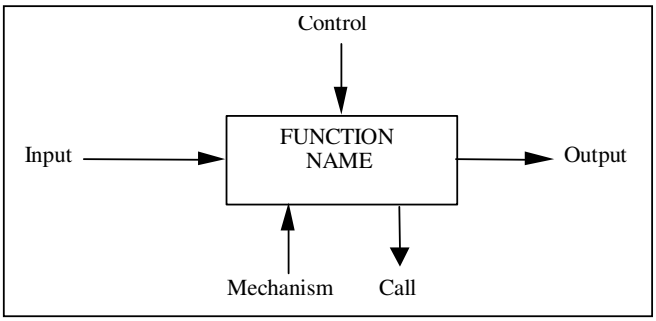

Figure 2.6 Function representation in IDEF0

One of the most important features of IDEF0 as a modeling concept is that it gradually introduces greater and greater levels of detail through the diagram structure comprising the model. In this way, communication is enhanced by providing the reader with a well-bounded topic with a manageable amount of detail to learn from each diagram. IDEF0 allows defining the functions and relationships of program modules through a systematic manner early in the design phase. IDEF0 may be combined with the IDEF1X to build the information model as well.

The Integration Definition for Information Modeling (IDEF1X) is announced as a Federal Information Processing Standard of United States on November 1985 [44]. It describes the IDEF1X modeling language (semantics and syntax) and associated rules and techniques, for developing a logical model of data. IDEF1X is used to produce a graphical information model, which represents the structure and semantics of information within an environment or system. Use of the standard permits the construction of semantic data models, which may serve to support the management of data as a resource, the integration of information systems, and especially the building of computer databases. In an IDEF1X view, an “attribute” represents a type of characteristic or property associated with a set of real or abstract things (people, objects, places, events, ideas, combinations of things, etc.). An entity must have an attribute or combination of attributes whose values uniquely identify every instance of the entity. These attributes form the “primary-key” of the entity. In this way, the primary key or the combination of primary keys define a unique entity through which the attributes of its own and related entities can be reached. Figure 2.7 depicts the attribute and primary key syntax in IDEF1X [43].

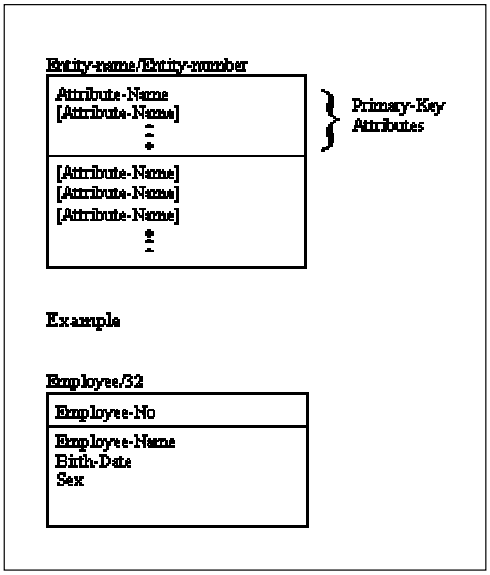

Figure 2.7 Attribute and primary key syntax in IDEF1X

In an IDEF1X view, connection relationships are used to represent associations between entities. A "connection relationship" (also referred as a "parent-child relationship") is an association or connection between entities in which each instance of one entity, referred to as the parent entity, is associated with zero, one, or more instances of the second entity, referred to as the child entity, and each instance of the child entity is associated with zero or one instance of the parent entity. A solid line depicts an identifying relationship between the parent and child entities as in Figure 2.8 [43]. If an identifying relationship exists, the child entity is always an identifier-dependent entity, represented by a rounded corner box, and the primary key attributes of the parent entity are also migrated primary key attributes of the child entity.

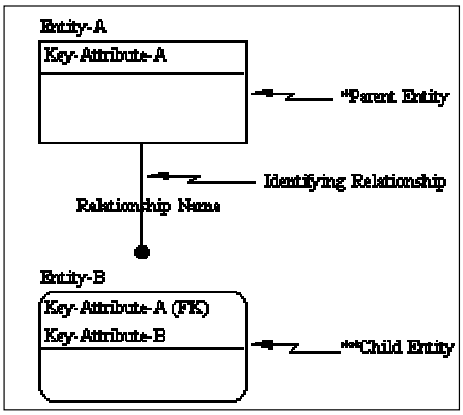

Figure 2.8 Identifying Relationship Syntax in IDEF1X

IDEF0 and IDEF1X are powerful tools in modeling functionality, information flow and structure of an enterprise. If used together they form a full picture of working modules and their communication.

2.4.2. UML

The Unified Modeling Language (UML) is an international standard notation for object-oriented analysis and design. UML is probably the most widely known and used notation for object-oriented analysis and design. It is the result of the merger of several early contributions to object-oriented methods. UML is the product of a long history of ideas in the computer science and software engineering area. The development of UML began in October of 1994 when Grady Booch and Jim Rumbaugh of Rational Software Corporation started their work on unifying the Booch and OMT (Object Modeling Technique) methods [45].

A key motivation behind the development of the UML has been to integrate the best practices in the industry, encompassing widely varying views

based on levels of abstraction, domains, architectures, life cycle stages, implementation technologies, etc.

The primary design goals of the UML are as follows:

- Provide users with a ready-to-use, expressive visual modeling language to develop and exchange meaningful models.
- Furnish extensibility and specialization mechanisms to extend the core concepts.
- Support specifications that are independent of particular programming languages and development processes.
- Provide a formal basis for understanding the modeling language.
- Encourage the growth of the object tools market.
- Support higher-level development concepts such as components, collaborations, frameworks and patterns.
- Advance the state of the industry by enabling object visual modeling tool interoperability.

There are several new concepts included in UML, containing:

- Extensibility mechanisms (stereotypes, tagged values, and constraints), threads and processes
- Distribution and concurrency (e.g., for modeling ActiveX/DCOM and CORBA)
- Patterns/collaborations
- Activity diagrams (for business process modeling)
- Refinement (to handle relationships between levels of abstraction)
- Interfaces and components
- A constraint language.

Many of these ideas mentioned above were present in various individual methods and theories but UML brings them together into a coherent whole.

In its current form UML is comprised of two major components: a meta-model and a notation. UML is unique in that it has a standard data representation, which is called the meta-model. It describes the objects, attributes, and relationships necessary to represent the concepts of UML within a software application. The UML notation is rich and full bodied. It is comprised of two major subdivisions.

There is a notation for modeling the static elements of a design such as classes, attributes, and relationships. There is also a notation for modeling the dynamic elements of a design such as objects, messages, and finite state machines. Static models are presented in class diagrams. The purpose of a class diagram is to depict the classes within a model. In an object-oriented application, classes have attributes (member variables), operations (member functions) and relationships with other classes. A class icon is simply a rectangle divided into three compartments. The topmost compartment contains the name of the class. The middle compartment contains a list of attributes (member variables), and the bottom compartment contains a list of operations (member functions). Figure 2.9 shows a typical UML description of a class that represents a circle [45].

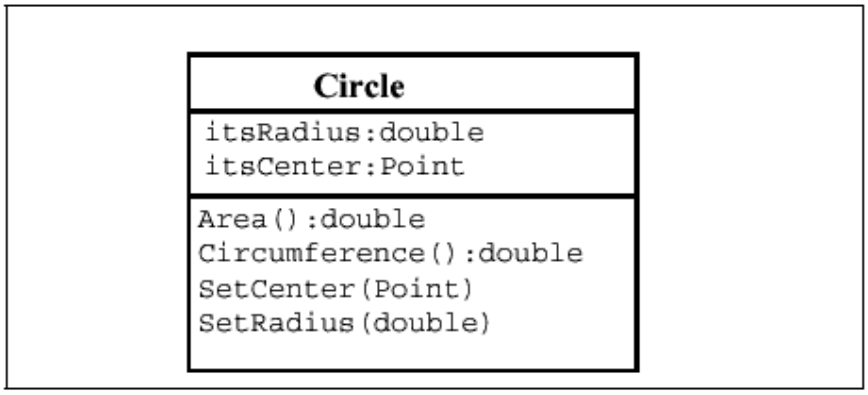

Figure 2.9 Circle Class

Also, Figure 2.10 shows how classes and instances are represented [45], however unfortunately, UML does not distinguish adequately between types and classes notationally; but the stereotype «type» can be added to the class icon to show the difference. Stereotypes are tags that can be added to objects to classify them in various ways. Stereotypes can be added to classes, associations, operations, use cases, packages, and so on. Stereotypes make the language extensible, adding extra meaning to the basic pieces of syntax. Notice that instance names are always underlined; otherwise the notation for an instance is exactly the same as that for a class (type).

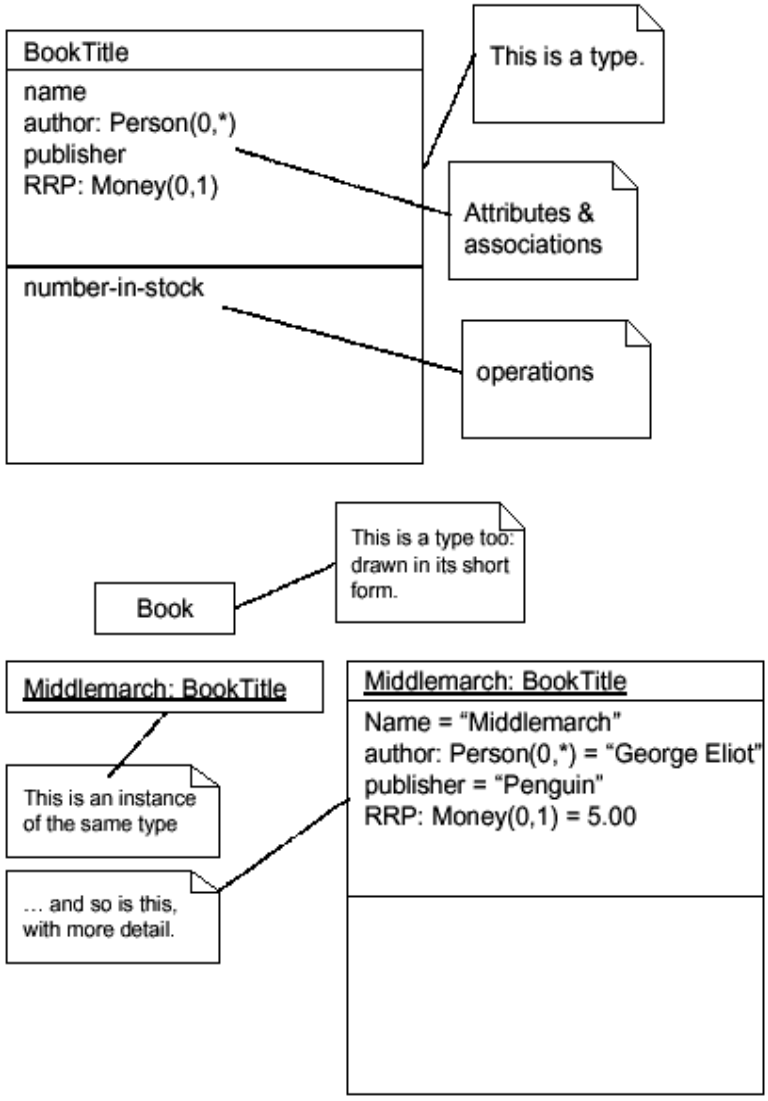

Figure 2.10 Types and Their Instances.

The operations and attributes of an object are called its features. The features (and possibly the name) constitute the signature of the object. It is often useful to think of the features of an object as responsibilities. Attributes and associations are responsibilities for knowing. Operations are responsibilities for doing. With an object-oriented approach to data management it seems reasonable to adopt the view that there are two kinds of object, called domain objects and application objects. Domain objects represent those aspects of the system

relatively stable or generic (in supplying services to many applications). Application objects are those, which can be expected to vary from installation to installation or from time to time quite rapidly.

There are some types of attributes that objects posses. Variable attributes are the opposite of fixed attributes and they are the default. Common attributes require that all instances have the same value, again without necessarily knowing what it is. Unique attributes are the opposite of common ones; each instance has a different value. A well-known example is a primary key in a database table. The default is neither common nor unique. The notation is one of the following: {variable}, {fixed}, {common}, {unique}, {fixed, common}, {fixed, unique}, {variable, common}, {variable, unique}.

The Unified Modeling Language (UML) provides system architects working on object analysis and design with one consistent language for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling. Also, in terms of the views of a model, it defines some graphical diagrams such as use case diagrams, class diagrams, collaboration diagrams, component diagrams, etc. As a whole, UML provides the following:

- Semantics and notation to address a wide variety of contemporary modeling issues in a direct and economical fashion.
- Semantics to address certain expected future modeling issues, specifically related to component technology, distributed computing, frameworks, and executability.
- Extensibility mechanisms so individual projects can extend the metamodel for their application at low cost.
- Extensibility mechanisms so that future modeling approaches could be grown on top of the UML.
- Semantics to facilitate model interchange among a variety of tools.

- Semantics to specify the interface to repositories for the sharing and storage of model artifacts.

2.5 Distributed Internet Applications (DNA) Architecture:

Microsoft® Windows® Distributed InterNet Applications architecture for Manufacturing (Windows DNA for Manufacturing) is a framework that allows manufacturing software applications to integrate seamlessly with one another [46]. It enables functional Plug and Play interchange between applications, from Enterprise Resources Planning (ERP) to the shop floor. Windows DNA for Manufacturing can be delivered over any network as well as link to legacy computer systems. Windows DNA also allows to build scaleable client/server systems that leverage new and existing Enterprise Resources Planning (ERP), and Manufacturing Execution (MES) systems; DCS and SCADA based applications, as well as the new breed of PC-based, shop-floor control systems[47].

The goal of Windows® Distributed interNet Applications (DNA) architecture is to create a framework for building applications based on the Windows platform that unifies and integrates the personal computer and the Internet. Windows DNA enables a computing model that fully capitalizes on the capabilities of both the personal computer and the Internet.

At the highest level, Windows DNA integrates these two worlds by enabling computers to interoperate and cooperate equally well across both corporate and public networks and by deeply integrating the core services into the operating system. This allows developers to more easily create sophisticated network-aware applications that can support large numbers of users. More important, Windows DNA provides an interoperability framework based on open protocols and published interfaces that allow customers to extend existing systems with new functionality such as the Web. This same open model provides extensibility "hooks," so third parties can realize new business opportunities by creating compatible products that extend the architecture. Windows DNA

applications use a standard set of Windows-based services that address the requirements of all tiers of modern distributed applications: user interface and navigation, business processes, and storage.

The heart of Windows DNA is the integration of Web and client/server application development models through a Common Object Model (COM). Windows DNA uses a common set of services such as components, Dynamic HTML, Web browser and server, scripting, transactions, message queuing, security, directory, database and data access, systems management, and user interface. These services are exposed in a unified way at all tiers for applications to use.

Windows Distributed interNet Application (DNA) is a general architecture that describes how to build three-tiered applications for the Windows platform. The purpose of this introduction is to provide developers with a general methodology for designing and building Windows DNA applications.

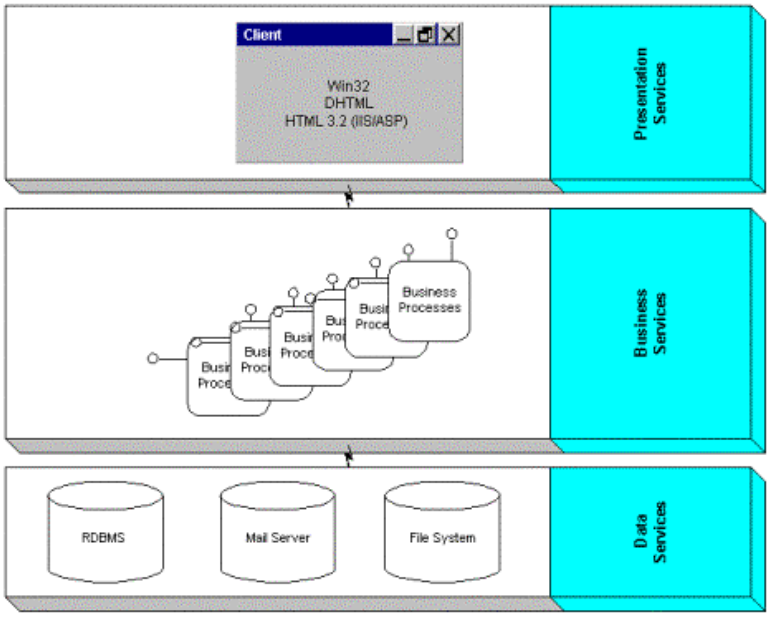

Figure 2.11 Three-tiered application

A three-tiered application is an application whose functionality can be segmented into three logical tiers of functionality: presentation services, business services, and data services.

The presentation services tier is responsible for:

- Gathering information from the user.
- Sending the user information to the business services for processing.
- Receiving the results of the business services processing.
- Presenting those results to the user.
- The business services tier is responsible for:
- Receiving input from the presentation tier.
- Interacting with the data services to perform the business operations that the application was designed to automate (for example, order processing).
- Sending the processed results to the presentation tier.

The data services tier is responsible for the:

- Storage of data.
- Retrieval of data.
- Maintenance of data.
- Integrity of data.

Data services come in a variety of shapes and sizes, including relational database management systems (RDBMSs) like Microsoft SQL Server, e-mail servers like Microsoft Exchange Server, and file systems like the NTFS File System.

By contrast, a two-tiered application is an application whose functionality can only be segmented into two logical tiers, presentation services and data services, and while the responsibilities of the data services are the same for two-

tiered and three-tiered systems, the responsibilities of the presentation services are not. In a three-tiered application, the presentation services are responsible for gathering information from the user, sending the user information to the business services for processing, receiving the results of the business services processing, and presenting those results to the user. However, the presentation services of a two-tiered application are slightly different. The presentation services of a two-tiered application are responsible for gathering information from the user, interacting with the data services to perform the application's business operations, and presenting the results of those operations to the user.

A two-tiered application is an application whose functionality can only be segmented into two logical tiers of functionality: presentation services and data services.

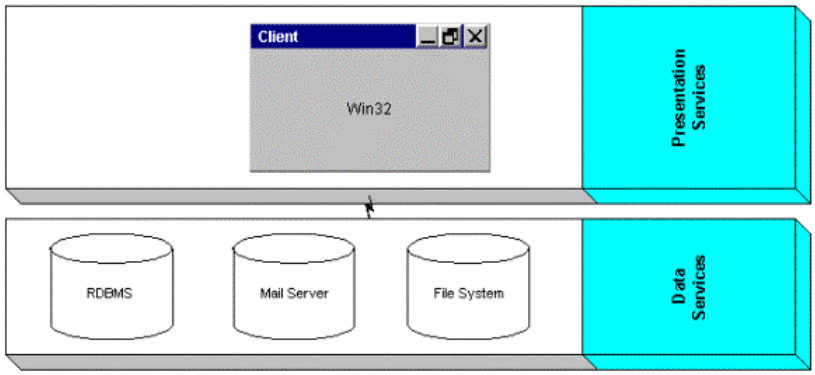

Figure 2.12 Two-tiered application

2.5.1 Windows DNA Design Objectives

Windows DNA is a general application architecture that describes how to build three-tiered applications for the Windows platform. There are a number of factors to consider when designing and building Windows DNA applications. However, generally speaking, developers should concentrate on maximizing overall application autonomy, reliability, availability, scalability, and interoperability.

- **Autonomy:** Use emissaries, executants, and MTS role-based security to prevent clients from accessing critical resources directly.

- **Reliability:** Use MTS transactions to ensure accurate results in a multi-user environment.

- **Availability:** Eliminate single points of failure by implementing redundant hardware and software systems, which include using MSCS for redundancy solutions involving clustering. Use MSMQ's store and forward, guaranteed delivery, and dynamic routing features to simulate increased network availability.

- **Scalability:** Minimize resource acquisition times by using MTS to share resources among users and to pool resources in short supply. Minimize resource usage times by avoiding network interaction as part of a transaction, avoiding user input as part of the transaction, and acquiring resources late and releasing them early. For increased throughput and truly dynamic load balancing, use MSMQ for emissary-to-executant and executant-to-executant communication.

- **Interoperability:** Use ADO or OLE DB for universal data access; XML to share information with other applications; DCOM to access application on UNIX and MVS; MSMQ to access message queuing systems on other platforms; and COM Transaction Integrator (COMTI) to execute CICS or IMS transactions on MVS systems.

CHAPTER 3

SYSTEM MODEL

3.1 An Overview

A new model for an agent-based dynamic scheduling methodology for controlling manufacturing cells using Windows DNA, will be discussed in this chapter. In the previous studies the framework and detailed design model of this approach has been developed by Özgür Ünver during his Ph.D. studies "Rapid Development Methodology for Control of Manufacturing Systems Using Windows DNA" and Tolga Cangar during his M.S. studies "Development of an Agent Based Flexible Manufacturing Cell Controller Using Windows DNA".

The agent-based dynamic scheduling method for manufacturing execution system software has been implemented in Integrated Manufacturing Technologies Research Group (IMTRG), Middle East Technical University Mechanical Engineering Department Computer Integrated Manufacturing (METUCIM) laboratory. User, Business, and Data Services of the "Agent Version 1.1" has been mostly written under Visual Basic 6.0. For the communication and event driven messaging of agents, Microsoft Message Queue Server (MSMQ) is used, stateless objects for database search and update has been deployed in Microsoft Transaction Server (MTS). The common database of the "Agent Version 1.1" has been constructed using SQL Server 7.0. Internet Information Server (IIS) has been used to grant access to the web sites ASP and HTML pages, which are designed in Visual InterDev 6.0, a product of Microsoft Visual Studio. In summary, the "Agent Version 1.1" package includes:

- 7 agent controller EXE programs to drive the hardware components accompanying 48 channels I/O card, RS 232 serial communications, MSMQ services and MTS components,
- 18 MTS components for database search, update, addition and modification, 1 stateless part agent component, and 1 messenger for MSMQ services,
- 1 SQL Server Database on Internet Information Server with 21 tables and 70 stored procedures,
- 1 complete web-site with 21 ASP and 8 HTML pages including a "live cam" link to control, monitor, and manage the enterprise including detailed help and information sources.

Before proceeding with the software modules, it is necessary to give information about the hardware of the manufacturing system, namely the METUCIM.

3.2 METUCIM Test-bed

The flexible manufacturing system at the Mechanical Engineering Department basically consists of a single manufacturing cell. The main material handling system is the closed loop buffer and the 6 axis robot. Also there is a static buffer for loading and unloading parts to the system. The movement of the robot between the CNC Turning- and CNC Milling Machine is accomplished by the Pneumatic Linear Robot Drive (PLRD). A general view of the system is given in Figure 3.1.

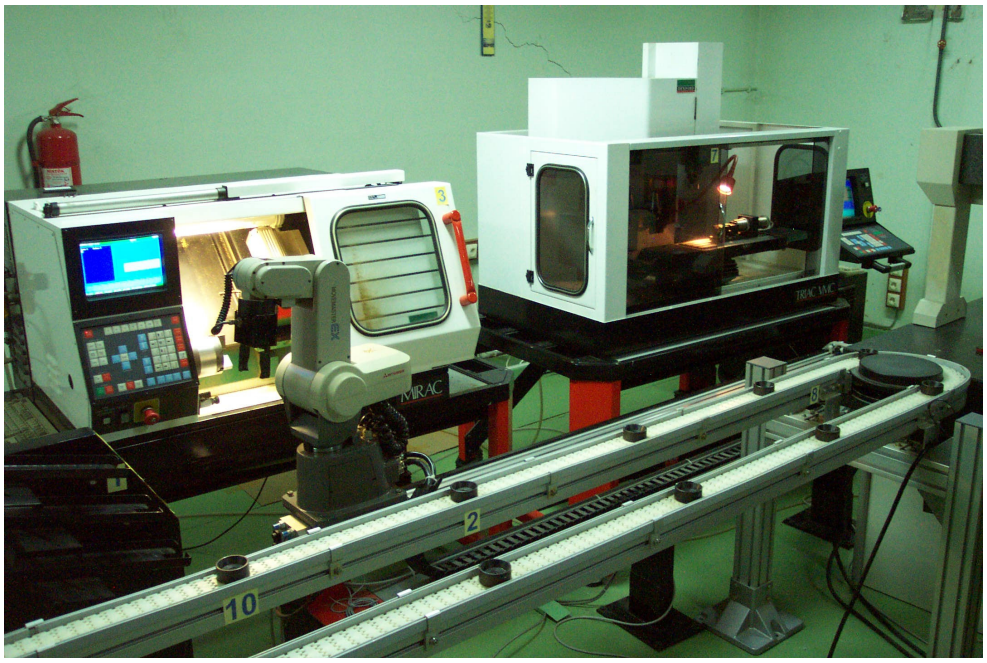

Figure 3.1 General view of METUCIM

Functionality, properties and capabilities of the manufacturing, transport and quality control hardware can be summarized as:

1. **CNC Turning Machine:** Mirac/Denford/UK. PC based, medium duty lathe having 2 simultaneously controlled axes. Equipped with a turret having 8 stations. Door and chuck are pneumatically powered. Can handle typically bars up to 50 mm in diameter and 150 mm in length, speeds up to 2500 rpm. Has a user-friendly built-in interface to visualize and debug part programs. The control is via standard RS 232 serial communication port and I/O card at a single sensor channel. Channel state OFF indicates that there is no part program running, or the task is finished. Channel state is ON when there is an active program running. “M63” and “M65” codes make the channel ON and OFF respectively.

2. **CNC Milling Machine:** Triac/Denford/UK. PC based, medium duty milling machine having 3 simultaneously controlled axes. Equipped with an automatic tool magazine with 6 stations. Door, chuck and tool magazine are pneumatically powered. Can handle parts up to 200 mm in width and 500 mm in length, speeds up to 2500 rpm. Has a user friendly built-in interface to visualize and debug part programs. The control is via standard RS 232 serial communication port and I/O card at a single sensor channel. Channel state OFF indicates that there is no part program running, or the task is finished. Channel state is ON when there is an active program running. “M62” and “M64” codes make the channel ON and OFF respectively.

3. **Closed Loop Buffer:** SKF/UK. Unidirectional, constant speed, closed loop buffer having 14 cups. Typically, it can handle cylindrical parts up to 50 mm in diameter. Makes a full rotation in 1.5 minutes approximately. Driven by a motor with gearbox. The control is via 48 channel I/O card. Has one operate channel and one counter channel. When the operate channel is ON, it starts to rotate and stops when the channel is OFF, the counter channel is used to count the cups passed.

4. **Robot:** Movemaster EX/Mitsubishi/Japan. 6 axis controlled material handling robot. Capable of handling bars of 50 mm in diameter, weight of 3 kg approximately. The control is by storing positions taught by the user in its EPROM and they can be executed by external triggering of program commands through RS232 connection from the computer. A DSR (data set ready) signal from the serial port indicates that there is no active program running or the task is finished.

5. **Pneumatic Linear Robot Drive (PLRD):** FESTO/Germany. Pneumatically powered linear drive for the robot. Has a movement range of 2m. Has two stop positions at both ends only. In METUCIM configuration it is used to move the robot from CNC Turning to CNC

Milling neighborhood. The control is via 48 channel I/O card. Has two operate- and two sensor channels. When the first operate channel is triggered and immediately released it moves to right and vice versa for the second. Sensor channels on the left- and right positions indicate ON when the robot is at left and right ends of its range respectively.

6. **Static Buffer (AGV):** Buffer used for in and out loading to the cell. Has 3 input stations which can handle bars of 70-90-100 mm, and has 3 output stations. Is not physically connected or driven by a computer, but as the agent, status information is kept. Although it has no computer control and moving capabilities, it is modeled as an AGV in the system.

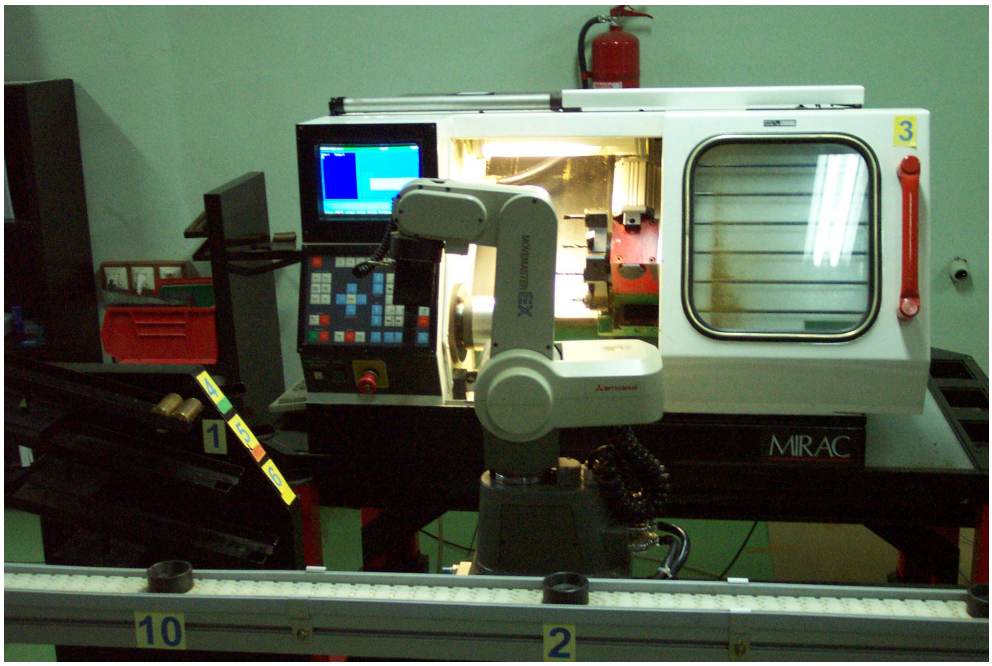

Figure 3.2 A closer look at the static buffer, CNC Turning machine and conveyor

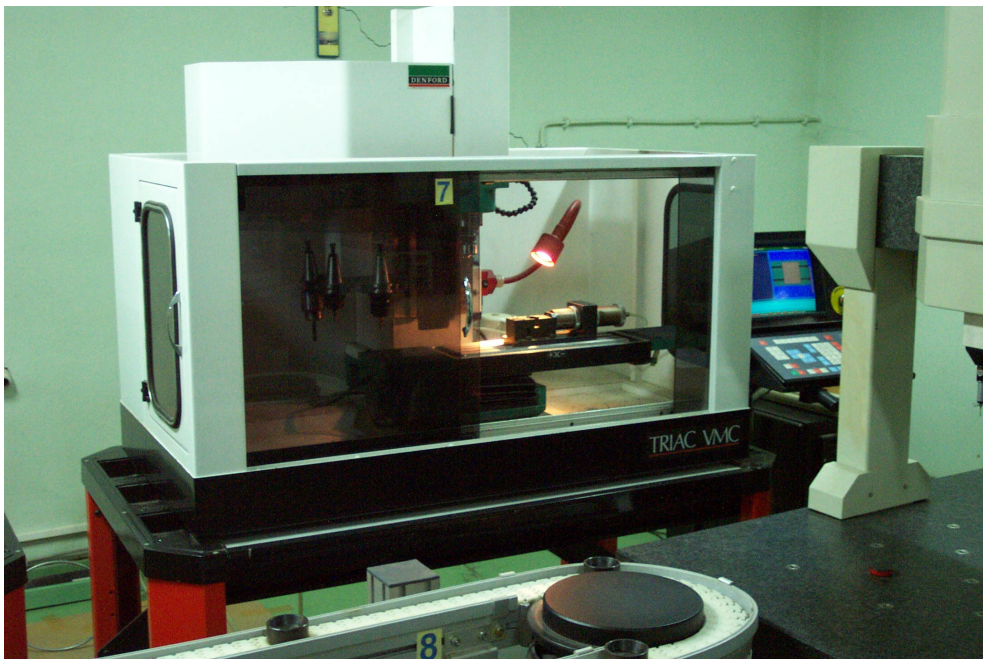

Figure 3.3 A closer look at the CNC Milling machine

Computers are essential parts of the METUCIM. Their properties and functions are summarized as:

1. **Agent PC:** Pentium III 450, 256 MB RAM, 6.2 GB hard-disk with 17" monitor. Has 48 channels I/O card installed. Responsible from driving all agents via MSMQ and MTS services and two serial ports. Running under Windows NT 4.0.

2. **Robot Host PC:** Pentium 100, 32 MB RAM, 3.2 GB hard-disk drive with 14" monitor. Responsible from driving the robot from its serial port, connected to the Agent PC via Ethernet. Receives and sends task status to the agent PC. Running under Windows 95.

3. **Primary Backup Controller PC (SQL Server):** Pentium III 450, 128 MB RAM, 6.2 GB hard-disk with 17" monitor. Serves as the backup Enterprise Controller. Internet Information Server, Microsoft Transaction Services (MTS) and Message Queue Services as the main backup and FTP site. Running under Windows NT 4.0.

4. **Enterprise Controller PC (IIS Server):** Pentium III 450, 392 MB RAM, 6.2 GB hard-disk with 17" monitor. The main server of Enterprise Controller, Internet Information Server, Microsoft Transaction Services (MTS) and Message Queue Services. Running under Windows NT 4.0.

A complete layout of METUCIM with the hardware and computers is given in Figure 3.4.

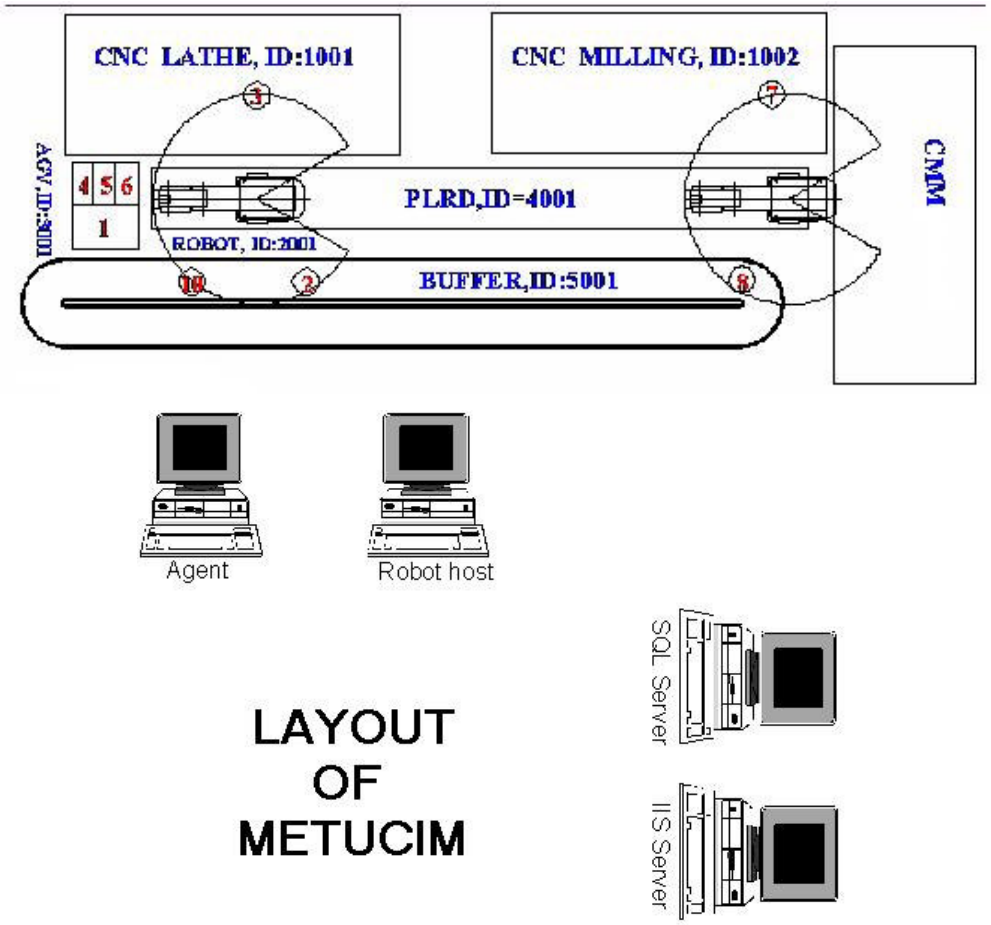

Figure 3.4 Complete Layout of METUCIM with computers

METUCIM laboratory has a Programmable Logic Controller (PLC) drilling and pressing station set with an additional Mitsubishi Movemaster EX robot, and one additional buffer. It also has a new Image Processing unit for educational purposes. This hardware is not related with the work discussed here, so it is beyond the scope.

3.3 Software Model

The previous software and the details of new developed software are given in the following sections.

3.3.1 Previous Software and Configuration

The system in METUCIM was built in 1992 and lately developed in 2000. It had designed for implementing the agent based shop floor control technology. The system was having a simple scheduling method just for production of single parts and it was lacking of a dynamic scheduling methodology for real manufacturing environment. In the real manufacturing environment, there are productions in batches, work orders have priorities (not always in a documented way but in practical), and urgencies may occur that the manufacturer decides to produce that order as soon as possible.

1. In the previous configuration work order can be given for only one part. It was impossible to manufacture the parts in batches.

2. The queuing procedure was simply the First Come First Served (FCFS) rule. The messages were automatically put in the first empty compartment of the manufacturing queue. In the FCFS rule, if the queue is open and event notification is enabled in that program module, the message at the cursor location is retrieved. If there are any remaining messages in the subsequent compartments, they move automatically to the cursor location and wait to be processed.

3. There was no way to interfere to the manufacturing queue. It was impossible to manufacture a part before the parts that had already added to the manufacturing queue. Even if an order is urgent, it had to wait.

4. There were bugs in programs of the system, so that unexpected errors have occurring. The system was not reliable.

5. There was no “live cam” application in web site of the system. The user could not see the manufacturing processes of his order. The system was lack of control.

6. The bidding was slow because of unnecessary messages between the customer and server agents. The customer was sending messages to all agents including irrelevant ones for that task.

7. The computers and the network hubs were old, so the processing and communication was slow. The cable connections were not healthy.

3.3.2 Modifications Done

“Dynamic scheduling” is the key word of today’s manufacturing strategy. The SMEs must be dynamic for the demands of customers to have a portion in manufacturing market. With this philosophy, the main developments that have been made in this study are listed below:

1. By this program, it is possible to manufacture the parts in batches. A single work order is enough to determine the number of parts in the batch.

2. The queuing mechanism is changed. At the new application a “Priority” is defined for each work order. The work orders are queued according to their priority values so the more prior part (or batch) can be manufactured before the others. The queuing list is dynamic that, it is updated by a decreasing priority, each time a new work order is added to the queuing list.

3. A new scheduling methodology is developed, including “Urgent order” application. By this application the manufacturer or the engineer can give a work order for an urgent part (or batch).

4. Some of agent codes are added or changed in an object oriented programming manner. Adding and compiling the new components increased the system reliability and robustness.

5. “Live cam” is added to the web site that the user can see all the activities in the system. After giving the order; while the manufacturing occurs, the engineer can see and control the processes simultaneously.

6. A new bidding procedure is developed to shorten the bidding time. In this procedure the customer agent only sends the messages to the agents that its probable server, not all the agents.

7. The computers are upgraded for faster processing and communication and there is a new hub for faster data transfer in network. Connection cables are renewed.

Figure 3.5 Front view of the system, agent and robot host computers, left to right

It is aimed that the hardware architecture will support long-term flexibility also for future modifications. All computers in the system have LAN connection and are physically connected to each other. The FMS model of the METUCIM reflects a distributed nature.

3.3.2.1 Manufacturing in Batches

Today in SMEs, type of production is still not in large batches as mass production factories but day by day, the volume is increasing by the help of new technologies. CNC work stations, robots, AGVs are used to make the manufacturing in larger volumes and flexible. In our system in METUCIM we model job shops; in other words the developed agent based manufacturing control system can be adapted to a job shop. In this situation our system must make production in batches. The previous configuration did not allow this type of production. Even if they are the same type of part or not, the manufacturing engineer had had to give a work order for each part. In the new configuration, if

an order includes same type of parts in batch, the manufacturing engineer gives to system a single work order. He just gives the number of parts in the batch. The batch size is limited to five parts because of the limited manufacturing capabilities of the system in METUCIM, but can be increased easily if needed.

3.3.2.2 "Priority" Application

In this study a priority based queuing methodology is used. While the manufacturing continues, new work orders are accepted and added to the queue. To make the production flexible, the scheduling must be dynamic for changes in queue. A priority value is defined for each work order and recorded to the database with this priority. Each time when the machine tool is idle, the program part written for this agent dynamically updates the queue arranging in order decreasing priority value, then searches for a part that has the highest priority value to accept and carry out the manufacturing task. If the priority values are same for two or more orders then the queuing procedure runs on the basic First Come First Served (FCFS) rule. That, the early given work order is manufactured earlier.

Priority can be defined not only for a single part but also for a batch. The parts in batches have all same priority values. In this method it is possible to reorder the compartments of the queue by giving more prior batch orders to manufacture.

3.3.2.3 "Urgent Order" Application

By the developing industry and increased customer demands, today's manufacturing environment is very dynamic that the manufacturers must respond very quickly. While there are many work orders in manufacturing queue, an urgent order may be taken. If the manufacturing engineer (or the staff who decides to accept or to reject manufacturing of an order) decides that it will be profitable,

the job shop manufactures the urgent order as soon as possible. This is an ordinary situation for SMEs, especially in Turkey.

The first aim of this study is developing a program that can be applicable to real manufacturing systems in job shops. From this point of view, this "Urgent Order" situation is applied to the system in METUCIM. As in the case, if an urgent order is taken, it is manufactured before the orders that had given already. The urgent part is begun machining just after one part on buffer is machined. The waiting for one part is for the simulation of time passing through row material is carried by AGV.

3.3.3 Communication Between Agents

With the trend toward distributed computing in enterprise environments, it is important to have flexible and reliable communication among agents. Distributed Internet Applications (DNA) architecture is composed of independent applications that are running on different systems to communicate with each other and exchange messages even though the applications may not be running at the same time. Message Queue Server technology enables these applications or simply COM objects running at different times to communicate across heterogeneous networks and systems that may be temporarily offline. Within an enterprise, applications send messages to queues and read messages from queues.

In the design of the software model of the "Agent Version 1.1" enterprise, agent-based communication approach is applied. This allows using the components of the software as stand-alone machines/parts as well as the elements of a complete manufacturing system. The messaging procedure involves a customer-server based negotiation mechanism in which the external input is given by the user (manufacturing engineer) from a locally restricted Internet web site, thus enhancing browsing and monitoring capabilities of online data and status. The agent philosophy itself is an open architecture to implementation and selection between alternatives.

There is no preexisting hierarchy of some agents to others, a customer-server relationship is adopted for the communication. A customer is simply the agent giving some task to other, where a server is the one who accepts the task. The heterarchy dictates that all are at the same level; that is, at some time a server machine/part may be the customer to other machines/parts. In Figure 3.6, an agent based relationship between a robot and a workstation is documented. To execute a processing task coming from a part agent, CNC turning agent requests a raw material from the server robot agent to load into its chuck, similarly the robot agent -now a customer to the conveyor- requests from the conveyor to index the required part. The messaging procedure continues until the acknowledgement responses from all server agents are reached, meaning that all required pre- and after tasks are finished.

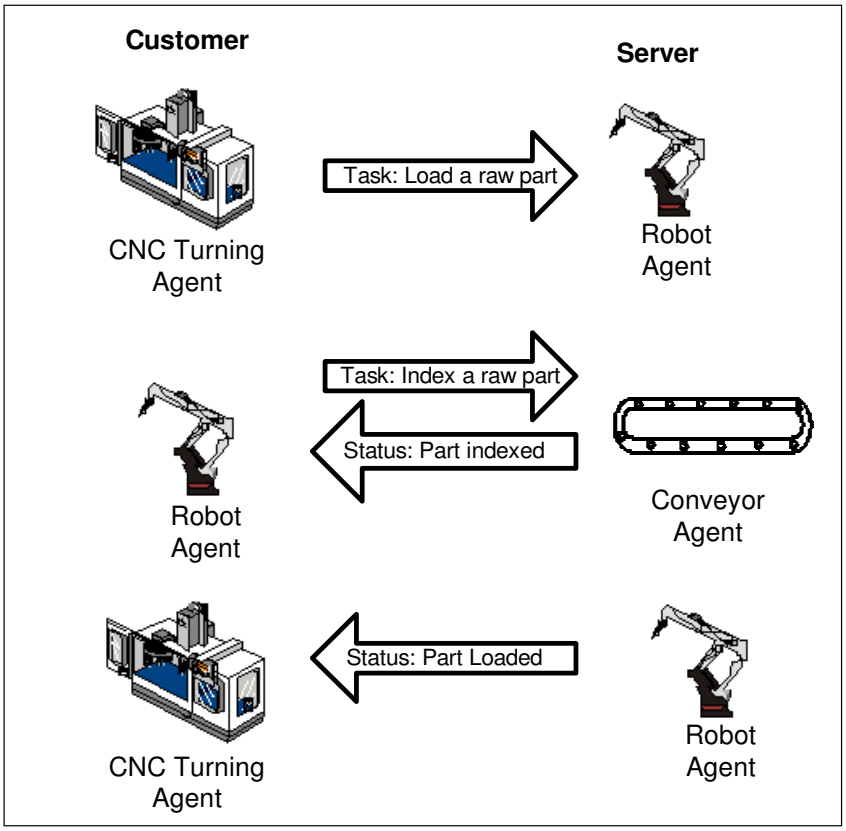

Figure 3.6 Customer-server relationship between machine agents [38]

3.3.3.1 Bidding Mechanism

The bidding procedure involves a more complex structure including task announcement, bid preparation, task offer, task commitment, task list and task status queues for each agent in the system. The queuing mechanism provides event driven function calling and program execution. It is the agent's responsibility to search for available information and report online manufacturing status on system database. A typical messaging procedure between two agents (customer-server) can be described as follows (Figure 3.7): When an agent receives a task in its task list queue, it may need to give a pre- or after-task to a remote agent. For example, when a CNC lathe receives the task to manufacture a part, initially some other robot agent has to load the part. So it sends announcements to its server agents. Announcement receivers are now responsible for preparing bids and sending them to the bids queue of the customer.

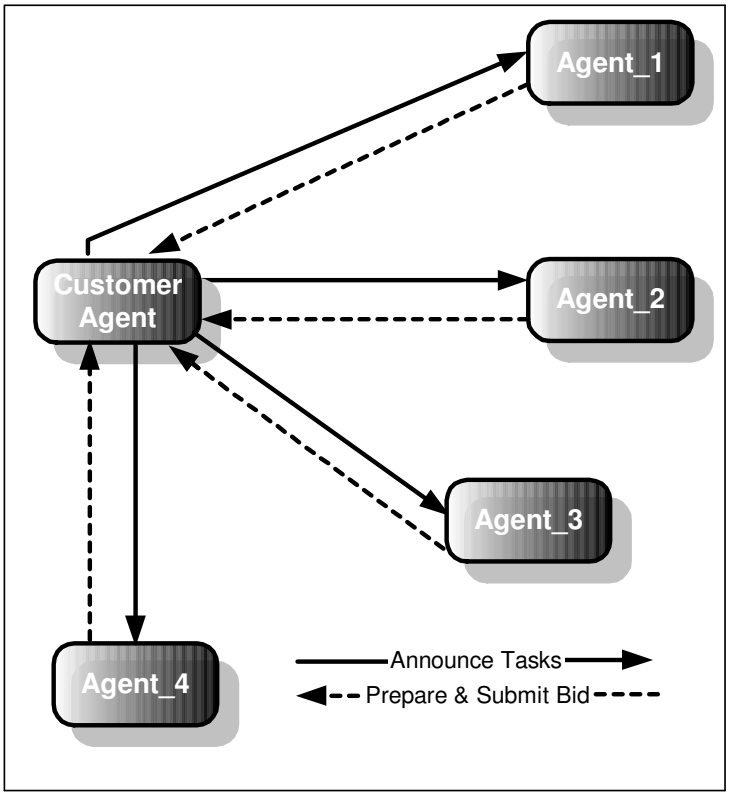

Figure 3.7 Bidding of the customer and server agents [38]

After the customer receives all bids from its servers, it selects the agent with the lowest bid value (or preferably the highest depending on the bidding scheme) and sends it to the task offers queue of the server. The server reads the task offer and sends an “Accept” or “Reject” acknowledgment to the customer. If the result is “Accept”, that means that the server agent is selected, the bidding is finished.

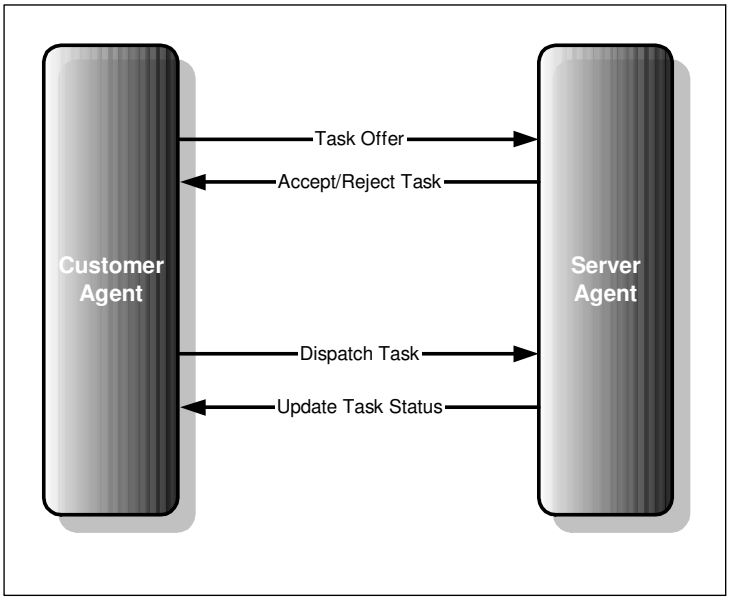

Figure 3.8 Message queues of the customer and server agents [38]

The customer sends the task to the task list queue of the selected server and waits for the status feedback. Any task dropped in the task list queue should be either finished or given an error to the customer agent’s task status queue.

The message based queuing allows the agents to communicate and select others to execute its own pre- and after-tasks. In actual manufacturing environments most of the agents will have limited, probably single servers and/or customers. For that case, the bidding algorithm is to be omitted. For example, when a part agent at some sequence of its operations has only a single workstation to produce itself, it would only send the task offer to that machine's task offers

queue and wait for the "Accept" acknowledgement message. If the answer is "Accept", it sends the task to the servers task list and wait for task status to complete its manufacturing. The relationship between the customer and server provides a simplification to NP-hard scheduling problems.

3.3.3.2 Bid preparation algorithm

There are a number of optimization approaches to the dynamic scheduling of heterarchical multi agent systems. In contract net model, the customer agent identifies and announces a task to be done and criteria for bids. The bidding servers send bids for the announced task to the customer. The bid satisfying some criteria is awarded and the task is given to the selected server. The preparation of the bid value, in other terms, becomes the determinant scheduling approach to the problem. In the contract net bidding algorithm, there is no pre-defined master schedule to be followed, but any decision is made at the real time, looking at the current situation. There are also alternate measurement criteria such as shortest processing time (SPT), earliest due date (EDD), least slack (LS), maximum utilization, and minimum waiting time.

In this research the basic approach for the calculation of the minimum expected completion time is preferred. In other words, the bid value of the server is equal to the time required for the completion of the given task. The expected completion time (C) is given by the formula:

$$C = N + Q + F \tag{3.1}$$

where,

N: time left to finish the current task
Q: total processing time of the tasks in the queue
F: processing time of the task announced

The queue time (Q) is decomposed of:

$$Q = \sum_{i=1}^{n} C_i \qquad (3.2)$$

where,

C_i: Expected completion time of the i'th task in the queue

n: Number of task in the queue

According to the algorithm mentioned, the bid value in response to a task announcement would be the sum of the current, future, and expected tasks. Let's demonstrate the preparation of the bid value by the following example:

Assume that there are two robots, which are both able to perform a specific task with the *Task_ID*=30150. The *Robot_Agent_1* and *Robot_Agent_2* at time t=0 receive a task announcement for task with the *Task_ID*=30150, at that instance the Robot_1 is executing the task with the *Task_ID*=30147, whose *Task_Start_Time*= -40. Also it has two more tasks in his *Task_List* queue with the *Task_ID*=30148 and *Task_ID*=30149. *Robot_2* is idle, and its task list queue is empty. The tasks are their database entries are given in Table 3.1.

Table 3.1 Sample database entry of the *Task_List* table

Task_ID	**Task_Description**	**Estimated_Time**
30150	Load part from the Buffer to the WS1	50
30147	Load part from the AGV to Buffer	62
30148	Load part from the WS2 to the Buffer	48
30149	Load part from Buffer to AGV	105

In this case the bid value of the *Robot_Agent_1* would be:

$$C_1 = (62 - 40) + (48 + 105) + 50 = 225$$

The bid value of the *Robot_Agent_2* is simply the expected processing time of the given task announcement:

$$C_2 = 50$$

As a result, at the given situation *Robot_Agent_2* will receive the *Task_Offer* since its bid value (expected completion time of the task announced) is smaller.

3.4 Data Model

One of the objectives of "Agent Version 1.1" system design is to localize data storage thus by distributing the information increase the robustness of the system. Based on the current situation in manufacturing enterprises two main features of database technology may be distinguished to overcome the integration problem; integrated data and process modeling, and distributed data management [48]. The following data model is based on the IDEF1X modeling technique.

Even the system is of distributed nature; it is almost inevitable to store global data in a centralized location, in a relational database. Relational databases have gained wide acceptance especially in the area of business and administration applications since they fulfill the demand of managing large amounts of simple-structured and equal-structured data and providing some intuitive querying and manipulation language. DNA provides to access almost any kind of data resource, located anywhere, and to manage information structure according to the physical needs of the system. One of the most commonly accepted data tools of DNA family is the SQL Server which is a powerful, multi-user, relational database management system designed to support high-volume transaction processing, as well as less demanding decision-support applications. SQL Server provides data processing capabilities (reliability, data integrity, performance, and security) that meet or exceed those found in production-oriented minicomputer- and/or mainframe-based database management systems. Because of its power and

stability, main “Agent Version 1.1” database has been constructed in SQL Server 7.0. Although the METUCIM system is physically a pilot shop floor consisting of one cell and a limited number of devices (agents), there is the need of thinking the data model for a generic system comprising the needs of a large-scale factory.

The IDEF1X model is used to define “entities” and their “relationships”. The concepts of this generalization and aggregation have to be extended by cardinality constraints, i.e. the amount of relationships a single object may participate in, and by constraints on the object's existence. Objects may be shared by several objects or they may belong exclusively to one object. They may be independent or dependent on the existence of other objects. To support the consistent querying and manipulation of complex objects and their relationships, cascading operations are necessary. Cascading operations ensure, for example, the retrieval of dependent component objects if the corresponding complex object is accessed. A summary of identifying relationships is given in Table 3.2.

Table 3.2 Identifying relationships in IDEF1X notation

Relationship Type	One to One	One to Many	Many to Many
Representation	has A —— B one A has one B	is customer of A ——● B one A is customer of many B	is server of A ●—● B many A is server of many B

The database model of the “Agent Version 1.1” system using IDEF1X notation is given in Figure 3.9. It may be observed that the relational structure of the database also represents the inheritance and hierarchy between entities. From the figure one can inform that one part defined with a *Part_ID* may have one or

many *Process_Plan*. A *Process_Plan* consists of one or many *Operations*, which are associated with one or more *WS_Agent_Operations*. Each entry in *WS_Agent_Operations* represents a physical connection of an operation to one or many workstation, thereby defines alternative process plans and builds the background of the bidding process, which is simply the self-selection between agents.

Part, *Process_Plan* and *Operations* entities store relatively long-term data of the parts manufactured in a typical job-shop environment. They also give a way to visualization with the *Drawing_File_Name* entry of the part and operations. Process and operation time predictions of the part, and device agents are based on the data stored in the *Estimated_Process_Time* entry of the *Operations*. In this structure, the operation is defined as the task that is accomplished by a *WS_Agent* only.

Each physical device, namely the *Device_Agent* in the environment occurs as a table in the corresponding *WS_Agent*, *Robot_Agent*, *AGV_Agent*, *PLRD_Agent* or *Buffer_Agent* tables of the database. The duplets in the *WS_AGV, WS_Robot, Robot_PLRD and Robot_Buffer* tables are used to define the customer-server relationship of the corresponding agents.

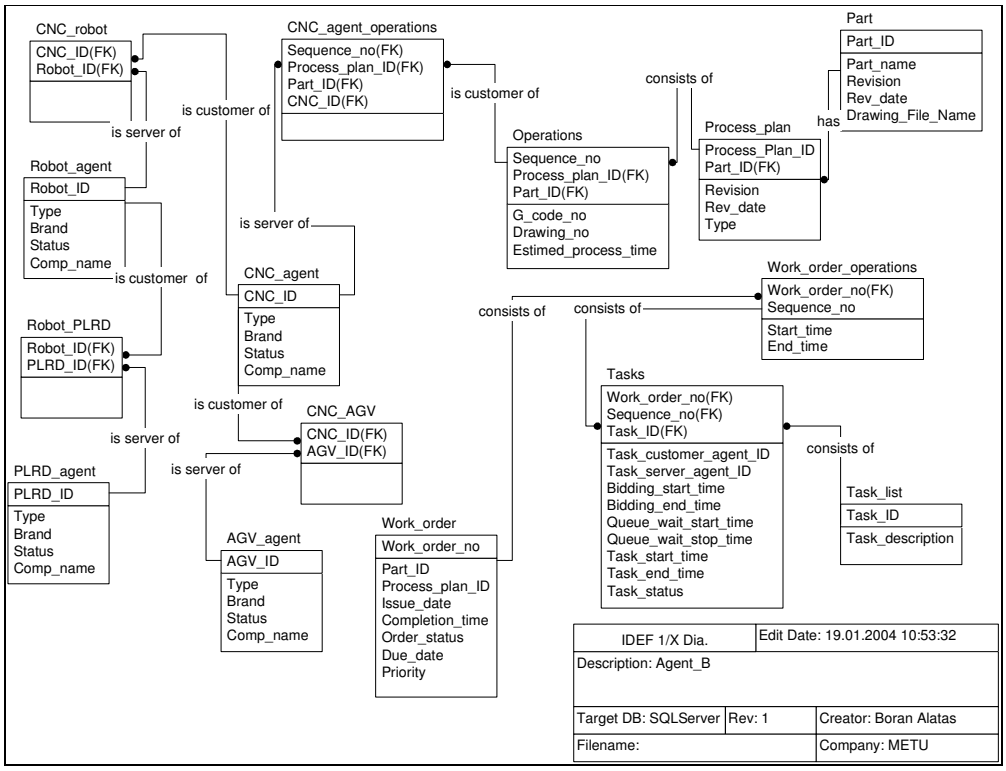

Figure 3.9 IDEF1X Data Structure [49]

The general operational information and priority of the part is kept in the *Work_Order* table. *Work_Order_Operations* and *Tasks* tables includes the details of operations. A work order is the order given by the manufacturing engineer to produce a part. The manufacturing sequence of the part with the selected process plan consists of *Work_Order_Operations,* which is then composed of generic *Tasks s*elected from the *Task_List* table. Note that tasks are the actual working units of the operation, an operation may be composed of many manufacturing, transportation, and part agent tasks. Tasks are dynamic components of an active work-order. They are very frequently added and updated following an event for task status update in *Task_Start_Time, Task_End_Time*; queue status in *Queue_Wait_Start_Time, Queue_Wait_End_Time*; bidding status in *Biding_Start_Time, Bidding_End_Time*; and the Idle/Busy status in *Own_Task_Start_Time, Own_Task_End_Time*. These information is then used for

statistical data about the cell devices, work orders, part transport/manufacturing times, lateness/tardiness values and utilization rates.

The *Config_String* property is related with the controlling mechanism of the device. For example, if the agent is controlled via serial port, the standard communication properties such as Baud Rate, Data Bit, Parity Bid, and Handshaking should be included in the *Config_String*. Device Agent tables, namely *WS_Agent*, *Robot_Agent*, *AGV_Agent*, *PLRD_Agent*, *Buffer_Agent* and duplets *WS_AGV, WS_Robot, Robot_PLRD and Robot_Buffer* contains data related with the physical and relational structure of the shop-floor, which changes only when there is a new device installed. However for a running system, *Status* property is highly dynamic. It contains the current operational status of each device and is of primary importance especially in the bidding mechanism.

3.4.1 Objects and Inheritance

When a class is derived based on another existing class, the existing methods and properties are referenced for reuse with the new class. A subclass uses all of the methods and properties defined in the super-classes above it within the class hierarchy. Inheritance can be broken for individual methods and properties when creating a subclass. Inheriting sub-classes gives the designer programming flexibility to use some object properties, methods, and events without rewriting the code. The nature of an agent-based system needs object-oriented approach plus a hierarchical object design to exploit the reusability and flexibility of the system for further modifications. It can be observed that *Part_Agent,* and *Device_Agents (WS_Agent, AGV_Agent, Robot_Agent, PLRD_Agent,* and *Buffer_Agent*) are all inherited from the abstract class *Agent*. So they have common properties such as *Agent_ID, State, Substate* and methods *Create_Queues, Delete_Queues, Close_Queue* etc. The base object *Agent i*s mainly responsible from the messaging and is susceptible to external events through the incoming messages. The *Part_Agent* is almost identical to the

Device_Agents, with a few exceptions. *Part_Agent* is an MTS DLL component, it is stateless, so it has no external properties to retrieve.

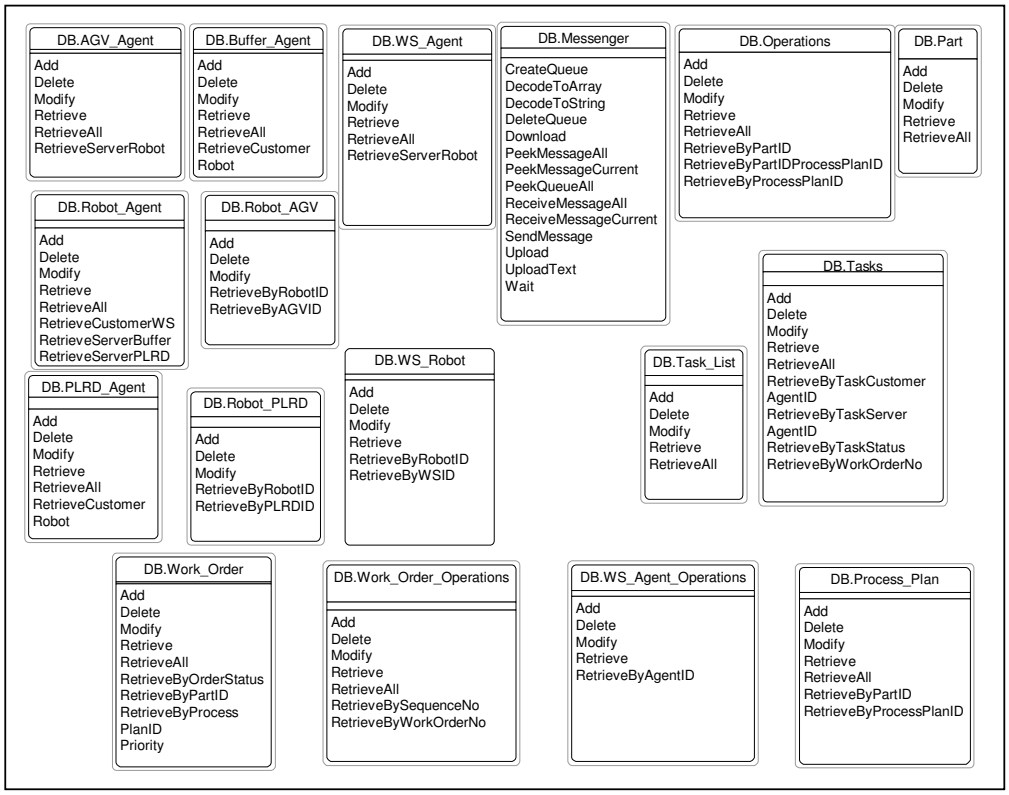

Figure 3.10 DB Objects and Messenger

The similarity between the data model and *DB_Object* is obvious. For each table in the database there is a corresponding object in the collection of database objects (Figure 3.10). The main functions are addition, deletion, and modification of data, search and retrieval in different forms. *DB.Messenger* object performs the basic functions related with the Microsoft Message Queue (MSMQ) services, such as creating/deleting queues, peeking, retrieving and decoding messages. Each agent has six queues, namely *Task_List, Task_Announcement, Bid, Task_Commitment, Task_Offer, and Task_Status*. Message parameter is a

string text, composed of items separated by semicolons. The parameters used in message formats of these queues are given in Table 3.3.

Table 3.3 Queues and message parameters

Queue	**Parameters**
Task_List	Customer_Agent_ID, Work_Order_no, Sequence_No, Task_ID,Sent_Time
Task_ Announcement	Customer_Agent_ID, Work_Order_No, Sequence_No, Task_ID,Sent_Time
Bid	Customer_Agent_ID, Work_Order_no, Sequence_No, Task_ID, Bid_Value
Task_ Commitment	Customer_Agent_ID, Work_Order_no, Sequence_No, Task_ID, Result {Accept, Reject}
Task_Offer	Customer_Agent_ID, Work_Order_no, Sequence_No, Task_ID, Result {Accept, Reject}
Task_Status	Customer_Agent_ID, Work_Order_no, Sequence_no, Task_ID, Status {0:Not Used, 1:Task_Started, 2:Task_Completed}

According to the contract-net bidding protocol, a customer, which requires a service to be completed, announces a task at the *Task_Announcement* queue of the server agents and waits for the bids to arrive at its *Bid* queue. After collection of bids, it decides on the best candidate and sends a *Task_Offer* to the selected agent. If the selected agent is ready to receive the task it sends its acceptance message to the customers *Task_Commitment* queue. If the result is “Accept” the customer sends the task to the receivers' *Task_List* queue and waits for the finish at its *Task_Status* queue.

3.4.2 Messaging

Asynchronous and synchronous messaging types can be used between applications. The asynchronous communication is like sending e-mail to a client. The targets for messages are queues, not a specific application, hence no dedicated session has to be established between the sending application and receiving application. Furthermore, both applications do not need to be running, or even connected, at the same time. On the receiving end, any number of applications may read messages from a given queue, whenever it is convenient. And the synchronous communication is creation of objects over the network, also called as Remote Procedure Call (RPC). The RPC tools make it appear to users as though a client directly calls a procedure located in a remote server program. The client and server each have their own address spaces; that is, each has its own memory resource that is allocated to data used by the procedure.

The basic deficiency of the synchronous messaging method is the overload of network traffic and overhead agents' processing capabilities caused by locks of synchronized calls. The first method provides the mechanism to preserve the command even if the receiver is not connected or is not running at all. However, the amount of asynchronous messages may saturate the receiver application event notifications.

The behavioral model of the agents is based on asynchronous sub-contractor approach. Each task is decomposed and forwarded to a server by the customer agent whenever there is a need of pre- or after-task. The states of the agent represent actually the state of their queues. When an MSMQ Queue is opened, any existing or incoming message will fire the related event in the *Agent* class. In any asynchronous communication model the use of states is necessary. For example, a *WS_Agent* cannot physically process two manufacturing tasks at the same time; whereas it may be an instance that two tasks are dispatched successively. In that case, it should read and remove the initial one and close the queue for further events. Note that, closing a queue is not removing any messages,

only the receiver application disables that queue for events in its own scope. After manufacturing, the queue is re-opened and the event *Ev_Task_Received* will fire for the next message. A complete list of events related with the MSMQ Queues is given in Table 3.4. *Ev_Own_Task_Completed* is not an external event but fired internally indicating the end of the own task

Table 3.4 Agent Events

Event	**Name**	**Mapped Sub**	**at**
Ev_1	Ev_Task_Received	ETask_Received	Agent
Ev_2	Ev_Bid_Received	EBid_Received	Agent
Ev_3	Ev_Task_Commitment	ETask_Commitment_Received	Agent
Ev_4	Ev_Service_Task_Completed	ETask_Status_Received	Agent
Ev_5	Ev_Own_Task_Completed	Own_Task_Completed (Internal)	Device/Part Agent
Ev_6	Ev_Task_Announced	ETask_Announcement_-Received	Agent
Ev_7	Ev_Task_Offered	ETask_Offer_Received	Agent

Agents' state is divided into two groups, *State* and *SubState*. The *State* is basically related with the operational status of the agent, whereas the *SubState* relates to the bidding mechanism.

- **St_Idle:** The agent whether pre-, after-, or own task has nothing to do. So its *Task_Commitment, Task_Status, Bid* are empty and closed.
- **St_Waiting_Pre_Task:** The agent has received a task, which requires a pre-task to be accomplished by a server agent. The bidding process of the pre-task has started.

- **St_Waiting_Own_Task:** All pre tasks are finished, the own processing (machining, transportation) of the agent has started.
- **St_Waiting_After_Task:** All pre tasks, and own task are finished, the bidding process of the after-task has started.

In any state or substate the *Task_Announcement* and *Task_Offer* queues are always open, since they do not require a physical action for completion. Arrival of an *Task_Announcement* requires bid preparation, arrival of an *Task_Offer* requires an "Accept/Reject" message to be sent to the customer. Both are computed instantaneously, no state change is required.

There are four main *Substates*:

- **Ss_Idle:** Either finished, or not completed there is no negotiation to do. Similar to *St_Idle*, *Task_Commitment, Task_Status, Bid* are empty and closed.
- **Ss_Waiting_Bid:** Task announcements are made to the server agents, either no bid is received or the number of bids received is not equal to the number of tasks announced or the pre-defined timeout value is not reached.
- **Ss_Waiting_Task_Commitment:** The server agent is selected and the task is offered, either the "Accept/Reject" acknowledgement is not received or the pre-defined timeout value is not reached.
- **Ss_Waiting_Service_Task:** Task is sent to the *Task_List* queue of the server, Either pre- or after- the service task is not yet finished.

Initially the state and sub-state are *St_Idle* and *Ss_Idle* respectively. The actual workflow begins with the arrival of the task at the task list queue. The task is added to the database's *Tasks* table, *Queue_Wait_Start_Time, Queue_Wait_End_Time, Task_Start_Time* entries are updated. *ETask_Received* function evaluates and judges whether a pre-task is required. If so the agent

becomes the state *St_Waiting_Pre_Task* and calls the function *Select_Service_Task*. This function is responsible for making task announcements to all servers for the specified task. When the first announcement is made, the *Bidding_Start_Time* is updated on the database and the state *Ss_Waiting_Bids* is resumed. If all bids are received, *EBid_Received* function selects the server agent with the lowest bid value (shortest expected processing time), sends a task offer, updates the database's *Bidding_End_Time* entry, and resumes the sub-state *Ss_Waiting_Task_Commitment*. Following the arrival of the task commitment message *ETask_Commitment_Received* function evaluates whether to commit or abort the task. If the result is to commit, it sends the required pre-task to the task list queue of the server and waits for the task status at *Ss_Waiting_Service_Task* sub-state, otherwise the customer re-announces the service task and waits for the bids at *Ss_Waiting_Bids* sub-state. *Ev_Task_Status_Received* event either will receive a task finished message or an error. If error, then it waits until the error is resolved and the service task is finished. A task finished message will be evaluated at *ETask_Status_Arrived* function and checked whether all service tasks are finished, if so, then the own task of the customer agent will be executed. Note that the event Ev_*Own_Task_Finished* is not related with any incoming messages, it is fired internally following an I/O state change related with the communication of the device drivers. Start and end of the own task is recorded at the database's *Own_Task_Start_Time* and *Own_Task_Finish_Time*. *EOwn_Task_Finished* function is called after the own tasks' finish, it simply checks whether an after-task is required, if so, an identical bidding procedure as in the pre-task will proceed, if not, that means the current task is finished. The agent updates its *Task_End_Time* and *Status* at the database resume idle state and re-checks its task list queue. Note that the events *Ev_Task_Announced* and *Ev_Task_Offered* do not correspond to any state change, they are answered instantaneously.

The sub-contraction model between agents is given in Figure 3.11. The part agent initially negotiates with the workstations, and sends the loading task of the selected cell to the AGV. Robot serves to the workstation and AGV, whereas Buffer and PLRD are servers of the robot. For the unloading task from the cell

again the part agent negotiates with the AGV, which is the customer to the robot and its servers.

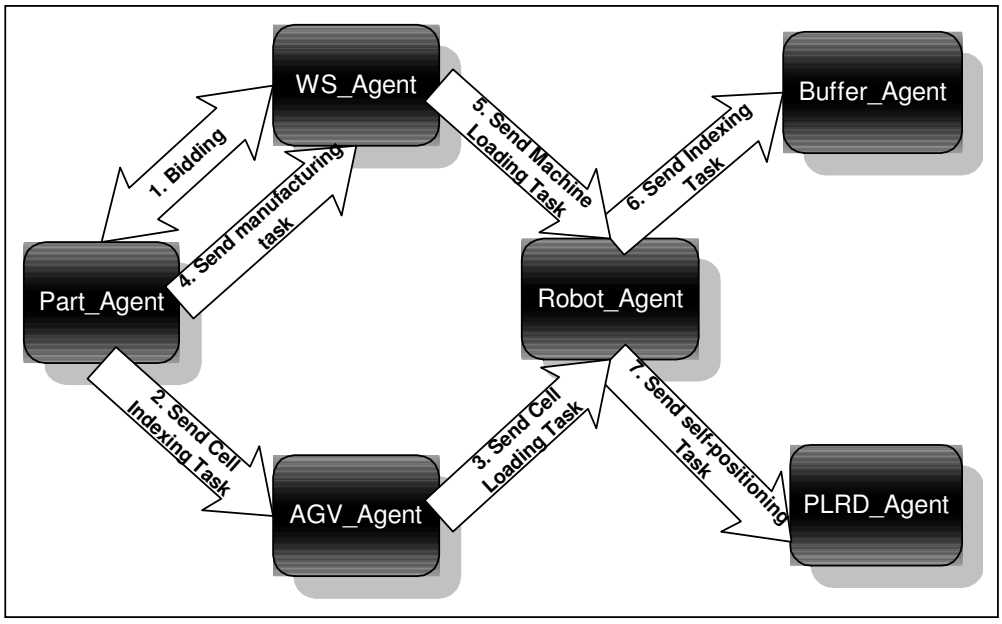

Figure 3.11 Sub-contraction model [50]

CHAPTER 4

SYSTEM DEVELOPMENT

This chapter aims to explain the work done in building the software components, extensions made to the present hardware, integration of the hardware to the software, development of new control mechanisms for the equipment, and to give a description on how to use and improve the existing package.

Details of DNA based three-tiered architecture in "Agent Version 1.1" system will be explained. The logical three-tiered model divides an application into three components:

- **Data services:** These services join records and maintain database integrity—for example, constraints on valid values for a work order number and an enforced foreign-key relationship between the *Work_Order* table and the *Work_Order_Operations* table.
- **Business services:** These services apply business rules and logic—for example, adding a new task to the *Work_Order* table and updating the *Work_Order_Status*.
- **Presentation services:** These services establish the user interface and handle user input—for example, code to display the part number and part drawing for a selected work order.

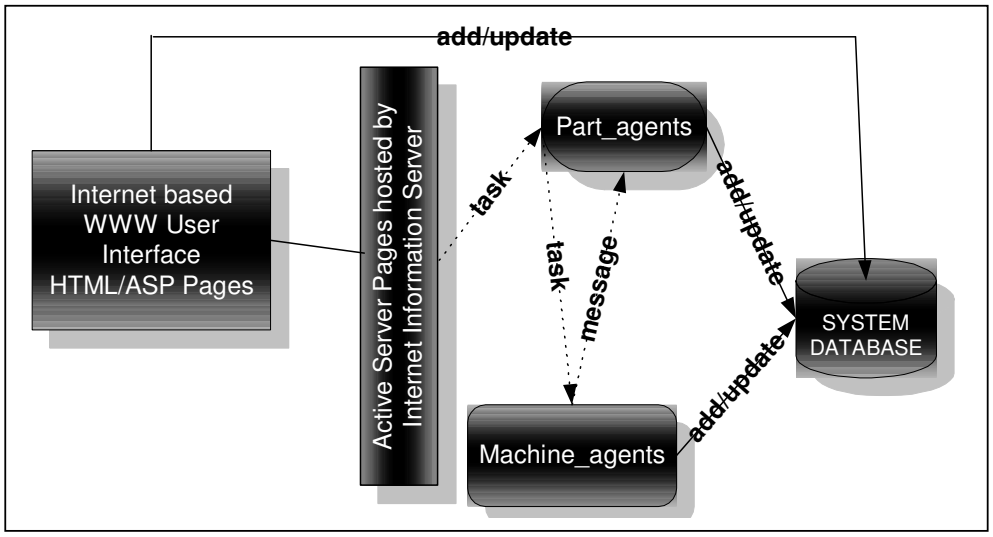

Figure 4.1 Communication between system components

In this development process the work is categorized using the well-known three-tiered model of computing in system components (business services), database and messaging (data services), and user interface (presentation). The overall relationship between system components is shown in Figure 4.1. Note that besides HTML/ASP pages, which are reachable from anywhere in the world.

4.1 Business Services

4.1.1 Agent Base Class

The core component of the agent system as its name reveals is the "Agent.cls". Device or Part agent classes inherit the base class agent. The agent class is responsible from firing MSMQ related events, opening/closing queues, and sending messages to other agents. The events of the base class Agent is given in Table 4.1.

Table 4.1 Events of the Agent base class

Event Name	**Queue Object**
ETaskAnnouncementReceived	*Task_Announcement*
EBidReceived	*Bid*
ETaskOfferReceived	*Task_Offer*
ETaskCommitmentReceived	*Task_Commitment*
ETaskReceived	*Task_List*
ETaskStatusReceived	*Task_Status*

An object is generally composed of properties, methods and events. Methods and properties of an object is classified into two major categories:

- **Public:** The method or property can be seen from outside the module.
- **Private:** The method or property cannot be seen from outside, but the internal functions can use it.

4.1.2 OCX Objects

The user controls (OCXs) are actually standard development tools of Visual languages; they have their own properties, methods, and events as well as their User-Interface. A standard user control example is the textbox, which is almost common in any development tool. The textbox is actually some piece of existing code with the programming language built-in as a User-control.

In this model, the objects for digital I/O and serial communications related with the control of the device are built as OCXs. The idea behind building the controller mechanisms of the Device Agents as User Controls, is that they can also be used later as individual tools. That is, if later in another model, the messaging procedures or the interfaces of the Device Agents should be completely

different, the programmer has still an object, which utilizes the driving mechanism the machine.

Each device agent inherits also its related OCX object. The OCX object is a User Control for digital I/O and serial communications related with the control of the device. For the control of the "Agent Version 1.1" system 6 OCX user controls have been implemented:

4.1.3 Device Agent

Any device agent inherits the base class for its MSMQ messaging and its OCX object for machine control. Note that, the functions of *Device_Agent* are alone capable of controlling the individual machines but the inherited class *Agent* is actually "the eyes of the device" open to the others. The functions of the Automated Guided Vehicle (AGV) device agent (*AGV_Agent_1*) are summarized in Figure 4.2 Also the other device agents share the identical methods but different control mechanisms in terms of serial or I/O communications.

Figure 4.2 Visual Modeler view of the *AGV_Agent* [40]

The user interface of the device agent is a tabbed form, and consists of: Properties, Events, Queues, Task, and Config. The task tab can also be used to dispatch individual tasks to the CNC Agent with a given or retrieved task parameter.

4.2 Data Services

4.2.1 Database

The database of the "Agent Version 1.1" system is built using SQL Server 7.0. SQL Server is a product in Windows DNA family, which helps construction of SQL Server relational databases in a systematic and user-friendly environment. The Structured Query Language (SQL) is a language used in querying, updating, and managing relational databases. SQL can be used to retrieve, sort, and filter specific data to be extracted from the database. SQL statements can be categorized into ALTER, CREATE, DELETE, INSERT, SELECT and UPDATE statements. These are used to change the structure of a table in the database, to create a new table, to delete records from a table, to add records to a table, to perform a search query from a database, and to change same values in the database respectively. For example to retrieve all part data for the part with the *Part_ID*=30001 from the *Parts* table the following SQL statement is required:

```
SELECT * FROM Parts WHERE Part_ID=30150
```

Also parameters may be passed to a query to perform searches according to user inputs. One can define a stored procedure to pass *Part_ID* as a parameter for the previous case. The stored procedure can be called by passing a value to *@prmPart_ID* parameter:

```
CREATE Procedure spRPart
@prmPart_ID as int
AS
SELECT *
FROM Part
WHERE (Part_ID=@prmPart_ID)
RETURN
```

A comprehensive model of the constructed “Agent Version 1.1” database had been explained in the previous chapter. An SQL Server database is a complete unit with its tables, primary keys, relationships, users, views and stored procedures. The design view of the database in SQL Server 7.0 is shown in Figure 4.3.

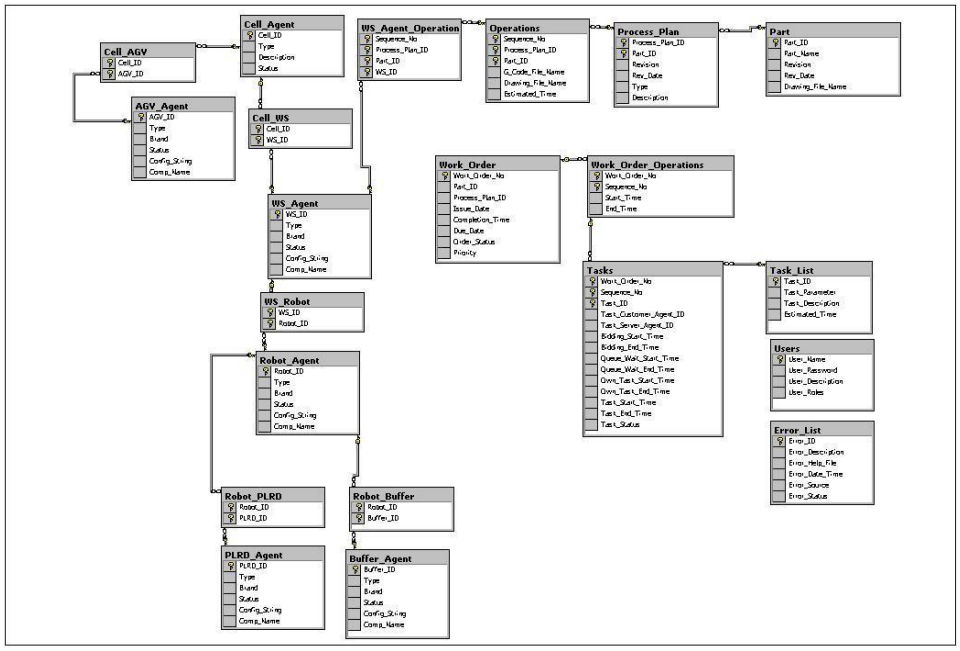

Figure 4.3 Design view of the SQL Server database

4.2.2 DB Objects

Database objects are used to add, delete, modify, and retrieve data from the SQL Server database. The objects are created and destroyed almost immediately once they complete their action. For every table in the database there is a corresponding *DB* object. They use generally the stored procedures in order to query data. The functions of the *DB.Part* object are given in Figure 4.4.

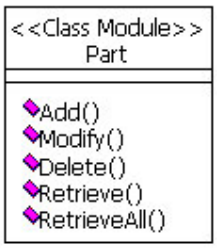

Figure 4.4 Methods of the *DB.Part* Object

DB objects are Microsoft Transactions Server components. An overview of the main MTS screen is shown in Figure 4.5.

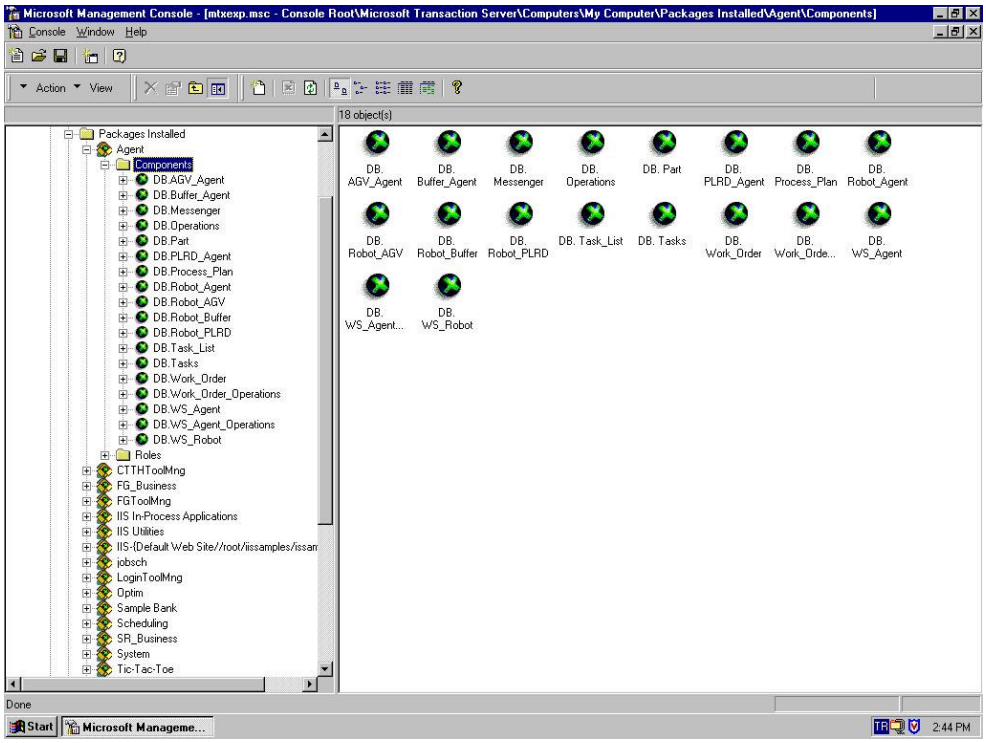

Figure 4.5 MTS Components

4.3 Presentation Services

The web-based interface of the system is implemented using the ASP programming model. VB-Script is used throughout the pages. Visual tools such as buttons, figures, combo- and textboxes are applied for a user-friendly interface.

The “Agent Version 1.1” is an actual manufacturing system to control the shop floor devices, retrieve information about parts, operations, and statistics, to form layout and relationships. People may have different roles in a society. In that manner the “Agent Version 1.1” system also represents a manufacturing company in which there are Administrator, Engineer, Operator, and User roles. The users are prompted to indicate a valid username and password to enter the web-site. By that, the identity of the current person is detected and stored, also unattended entries to the system has been prohibited. The “Login Page” is shown in Figure 4.6.

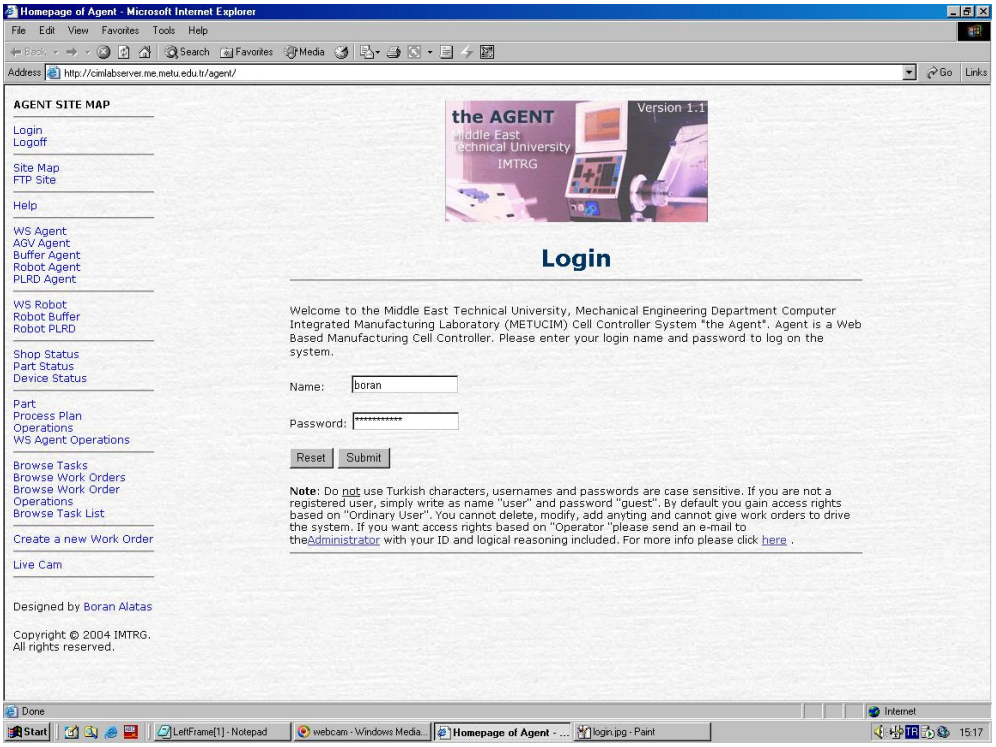

Figure 4.6 Login to the system

User rights are based on:

1. **Administrator:** Represents users that can fully administer all information contained in the database.
2. **Engineer:** Represents users that can add, remove, delete parts, process plans, operations, give work orders but cannot change the shop floor structure.
3. **Operator:** Represents users that cannot change part information or system structure but they are allowed give work orders.
4. **User:** Represents users that cannot change any database entry, cannot give any work orders but can browse all information.

The complete web site root is shown in Figure 4.7. The pages can be classified into six categories:

1. *AGV_Agent, WS_Agent, PLRD_Agent, Buffer_Agent, Robot_Agent* relate to the devices available on the shop floor. *WS_Robot, Robot_PLRD, Robot_AGV* indicate the connections between the indicated agents. The *Task_List* page contains information about the generic tasks. Data in these pages can be altered by Administrators only.
2. *Part, Process_Plan, Operations, WS_Agent_Operations* store the static part information. Data in these pages can be altered by Administrators and Engineers only.
3. *Work_Order_Create* is the starting point in execution of a work order (manufacturing a part). Administrators, Engineers and Operators are allowed to give work orders.
4. *Work_Order_Browse, Work_Order_Operations_Browse, Tasks_ Browse, Shop-, Device-, and Part_Status* pages are used to access to all kind of online and statistical information. Everybody can browse the data contained in these pages.
5. *Help, Login, Logout, Root* are auxiliary pages to direct the user and give information.

6. *Live_Cam* runs Windows Media Player and connects to the web-cam in METUCIM. The user can watch all the activities live. A screen shot of *Live_Cam* link is shown in Figure 4.8.

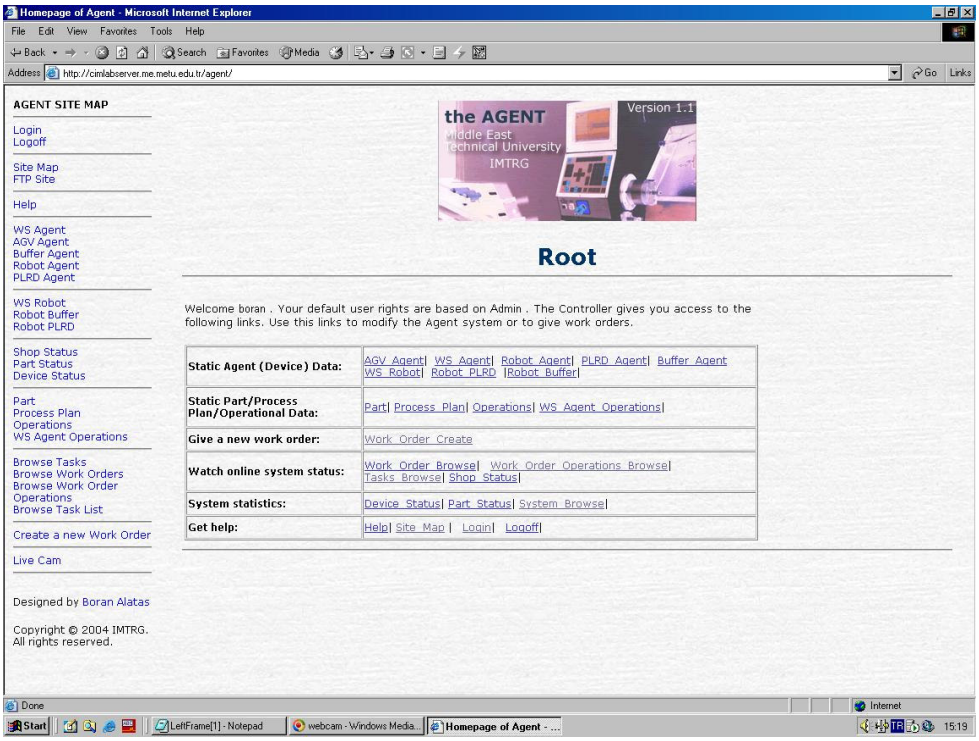

Figure 4.7 Root page of the web site

The web is published on Internet Information Server site at the address: http://cimlabserver.me.metu.edu.tr/Agent Information about "how to use the system" can be found at http://cimlabserver.me.metu.edu.tr/Agent/Help.htm. Integrated Manufacturing technologies homepage can be reached from http://www.imtrg.me.metu.edu.tr. A sample screen of Agent Explorer's "Work Order Create" is given in Figure 4.9.

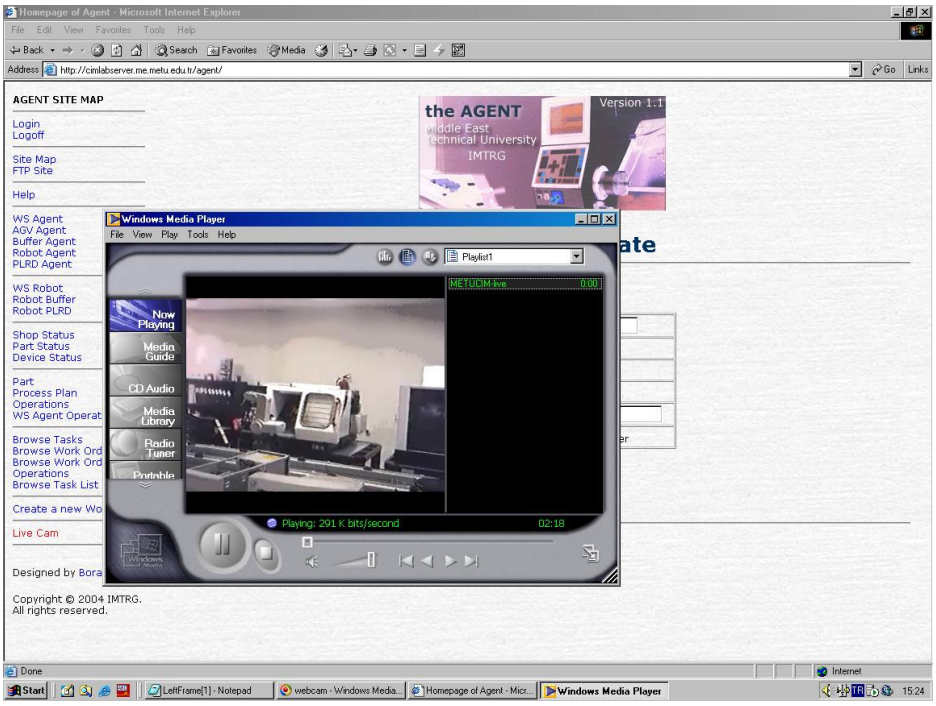

Figure 4.8 Live Cam screen

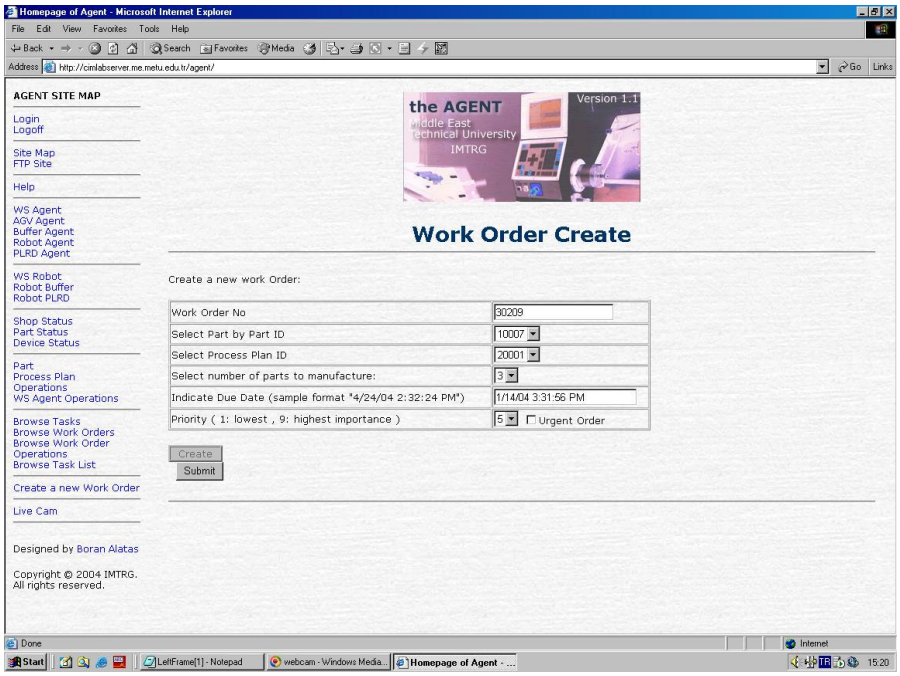

Figure 4.9 Work Order Create screen

CHAPTER 5

TEST RUNS

5.1 First Test Run ("Priority" and "Urgent Order" Application)

In this test run the "Urgent Order" application is observed. First test run is made with five parts with the part IDs 10001, 10002, 10003, 10004 and 10005 respectively; they have the corresponding *Work_Order_No*s 30198, 30199, 30200,30201 and 30202. The operations for all work orders are given in Table 5.1. Photographs of the parts are given in Figures 5.1, 5.2, 5.3, 5.4 and 5.5 respectively. G-Codes of the operations in the first test-run are given in Appendix B, Figures B.1 to B.10. The detailed manufacturing scenario is explained in this section.

In a job shop, the manufacturing engineer had taken an order for the part 30198, he decided to give a priority value "3" for that order because he thought it has very low urgency. The customer wanted the part in a month. He gave the manufacturing order of that part using the "Agent Version 1.1". Now the rough part 30198 is waiting for manufacturing. According to its process plan the first operation is turning, when the work order is given the lathe was idle and it has began to manufacture. The turning operation is long for that part and the company has taken the orders very fast.

A new work order had taken at that time for the part 30199, the manufacturing engineer had decided to give a priority value "5" because the customer wanted the part in a week. It has added to the manufacturing queue and loaded to the conveyor waiting for the turning operation.

While the part 30198 was still in lathe, the orders for the part 30200 had taken. The customer wanted the part in a week. The engineer thought that this customer always gives orders and always work with them. So decided to manufacture the part 30200 earlier then 30199. He gave priority value “7”. It has loaded to the conveyor.

A new work order has taken. The customer wanted the part in three days. The priority value “8” is given for this new order 30201. The system has still waiting for the part 30198. The part 30201 has loaded to the conveyor.

Lastly a work order 30202 has taken and the customer said that this part must be produced as soon as possible, it has urgency. The engineer decided to manufacture this part first so using the software “Agent Version 1.1”, he clicked on the “Urgent Order” checkbox in Work Order Create page. While the part 30202 has carrying by AGV and loading to the conveyor the lathe become idle. The part that has the highest priority except the urgent one began to machine. The priorities and manufacturing urgencies of the parts are same for all machine tools. The manufacturing has finished in a sequence 30198, 30201, 30202, 30200 and 30199 respectively. The details are given that given in Table 5.2.

It is obvious that the time values are not simulate this scenario because the machining times are short and the work order giving procedure is faster. But the values give an idea for manufacturing sequence.

Table 5.1 First test run work order details

Work Order No	Part ID	Part_Name	Process Plan ID	Opera-tions	Alter-native Agent IDs
30198	10001	Sample_Part_1	20001	Turning, Milling	1001,1004 1002
30199	10002	Sample_Part_2	20001	Turning, Milling	1001,1004 1002
30200	10003	Sample_Part_2	20001	Turning, Milling	1001,1004 1002
30201	10004	Sample_Part_2	20001	Turning, Milling	1001,1004 1002
30202	10005	Sample_Part_3	20001	Turning, Milling	1001,1004 1002

Figure 5.1 Part 10001

Figure 5.2 Part 10002

Figure 5.3 Part 10003

Figure 5.4 Part 10004

Figure 5.5 Part 10005

The statistics of the completed work orders, the priorities and its operations are given in Table 5.2 and Table 5.3. The data has been deduced from the *Work_Order_Browse* and *Work_Order_Operations_Browse* screen by selecting the work orders 30198, 30199, 30200, 30201 and 30202.

Table 5.2 Work orders statistics of the first test run

Work Order No	**30198**	**30199**	**30200**	**30201**	**30202**
Issue Date	3:07:42 PM	3:08:17 PM	3:08:49 PM	3:09:02 PM	3:10:39 PM
Completion Time	3:30:38 PM	3:38:46 PM	3:38:04 PM	3:35:53 PM	3:32:50 PM
Priority	3	5	7	8	Urgent

Table 5.3 Work order operations statistics of the first test run

Work Order No	**30198**	**30199**	**30200**	**30201**	**30202**
Op.1 Start Time	3:07:51 PM	3:08:27 PM	3:08:59 PM	3:09:12 PM	3:10:48 PM
Op.1 End Time	3:11:47 PM	3:28:02 PM	3:24:26 PM	3:16:13 PM	3:20:26 PM
Op.2 Start Time	3:11:48 PM	3:28:03 PM	3:24:26 PM	3:16:14 PM	3:20:26 PM
Op.2 End Time	3:30:37 PM	3:38:45 PM	3:38:04 PM	3:35:53 PM	3:32:52 PM

Any operation is composed of several tasks, which are accomplished by individual agents. These are classified in:

- **Part Agent Task:** The typical part agent task, which is dispatched from the part coordinator, is to manufacture the part itself in a selected WS. Logically, an operation will contain a single part agent task.
- **Manufacturing Task:** Manufacturing of the part in the selected workstation.
- **Transport Task:** Any transportation task accomplished by an AGV, Buffer, PLRD, Robot.

Task statistics of the first test run work orders are given in Table 5.4. For information about the devices please refer to Appendix.

Table 5.4 Task statistics of the first test run

Work Order No	**30198**	**30199**	**30200**	**30201**	**30202**
Total Number of Tasks	21	21	21	21	21
Number of Manufacturing Tasks	2	2	2	2	2
Number of Part Agent Tasks	2	2	2	2	2
Number of Transport Tasks	17	17	17	17	17
Task Makespan Cmax	22m:47s	30m:18s	29m:5s	26m:41s	22m:2s
Queue Time	20m:37s	43m:20s	36m:32s	25m:37s	23m:51s
Bidding Time	1s	2s	1s	1s	2s
Transport Time	4m:12s	4m:44s	6m:12s	6m:16s	5m:54s
Manufacturing Time	4m:08s	3m:45s	4m:35s	3m:23s	3m:32s
Idle Time	18m:0s	24m:59s	22m:18s	19m:50s	15m:33s

Device statistics and utilization rates are given in Table 5.5

Table 5.5 Device statistics of the first test run

Device ID	**1001**	**1002**	**2001**	**3001**	**4001**	**5001**
Number of Tasks	5	5	30	10	15	30
Utilization Time	7m:12s	6m:23s	15m:50s	7s	35s	10m:46s
Makespan	16m:1s	16m:13s	30m:48s	30m:50s	30m:12s	30m:16s
Device Utilisation in spec. Interval	11%	8.8%	15%	0.1%	0.9%	10.5%
Search Interval: 3:07:00 PM-3:39:00 PM						

5.2 Second Test Run ("Manufacturing in Batches" Application)

The second test run is made with a batch with three parts. The part IDs are 10002 for all parts , they have the corresponding *Work_Order_No*s 30203, 30204, and 30205. Part- and work order information is given in Table 5.6. Part photograph of 10002 is given in Figure 5.2. G-Codes of the operations in the second test-run are given in Appendix B, in figures B.3 and B.4. The process plan of parts is identical to the one in the first test run.

Table 5.6 Second test run work order details

Work Order No	Part ID	Part_Name	Process Plan ID	Opera-tions	Alter-native Agent IDs
30203	10002	Sample_Part_2	20001	Turning, Milling	1001,1004 1002
30204	10002	Sample_Part_3	20001	Turning, Milling	1001,1004 1002
30205	10002	Sample_Part_4	20001	Turning, Milling	1001,1004 1002

The statistics of the completed work orders and its operations are given in Table 5.7 and Table 5.8.

Table 5.7 Work orders statistics of the second test run

Work Order No	**30203**	**30204**	**30205**
Issue Date	4:51:18 PM	4:51:18 PM	4:51:18 PM
Completion Time	4:57:27 PM	5:01:17 PM	5:03:25 PM
Due Date	4:57:30 AM	4:57:30 AM	4:57:30 AM

Table 5.8 Work order operations statistics of the second test run

Work Order No	**30203**	**30204**	**30205**
Op.1 Start Time	4:51:18 PM	4:53:38 PM	4:55:59 PM
Op.1 End Time	4:53:28 PM	4:55:48 PM	4:58:09 PM
Op.2 Start Time	4:53:28 PM	4:55:48 PM	4:58:09 PM
Op.2 End Time	4:54:18 PM	4:56:38 PM	4:59:10 PM

Task statistics of the second test run work orders are given in Table 5.9. For information about the devices please refer to Appendix.

Table 5.9 Task statistics of the second test run

Work Order No	**30203**	**30204**	**30205**
Total Number of Tasks	21	21	21
Number of Manufacturing Tasks	21	21	21
Number of Part Agent Tasks	2	2	2
Number of Transport Tasks	17	17	17
Number of Bids Received	1	1	1
Task Makespan Cmax	13m:21s	9m:31s	16m:14s
Queue Time	4m:12s	10m:42s	13m:48s
Bidding Time	2s	1s	2s
Transport Time	4m:35s	5m:02s	5m:07s
Manufacturing Time	3m:33s	2m:59s	3m:22s
Idle Time	14m:13s	10m:5s	5m:44s

Device Statistics and utilization rates are given in Table 5.10 and 5.11.

Table 5.10 Device statistics of the second test run (agents 1001 and 1002)

Device ID	**1001**	**1002**
Number of Tasks	2	2
Utilization Time	4m:31s	4m:13s
Makespan	11m:59s	12m:14s
Device Utilisation in spec. Interval	5.1%	4.8%
Search Interval: 4:53:00 PM-5:02:00 PM		

Table 5.11 Device statistics of the second test run (agents 2001 to 5001)

Device ID	**2001**	**3001**	**4001**	**5001**
Number of Tasks	16	6	8	16
Utilization Time	8m43s	5s	33s	3m:22s
Makespan	18m:54s	22m:1s	20m:08s	18m:57s
Device Utilisation in spec. Interval	9.2%	0.03%	1%	3.2%
Search Interval: 4:53:00 PM-5:02:00 PM				

In these demonstrative test runs it is observed that the actual manufacturing times come up around 8-32% of the makespan of the current work order. Note that, the time required for bidding and selection (communication between agents) is around 0-3 seconds which contributes negligibly to idle time. The majority of the idle time lost occurs in serial and I/O card communications and part program downloads, which can be enhanced by faster but increasingly unreliable settings. Also faster transport speeds of the conveyor, PLRD, and robot will result in increased effectiveness but may cause errors in positioning and material handling. Since the “Agent Version 1.1” system aims a demonstrative integrated automation, these settings are intentionally kept at a low level for increased safety.

CHAPTER 6

CONCLUSION AND FUTURE WORK

In this study, a new model for an agent-based dynamic scheduling methodology for controlling manufacturing cells developed in METUCIM. Windows DNA is used for the application of agent-based strategy and for the system reliability and robustness. Re-configurable and flexible systems have been the main concern of the manufacturing industry. The ability to react changes in the product demand and controller software technology has been the determining factor in automation. Thus, the modeling of the information and control strategy; not only for today, but also for tomorrow; has been the key success of the manufacturing engineer

By the developing industry and increased customer demands, today's manufacturing environment is very dynamic that the manufacturers must respond very quickly. The first aim of this study is making a program that can be applicable to real manufacturing systems in job shops. From this point of view "Urgent Order" application is developed. While there are many work orders in manufacturing queue, an urgent order may be taken. If the manufacturing engineer decides that it will be profitable, the job shop manufactures the urgent order as soon as possible.

Today in SMEs, the manufacturing volume is increased by the help of new technologies. In our system in METUCIM we model job shops so the developed agent based manufacturing control strategy can be adapted to a job shop. For a control system, manufacturing in batches is a must. In the new configuration, it is possible.

In the design of the software model of the "Agent Version 1.1" enterprise, agent-based communication approach is applied. This allows using the components of the software as stand-alone machines/parts as well as the elements of a complete manufacturing system. The messaging procedure involves a customer-server based negotiation mechanism in which the external input is given by the user (manufacturing engineer) from a locally restricted Internet web site, thus enhancing browsing and monitoring capabilities of online data and status. The agent philosophy itself is an open architecture to implementation and selection between alternatives. There is no preexisting hierarchy of some agents to others; the customer-server relationship is adopted for the communication. A customer is simply the agent giving some task to other, where a server is the one who accepts the task. The heterarchy dictates that all are at the same level; that is, at some time a server machine/part may be the customer to other machines/parts. The messaging procedure continues until the acknowledgement responses from all server agents are reached, meaning that all required pre- and after tasks are finished. Agent based communication is among the newest approaches, supporting "stand-alone" machines to behave as individuals, with own judgement, selection, scheduling, and machine loading features. Each individual is visible to its own, but also has a message based relationship with the other, connected to a main information system.

With the trend toward distributed computing in enterprise environments, it is important to have flexible and reliable communication among agents. Distributed Internet Applications (DNA) architecture is composed of independent applications that are running on different systems to communicate with each other and exchange messages even though the applications may not be running at the same time. Message Queue Server technology enables these applications or simply COM objects running at different times to communicate across heterogeneous networks and systems that may be temporarily offline. Within an enterprise, applications send messages to queues and read messages from queues.

The machine/part agent is programmed to meet its objectives based on the available data on the main "Agent" database.

This research on Computer Integrated Manufacturing is mainly focused on the implementation of a flexible, re-configurable manufacturing and information system. It focuses on realizing an Agent based manufacturing cell control system with DNA technology by the use of the hardware in Middle East Technical University, Mechanical Engineering Department, Computer Integrated Manufacturing Laboratory (METUCIM), Ankara, Turkey. The result is an agent system with reduced complexity in scheduling, manufacturing execution, part routing, software generation and a well-defined communication model. The modular architecture has open doors to rapid system integration and code reuse. Browsing and visualization capabilities of the dynamic interfaces will provide user satisfaction and real-time data processing. The research is not limited to pilot systems but aims to develop a full system model and flexible software components especially for Small and Medium Sized Enterprises (SMEs) in industry.

Developed software can be interpreted as "putting things in place". Its scope extends from scheduling to shop floor control with online monitoring and dispatching capabilities from the widespread Internet environment, equipped with a user-friendly interface. However, its has certain limitations and drawbacks. These can be summarized as:

1. The existing Quality Control Software on CMM Host Computer is hard to be modified. So the "quality aspects" and remote programming and control cannot be possible via integration of CMM to the system.

2. The Live Cam application must run in a Microsoft environment. Microsoft Internet Explorer must be used and also Windows Media Player must be downloaded. Live Cam, Active Server Pages (ASP) and VB-Script is not yet common in all browsers. For example,

Netscape does not allow to correctly browsing the system pages running the applications. The use of Internet Explorer 4.0 or higher versions and the use of Windows Media Player 7.0 or higher versions are expected.

3. At the shop floor level, Device Agents give indications and warnings for error recovery; the watch from the Internet environment is not currently implemented.

4. The system hardware is not fast enough. The conveyor speed is low so the machine tools wait for the part too long.

5. The programming system based on Windows NT, that an old technology operating system for servers.

The concepts of CIM are changing rapidly. Every technological improvement adds a new word to the FMS dictionary. The developed Agent based architecture and database system is itself an open system for future reconfiguration and development. By the work done in the current research, following possible future improvements can be deducted:

1. Quality control modules should be improved, for this; a quality control cell can be implemented for integrated use with the CMM. Quality Control cell can be equipped with an image-processing unit, for example a digital camera for sensing and identifying manufactured parts.

2. Error handling and recovery functions should be improved and coordinated to the use on ASP pages.

3. Tooling and tool management modules can be implemented in the database.

4. A conveyor, which has a higher linear speed, can be added to the system. That will decrease the idle time for manufacturing.

5. The Internet pages can be enhanced with Computer Aided Design (CAD) and Computer Aided Process Planning (CAPP), and G-Code Generation to form an integrated peripheral of (CAD/CAM).

6. The operating system may be upgraded for a more reliable agent communication and messaging on network.

REFERENCES

[1] Platt; A. F., 1999, White Paper Draft, "Windows DNA for Manufacturing", Microsoft Corporation

[2] Friedrich, H., Rogalla, O., Dillmann, R., 1998, "Integrating skills into multi-agent systems", Journal of Intelligent Manufacturing, Vol. 9, pp 119-127

[3] Moffatt, C., 1992, "Designing Client/Server Applications for Enterprise Database Connectivity", Microsoft Press

[4] Luggen, W. W., 1991, Flexible Manufacturing Cells and Systems, Prentice Hall, p. 19, 378

[5] Shen, W., Norrie, D.H., 1999, "Agent-Based Systems for Intelligent Manufacturing: A State-of-the-Art Survey", Knowledge and Information Systems, an International Journal, Vol.1, No.2, pp 129-156

[6] A. Quintas and P. Leitão, 1997, "A Manufacturing Cell Controller Architecture", in Proceedings of Flexible Automation and Intelligent Manufacturing Conference, Middlesbrough, pp 483-493.

[7] Paulo Leitão, Francisco Restivo, Goran Putnik, 2001, "A Multi-Agent Based Cell Controller"

[8] J. Ferber Multi-Agent Systems, 1999, "An Introduction to Distributed Artificial Intelligence", Addison-Wesley.

[9] Kim, Y. D., Yano, C. A., 1997, "Impact of throughput-based objectives and machine grouping decisions on the short-term performance of flexible manufacturing systems", International Journal of Production Research, Vol. 33, No. 12, pp. 3303-3322

[10] H. Van Dyke Parunak, 1998, "What can Agents do in Industry, and Why? An Overview of Industrially-Oriented R&D at CEC", Industrial Technology Institute.

[11] Lu, T.-P. and Yih, Y., 2001, "An agent-based production control framework for multiple-line collaborative manufacturing", International Journal of Production Research, 39(10), pp. 2155-2176

[12] Maturana, F. P. and Norrie, D.H., 1996, "Multi agent mediator architecture for distributed manufacturing", Journal of Intelligent Manufacturing, 7, pp. 257-270

[13] Krishnamoorthy, B. and Kamath, M., 1999, " A new approach to the design of FMS control architectures", Proceedings of the 8th Industrial Engineering Research Conference, Phoenix, AZ, CD-ROM.

[14] Brenner, W., Zarnekow, R. and Hartmut, W., 1998, "Intelligent Software Agents: Foundations and Applications", Springer-Verlak, New York.

[15] Rosenschein, S. J., 1999, " Intelligent agent architecture", The MIT Encyclopedia of the Cognitive Sciences, Wilson, R.A. and Keil, F. C. (eds.), The MIT Press, Cambridge, MA, pp. 411-412

[16] Lin, G. Y.-J. and Solberg, J. J., 1992 " Integrated shop floor control using autonomous agents" IIE Transactions, 24(3), pp. 57-71

[17] Hatvany, J., 1985, “Intelligence and cooperation in heterarchic manufacturing systems”, Robotics and Computer-Integrated Manufacturing, 2, pp.101-104

[18] Duffie, N. A. and Piper, R. S., 1987, “Non-hierarchical control of a flexible manufacturing cell” Robotics and Computer-Integrated Manufacturing, 3, pp. 175-179

[19] Shaw, M. J., 1988, “Dynamic scheduling in cellular manufacturing systems: A framework for networked decision making” Journal of Manufacturing Systems, 7(2), pp. 83-94

[20] 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, 3(3), pp. 229-248

[21] Sousa, P. and Ramos, C., 1999, “A distributed architecture and negotiation protocol for scheduling in manufacturing systems”, Computer in Industry, 38(2), pp. 103-113

[22] Chan, T.S. Felix, Chan, H., K. and Kazerooni, A., 2003, ”Real time fuzzy scheduling rules in FMS”, Journal of Intelligent Manufacturing, 14, pp. 341-350

[23] Baid, N. K. and Nagarur, N. N., 1994, “An integrated decision support system for FMS: Using intelligent simulation”, International Journal of Production Research 32(4), pp. 951-965

[24] Ishii, N. and Talavage, J. J., 1994, “ A mixed dispatching rule approach in FMS scheduling”, International Journal of Flexible Manufacturing Systems, 6, pp. 69-87

[25] Wen, H. J., Smith, C. H. and Minr, E. D., 1996, " Formation and dynamic routing of part families among flexible manufacturing cells", International Journal of Production Research, 34(8), pp. 2229-2245

[26] Yu, L., Shih, H. M. and Sekiguchi, T., 1999, "Fuzzy interface-based multiple criteria FMS scheduling", International Journal of Production Research, 37(4), pp. 2315-2333

[27] Shafai, R. and Bruno, P., 1999, "Workshop on scheduling using practical inaccurate date-Part2: An investigation of the robustness of scheduling rules in a dynamic and stochastic environment" International Journal of Production Research, 37, pp. 4105-4117

[28] Cowling, P. I. And Johansson, M., 2001, "Using real-time information for effective dynamic scheduling", European Journal of Operational Research, 139(2), pp. 230-244

[29] Dupon, A., Nieuwenhuyse, Van, I. and Vandaele, N., 2002, "The impact of sequence changes on product lead time", Robotics and Computer-Integrated Manufacturing, 18, pp. 327-333

[30] Joshi, S. B. and Smith, J. S., 1994 "Computer control of flexible manufacturing systems", Chapman & Hall, New York, pp. 10-12

[31] Zhang, J., Gu, J., Li, P., Duan, Z., 1999 "Object-Oriented Modeling of Control System for Agile Manufacturing Cells", International Journal of Production Economics, Vol. 62, pp.145-153

[32] Venkatesh, K. and Zhou, M., 1998, "Object-Oriented Design of FMS Control Software Based on Object Modeling Technique Diagrams and Petri Nets" , Journal of Manufacturing Systems, Vol.17, No.2, pp. 118-136

[33] Aguirre, O., Weston, R., Martin, F., and Ajuria, J. L., 1999, "MCSARCH: An Architecture for the Development of Manufacturing Control Systems", International Journal of Production Economics, Vol.62, pp. 45-59

[34] Buzacott, J. A. and Shanthikumar, J. G., 1980, "Models for Understanding Flexible Manufacturing Systems", AIIE Transactions, Vol.12, pp. 339-350

[35] Gong, D. C. and Lin, K. F., 1994, "Conceptual Design of a Shop Floor Control System from IDEF0", Computers Industry Engineering, Vol.27, pp. 119-122

[36] D'Souza, K. A. and Khator, S. K., 1994, "A Survey of Petri Net Applications in Modeling Controls for Automated Manufacturing Systems", Computers in Industry, Vol.24, pp. 5-16

[37] Haddock, J., 1995 "Automated Simulation Modeling and Analysis for Manufacturing Systems", Production Planning and Control, Vol.6, No.4, pp. 352-357

[38] Booth, A. W., 1998, "Object-Oriented Modeling for Flexible Manufacturing System", International Journal of Flexible Manufacturing Systems, Vol.10, No.3, pp. 301-314

[39] Bruccoleri, M., Perrone, G. and Noto La Diega, S., 2000, "Object Oriented Modeling for Concurrent Engineering Design", in Proceedings of the 33rd CIRP International Seminar on Manufacturing Systems, pp. 341-346

[40] Cangar, Tolga 2000, "Development of an Agent Based Flexible Manufacturing Cell Controller Using Distributed Internet Applications", Master Thesis, Graduate School of Natural and Applied Sciences, Middle East Technical University

[41] MSDN Library Visual Studio 6.0, “Transactions”, ODBC Programmers Reference, Part 2, Chapter 14

[42] Draft Federal Information, Processing Standards Publication 183, December 21, 1993, the Standard for Integration Definition for Function Modeling (IDEF0).

[43] S. E. KILIÇ (1982) Scheduling of Cutting Conditions in Multi-Pass Turning Operations, Post-Doctoral Thesis Middle East Technical University Ankara Turkey

[44] Draft Federal Information, Processing Standards Publication 184, December 21, 1993, the Standard for Integration Definition for Information Modeling (IDEF1X).

[45] H. ESKÌCÌOGLU, M. S. NİŞLİ, and S. E. KILIÇ (1985) An Application of Geometric Programming to Single-Pass Turning Operations, Proceeding of the MTDR Conference.

[46] A. F. PLATT (1999), White Paper Draft, “Windows DNA for Manufacturing”, Microsoft Corporation.

[47] K. HITOMI (1971) Analysis of Production Models-Part I The Schedal Decision of Production Speeds, AIIE Transactions 8(1) 96.

[48] Kappel, G., Vieweg, S., 1994, “Database Requirements for CIM Applications”, Integrated Manufacturing Systems, Vol. 5 No. 4/5, pp. 48-63

[49] Ünver, H. Ö., Anlağan, Ö., 2000, “Design and Implementation of an Agent Based Shop Floor Control System using Windows DNA”, International Journal of Computer Integrated Manufacturing, - submitted –

[50] Ünver, H. Ö., Cangar, T., Anlağan, Ö., Kılıç, E., 2000, "A structured methodology for development of heterarchical control software for manufacturing cell using Windows DNA", Intelligent Control Systems (ICS 2000), 14-17 August 2000, Honolulu, Hawaii, -accepted-

APPENDIX A

USERS MANUAL

A.1 Hardware Boot Up

The hardware used by the “Agent Version 1.1” system can be divided into two groups: machines and computers. Initially, it is convenient to power up the hardware before starting the programs. The machines' boot up sequence of METUCIM, used by the “Agent v1.1” is listed below:

- **Compressor:** It is located at the floor level of the B-Block Building at the entrance. Power up and wait until the steady state is reached.
- **CNC Turning Machine (Mirac):** Power up the lathe by the switch located at the bottom of the machine. The serial port connection (RS232) to the Agent PC and the auxiliary port connection to the relay box should be established. By default CNC Turning Machine uses COM2 for serial communication and Sensor Channel 2 for I/O Card. Chuck and door are pneumatically powered, and require a pressure level of 100-110 psi (7-7.5 bar). After turning on the editor screen will appear. Move the axes to their end positions by pressing +X and +Z keys. Press “Jog” and index the tool magazine to the Left Hand Side Cutting tool by writing “T1<EOB>”. By default the “Agent v1.1” system uses pre-defined tool offsets based on work pieces of 70 mm in length. Load the tool offset file “70” from the “F9” menu. Leave the chuck open by writing “M10” in the Jog screen.

- **CNC Milling Machine (Triac):** Power up the milling machine by the switch located at the left side of the control box. The serial port connection (RS232) to the Agent PC and the auxiliary port connection to the relay box should be established. By default CNC Milling Machine uses COM1 for serial communication and Sensor Channel 5 for I/O Card. Chuck, door, and tool magazine are pneumatically powered, and require a pressure level of 100-110 psi (7-7.5 bar). After turning on the editor screen will appear. Move the axes to their end positions by pressing +X, +Y and +Z keys. Position the chuck at the most right, top and front position of the machine. By default the "Agent v1.1" system uses pre-defined tool offsets based on work pieces of 70 mm in length. Load the tool offset file "70" from the "F9" menu. Leave the chuck open by writing "M10" in the Jog screen.
- **Robot (Mitsubishi, Movemaster EX):** Turn on the robot and by the switches located the back of the control box. The serial port connection (RS232) to the Robot Host PC and the LAN connection of the Host PC to the network should be established. After turning on, move the switch at the control box to "ON" and press "NST" and "ENT" to move to the nest position. Leave the switch at the "OFF".
- **Pneumatic Linear Robot Drive (Festo):** Connect the power supply, note that PLRD shares the same relay box with the CNC Turning and Milling Machines' sensor channels, located at its middle feet.
- **Buffer (SKF):** Connect the power supply. The correct initial position of the cups is when the counter switch is at the OFF position (when the cup just passes the switch).

A.2 Software Boot Up

Also computers and related software should be boot up properly for the system to function. There are a total number of six computers and related software in the system:

- **Primary Domain Controller:** Its computer name is "cimlabserver" and its IP address is "144.122.69.170". The PDC hosts to the Microsoft Message Queue (MSMQ) services, Internet Information Server (IIS), and Microsoft Transaction Server (MTS). It is the main storage of VB program source codes, and web ASP and HTML pages.
- **Primary Backup Controller:** Its computer name is "cimlabbackup" and its IP address is "144.122.69.47". The PBC also hosts to the Microsoft Message Queue (MSMQ) services, Internet Information Server (IIS), and Microsoft Transaction Server (MTS). The web site, and SQL Server is hosted on the PBC. The IIS, SQL Server Service Manager and Agent Explorer should be running during the service.
- **Agent Controller:** Its computer name is "cimlab-ws5" and its IP address is "144.122.69.164". The Agent Controller hosts to the Microsoft Message Queue (MSMQ) services, Microsoft Transaction Server (MTS), and all Agent Controllers. "Agent Starter" program can be used to start/stop the agents and perform auxiliary clean up services for the queues and database. A total number of 8 agent controllers should be running on this computer: *CNC_Agent_1, CNC_Agent_2, CNC_Agent_3, CMM_Agent, Robot_Agent, Buffer_Agent, PLRD_Agent, and AGV_Agent*.
- **Robot Controller:** Its computer name is "METUCIM1" and its IP address is "144.122.69.87". The Robot Controller hosts to the "Robot Server" program, which must be started initially.

A.3 Installation

The work includes a complete web site, an SQL Server 7.0 Database, 8 device controllers and 18 DB objects including the *Part_Agent*, and an *Agent_Explorer*. For a complete installation please follow the indicated steps:

1. The primary domain controller, primary backup controller, and agent controller PCs should run on Windows NT 4.0 or higher. At least 100 MB free hard disk space and 16 MB RAM is required.
2. PDC and PBC require Windows NT 4.0 Option Pack Setup Internet Information Server, Message Queue Server, and Transaction Server installed, PBC also requires Microsoft SQL Server 7.0 installed. Agent Controllers require Message Queue and Transaction Servers installed.
3. Install the web site "Agent" on the IIS of the PBC by copying all pages in "\WebSite" directory on the CD supplied.
4. Install the database "\Database" on the SQL Server 7.0 of the PBC by selecting restore database command from the SQL Server menu.
5. Install the Agent Explorer to the PBC by double clicking the "\AgentExplorer\Setup.exe".
6. Install all agent controllers, *CNC_Agent_1, CNC_Agent_2, CNC_Agent_3, Robot_Agent, Buffer_Agent, PLRD_Agent_1, AGV_Agent* by clicking on the setup icon "\DeviceController\[Controller_Name]\Setup.exe" on the related Agent Controller PC from the CD supplied.
7. Install the host controller of robot by clicking on the setup icon "\DeviceController\Robot_Agent\RobotServer\Setup.exe" on the robot host PC.
8. Install the database drivers, Messenger and Part_Agent objects by clicking on the setup icon in the "\MTSObject\Setup.exe" on every computer in the system.

Note that the "Agent v1.1" system requires administrator assistance in installation of numerous distributed program components.

A.4 Agent Explorer

The Agent Explorer is designed to act as a server for externally sent work orders from the web site. It should be running on the PBC which also hosts to web

site and database system. It has an Browser like style to watch Computers, installed Device Agents, created Part Agents, active/finished tasks of the active work orders. There are also connections to the web site and database. A screenshot from the Agent Explorer's main screen with the Part_Agent_Operations selected is shown in Figure A1.

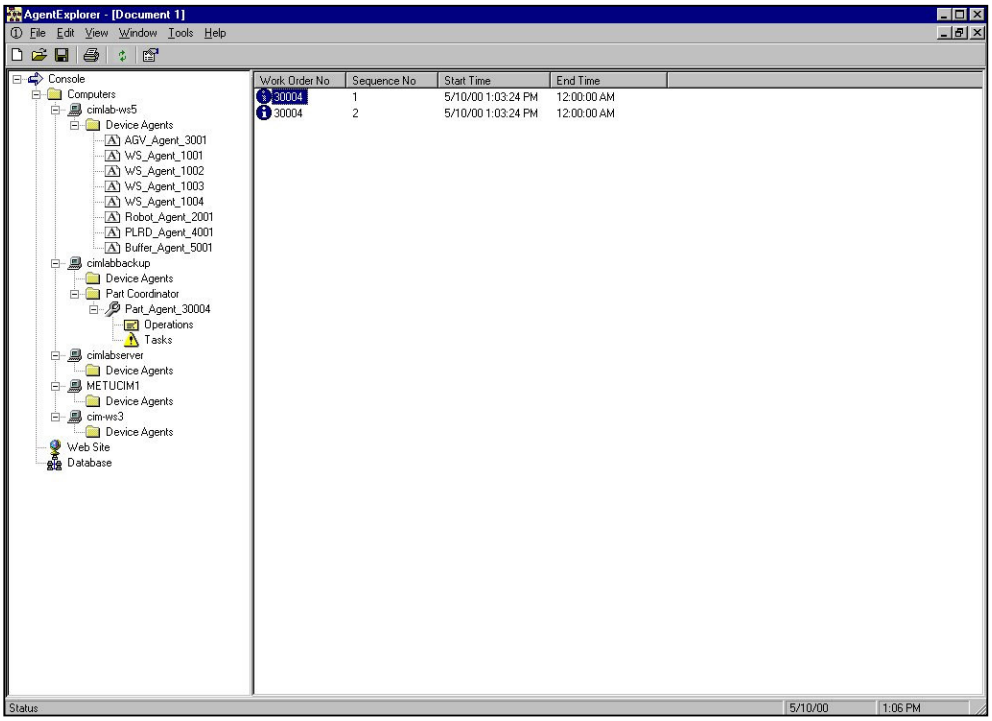

Figure A.1 Agent Explorer main screen, Part_Agent_Operations

A.5 Device Controllers

Each Device Controller is responsible for utilizing the capabilities of the manufacturing/transportation device. For this it may use parallel-, serial communication, PLC control, Digital/Analog I/O. The agent controllers are simple driver software for the individual machines, connected to a main database and messaging system. Agent controllers are classified in 5 tabs:

- **Properties:** Displays the static data about the name, type, brand, computer name and configuration string, also the current status is indicated (Figure A.2).
- **Events:** Displays both queue and task related events occurred.
- **Queues:** Displays the queues and current messages in the selected queue.

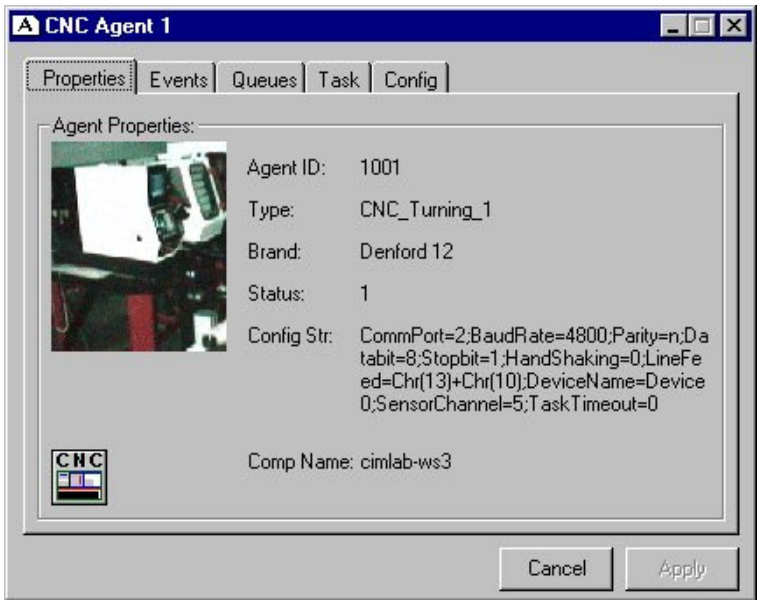

Figure A.2 CNC Turning Machines Agent Properties Screen

- **Task:** Besides the automatic application integrated with the Agent system, the user can also dispatch single tasks to the device. The interface displays the task dispatching by the task parameter to the CNC Turning machine.
- **Config:** The configuration screen gives a list of configuration alternatives for the specified control mechanism of the device agent. For example the robot agent should be configured at comport 2, with the indicated serial port properties on the remote robot host machine METUCIM1. The task timeout indicates the allowable time period of the task to complete.

A.6 Web Site

The web site is published on http://cimlabserver.me.metu.edu.tr/Agent. One can reach a detailed explanation about the use of the Agent at http://cimlabserver.me.metu.edu.tr/Agent/Help.htm. The complete web site is also on the "\website\" directory located on the CD. To publish these pages on an IIS server create a new folder on IIS Web "Agent" directory for example "C:\InetPub\wwwRoot\Agent" and simply copy all files here. Figure A.3 shows the published Help page of the site.

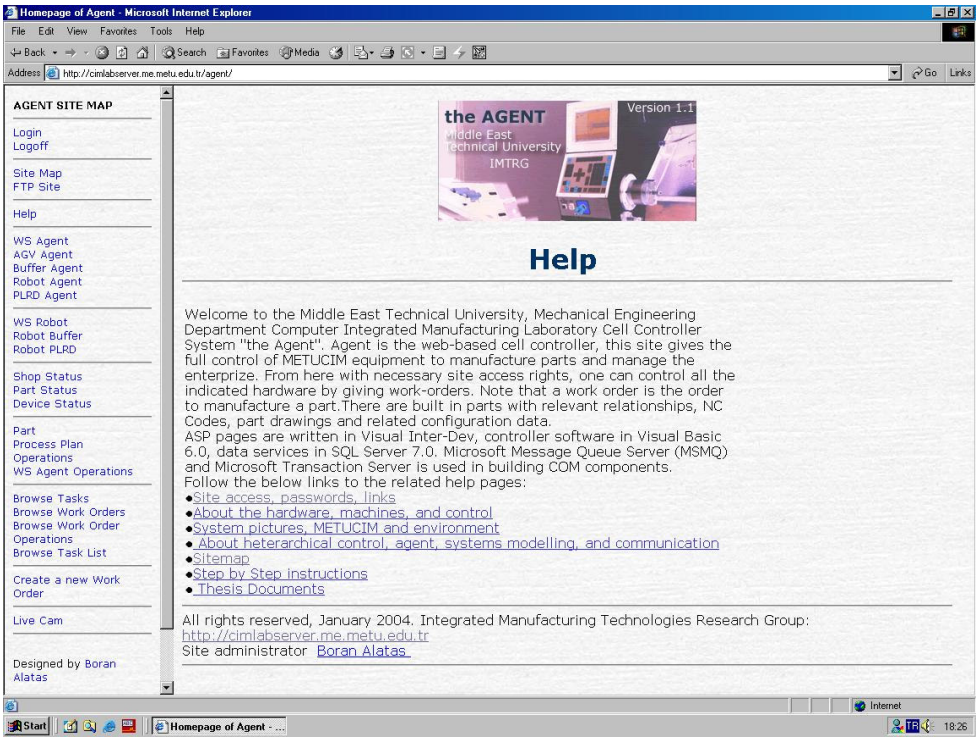

Figure A.3 Help page of the published web

APPENDIX B

G CODES OF TEST RUNS

In Chapter 5 two test runs have been demonstrated. The first run is based on manufacturing of five parts each having two operations. The parts are brass bars of 30 mm in diameter and 70 mm in length. Similarly in the second run there is a single work order having three parts batch, but three work orders recorded to database for every part produced. Work orders, process plan IDs and corresponding G-Code Listings of the first and second test runs are given in Table B1.

Table B.1 Test runs and corresponding G-Code Listings

Part ID	Process Plan ID	Work Order No	Seq. No	In Test Run	Machine	G-Code Listing
10001	20001	30198	1	1st	Turning	Fig. B1
"	"	"	2	1st	Milling	Fig. B2
10002	20001	30199	1	1st	Turning	Fig. B3
"	"	"	2	1st	Milling	Fig. B4
10003	20001	30200	1	1st	Turning	Fig. B5
"	"	"	2	1st	Milling	Fig. B6
10004	20001	30201	1	1st	Turning	Fig. B7
"	"	"	2	1st	Milling	Fig. B8
10005	20001	30202	1	1st	Turning	Fig. B9
"	"	"	2	1st	Milling	Fig. B10
10002	20001	30203 30204 30205	1	2nd	Turning	Fig. B11
"	"	"	2	2nd	Milling	Fig. B12

```
[BILLET Z80 X30
G21
M11
M39
G28 U0 W0
M06 T1
G99
G40
G97 S1500
M03
!FACING
G00 X32 Z2
G01 Z-0.1 F0.2
X-1 F0.1
X32 Z2
G00 X26.5
G01 Z-39.5 F0.1
X32 F0.2
G00 Z2
X25.5
G01 Z-39.5 F0.1
X32 F0.2
G00 Z2
X22
G01 Z-30 F0.1
X25.5 Z-35
X28 F0.2
G00 X32
Z2
```

Figure B.1 Cont'd

```
X20.5
G01 Z-27.5 F0.1
X28 F0.2
G00 X32
Z2
!SURFACE FINISH OF UPPER
!SIDE
G00 X20
G01 Z-25 F0.1
X23 F0.2
G00 X32 Z0.2
!POCKET
G28 U0 W0
M05
M06 T3
M03
G00 Z-12
X22
G01 X15 F0.1
G00 X22
Z-13.9
G01 X15
G00 X22
Z-15.8
G01 X15
G00 X22
Z-17.7
G01 X15
G00 X22
Z-19.5
G01 X15
G00 X22
Z-20
G01 X15
G00 X22
X32 Z2
!AGAIN CUTTING INCLINED
!SURFACE
G28 U0 W0
M05
M06 T1
M03
G00 X25 Z3
Z-18
X17
G01 Z-20 F0.1
Z-30 X22
Z-28
G00 Z-18
X15.5
G01 X15 F0.1
Z-20
Z-35 X25
Z-40
G03 X30 Z-42.5 R2.5
G00 X32
Z2
!DRILLING
G28 U0 W0
M05
M06 T2
M03
G00 X0 Z2
G01 Z-4 F0.1
G00 Z2
G28 U0 W0
M05
M06 T8
M03
G00 X0 Z2
G01 Z-8
G00 Z2
G28 U0 W0
M05
M38
M10
M30
```

Figure B.1 Turning operation G-Code of part 10001

```
[BILLET X40 Y40 Z80
G21
G94
G40
G97 S2000
M11
M39
G28
M06 T3
M03
G00 X0 Y0 Z2
G01 Z-10 F100
G01 X0 Y0 Z2 F400
G28
M05
M06 T1
M03
G00 X15 Y0 Z2
Z-2
G01 X-15 F100
G00 Z-4
G01 X15 F100
G00 Y15
X0
Z-2
G01 Y-15 F100
G00 Z-4
G01 Y15 F100
Z4
G28
M05
M38
M10
M30
```

Figure B.2 Milling operation G Code of part 10001

```
[BILLET Z80 X30
G21
M11
M39
G28 U0 W0
M06 T1
G99
G40
G97 S1500
M03
G00 X28 Z2
G01 X28 Z-43 F0.1
G01 X32 F0.2
G00 Z2
G00 X26
G01 Z-15 F0.1
G01 X28 F0.2
G00 Z2
G00 X20
!CREATING THE TIP IN TWO
!STEPS
G01 Z-10 F0.1
G00 X24
Z2
G00 X16
G01 Z-10 F0.1
G01 X18 F0.2
G00 Z2
G00 X10
G01 Z-10 F0.1
G01 X12 F0.1
Z2
!GIVING THE FINAL SHAPE TO
!THE TIP
G01 X5 Z0 F0.2
G01 X10 Z-10 F0.1
G00 Z2
G00 X0
G01 Z0 F0.2
G01 X10 Z-10 F0.1
G00 Z2
G00 X0
G01 Z-2.5 F0.1
!ROUNDING THE END OF THE
!TIP
G03 X4 Z-4 R3 F0.1
```

Figure B.3 Cont'd

```
G00 X30
G00 Z-20
G01 X28 F0.2
!CREATING THE POCKET
G01 X20 Z-28 F0.1
G01 Z-40
G02 X30 Z-48 R17 F0.1
G00 Z2
!OPERATION IS COMPLETE
G28 U0 W0
M05
M38
M10
M30
```

Figure B.3 Turning operation G Code of part 10002

```
[BILLET X50 Y50 Z80
G21
G94
G40
G97 S2000
M11
M39
G28
M06 T2
M03
G00 X20 Y0
G00 Z2
G00 Z-15
!CREATING THE ARCS ON THE FIRST STEP
G01 X16 F150
G01 X21
Y21
G01 X12
G01 X8 Y14
G01 X12 Y21
G00 X-12
G01 X-8 Y14
G01 X-12 Y21
G00 X-21
G00 Y0
G01 X-16
G00 X-21
G00 Y-21
G00 X-12
G01 X-8 Y-14
G01 X-12 Y-21
G00 X12
G01 X8 Y-14
G01 X12 Y-21
!CREATING THE HEXAGONAL !SHAPE ON THE SECOND STEP
G00 X17.5 Z-25
G01 Y9.5 F150
G01 X0 Y19.5
G01 X-17.5 Y9.5
G01 Y-9.5
G01 X0 Y-19.5
G01 X17.5 Y-9.5
G00 X20 Y20
!CHANGE THE TOOL AND CREATE THE SLOT AROUND THE TIP
G28
M05
M06 T1
M03
G00 X7 Y0 Z0
G00 Z-9
G01 Z-11.5 F150
G03 X-7 Y0 R7 F150
G03 X7 Y0 R7
G01 Z2 F400
!MANUFACTURING OF THE PART IS COMPLETE
G28
M05
M38
M10
M30
```

Figure B.4 Milling operation G Code of part 10002

```
[BILLET Z80 X30
!PREPATORY
G21
M11
M39
G28 U0 W0
M06 T1
G99
G40
G97 S1500
M03
!FACING
G00 X32 Z2
G01 Z-0.2 F0.2
X-1 F0.1
G00 X32 Z2
!FINE CONTOURING
G00 X29.8
G01 Z-49.7 F0.1
G00 X32 Z2
!THINNING THE CYLINDER
!UNTIL 23.8 MM IN 2 TIMES
G00 X26.8
G01 Z-43.7 F0.1
X32 F0.2
G00 Z2
!SECOND PASS
G00 X23.8
G01 Z-43.7 F0.1
X32 F0.2
G00 X32 Z2
!MAKING CURVED SHAPE IN
!TWO PASS
G00 Z-14.2
G01 X23.8 F0.1
!FIRST PASS
G02 X23.8 Z-39.7 R39.5 F0.1
G00 Z-14.2
!SECOND PASS
G02 X23.8 Z-39.7 R21.4 F0.1
G00 X32 Z2
!CHANGE THE TOOL TO
!PARTING TOOL
M05
G28 U0 W0
M06 T3
M03
!OPENNING 4 SLOTS
G00 X32 Z-21.8
G01 X12 F0.1
G00 X32
G00 Z-27
G01 X12 F0.1
G00 X32
G00 Z-32.1
G01 X12 F0.1
G00 X32
G00 Z-37.7
G01 X12 F0.1
G00 X32
G00 Z2
!OPERATION COMPLETE
M05
M38
M10
M30
```

Figure B.5 Turning operation G Code of part 10003

```
[BILLET X30 Y30 Z80
G21
G94
G97 S1500
M11
M39
G28
!CHANGING TOOL TO BALL
!MILLING
G28
M05
M06 T3
M03
!FIRST DRILL
G00 X9 Y0 Z4
G01 Z-14 F150
G00 Z4
!second DRILL
G00 X0 Y9 Z4
G01 Z-14 F150
G00 Z4
!THIRD DRILL
G00 X-9 Y0 Z4
G01 Z-14 F150
G00 Z4
!FOURTH DRILL
G00 X0 Y-9 Z4
G01 Z-14 F150
G00 Z4
M05 !changing tool
M06 T1
M03
!OPENNIG THE GROOVE WITH
TWO CIRCLE AND ONE PASS
!FIRST CIRCLE
G00 X7 Y0 Z4 !first pass
G01 Z-2 F150
G03 X-7 Y0 R7 F150
G03 X7 Y0 R7
G00 Z4
G00 X7 Y0  !second pass
G01 Z-4 F150
G03 X-7 Y0 R7 F150
G03 X7 Y0 R7
G00 Z4
G00 X7 Y0 !third pass
G01 Z-6 F150
G03 X-7 Y0 R7 F150
G03 X7 Y0 R7
G00 Z4
G00 X7 Y0 !fourth pass
G01 Z-8 F150
G03 X-7 Y0 R7 F150
G03 X7 Y0 R7
G00 Z4
!SECOND CIRCLE
G00 X5 Y0 !first pass
G01 Z-2 F150
G03 X-5 Y0 R5 F150
G03 X4 Y0 R5
G00 Z4
G00 X5 Y0 !second pass
G01 Z-4 F150
G03 X-5 Y0 R5 F150
G03 X4 Y0 R5
G00 Z4
G00 X5 Y0 !third pass
G01 Z-6 F150
G03 X-5 Y0 R5 F150
G03 X4 Y0 R5
G00 Z4
G00 X5 Y0 !fourth pass
G01 Z-8 F150
G03 X-5 Y0 R5 F150
G03 X4 Y0 R5
G00 Z4
!FINAL PASS
G00 X5 Y2 Z-8
G01 X-5 F150
G00 Z4
G00 X5 Y-2 Z-8
G01 X-5 F150
!FINISHING OPERATION
G28
M05
M38
M10
M30
```

Figure B.6 Milling operation G Code of part 10003

```
[BILLET Z80 X30
G21
M11
M39
G28 U0 W0
M06 T1
G99
G40
G97 S2500
M03
G00 X32 Z2
G01 Z-0.1 F0.2
X-1 F0.1
G00 X32 Z2
X27
G01 Z-37 F0.1
X32 F0.2
G00 Z2
X24
G01 Z-37 F0.1
G01 Z-40 X30
X32 F0.2
G00 Z2
X21
G01 Z-37 F0.1
X32 F0.2
G00 Z2
X20
G01 Z-37 F0.1
X32 F0.2
G00 Z-10
G01 X20 F0.1
G02 X20 Z-37 R37.7 F0.1
G00 X32
G28 U0 W0
M05
M06 T3
M03
G00 Z-20.5 X20
G01 X10 F0.1
X20
G00 Z-24.5
G01 X10 F0.1
X20
Z-28.5
G01 X10 F0.1
X20
G28 U0 W0
M05
M06 T2
M03
G00 X0 Z2
G01 Z-4 F0.1
G00 Z2
G28 U0 W0
M05
M06 T8
M03
G00 X0 Z2
G01 Z-7 F0.1
G00 Z2
G28 U0 W0
M05
M06 T4
M03
G00 X0 Z2
G01 Z-7 F0.1
G00 Z2
G28 U0 W0
M05
M38
M10
M30
```

Figure B.7 Turning operation G Code of part 10004

```
[BILLET X40 Y40 Z80
G21
G94
G40
G97 S1500
M11
M39
G28
M06 T1
M03
G00 X0 Y0 Z3
Z-2
G01 X14 F200
X0
Y14
Y0
X-14
X0
Y-14
Y0
Z-4
X14
X0
Y14
Y0
X-14
X0
Y-14
Y0
G00 Z3
G28
M05
M06 T3
M03
G00 X0 Y0 Z3
G01 Z-7 F400
G00 Z3
G28
M05
M38
M10
M30
```

Figure B.8 Milling operation G Code of part 10004

```
[BILLET Z80 X30
!PREPARATORY FUNCTIONS
G21
M11
M39
G28 U0 W0
M06 T1
G99
G97 S1500
M03
G00 X32 Z2
!FACING OPERATION
G01 Z-0.1 F0.2
X-1 F0.1
G00 X32 Z2
!ROUGH CUTTING (ONLY TWO
!PASSES ARE ENOUGH)
!1ST PASS
G00 X28
G01 Z-38 F0.1
X32 F0.2
G00 Z2
!2ND PASS
G00 X24.5
G01 Z-20 F0.2
X32 F0.2
G00 Z2
!3RD PASS
G00 X20
G01 Z-10 F0.2
X32 F0.2
G00 Z2
!4TH PASS
G00 X14.5
G01 Z-13 F0.2
X32 F0.2
G00 Z2
!FINE CONTOUR CUTTING
G00 X14
G01 Z-14 F0.1
```

Figure B.9 Cont'd

```
X24 Z-15
Z-20
G02 X28 Z-22 R2 F0.1 C.
G01 X28 Z-38
G00 X32
Z2
!SLOT CUTTING
G28 U0 W0
M05
M06 T3
M03
G00 X32 Z-30
G01 X20 F0.1
G00 X32
Z2
!OPERATION IS COMPLETE
G28 U0 W0
M05
M38
M10
```

Figure B.9 Turning operation G Code of part 10005

```
[BILLET X30 Y30 Z70
!PREPARATORY FUNCTIONS
G21
G94
G40
G97 S1500
M11
M39
G28
M06 T1
M03
!SQUARE SLOT CUTTING
!OPERATION IN ONE CUT
G00 X0 Y0 Z4
G01 Z-5 F100
X2
Y2
X-2
Y-2
X2
Y2
G00 Z5
!OPERATION IS COMPLETE
G00 X-132 Y65 Z61
M05
M38
M10
```

Figure B.10 Milling operation G Code of part 10005

APPENDIX C

PROCESS DIAGRAM

The following process flow diagrams have been constructed using IDEF1X modeling technique. They intend to summarize the process flow of the sample functions in the agent system, namely the "Work order creation", "Performing a pre-task", "Construction of a bid", and "Performing the own task". The key for reading the diagrams is in Figure C.1.

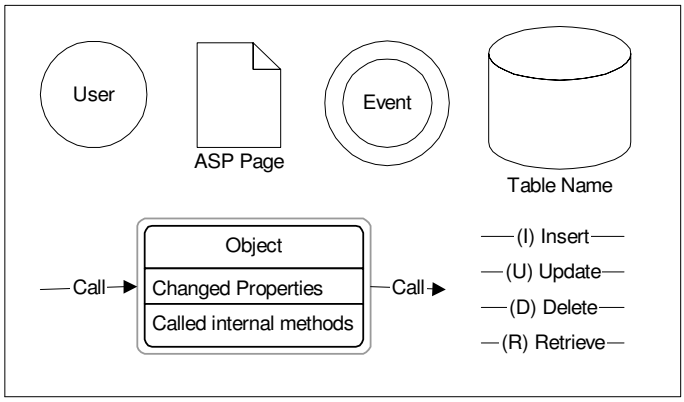

Figure C.1 Key for the process diagram

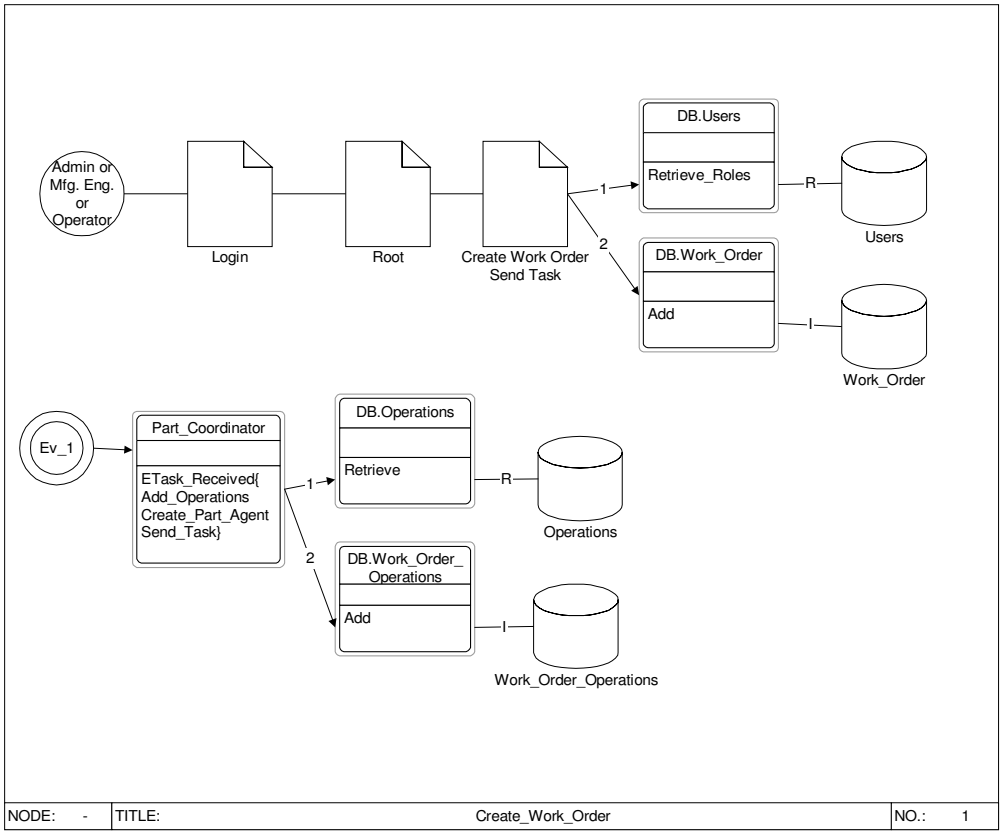

Figure C.2 Process flow of the "Create Work Order" Function

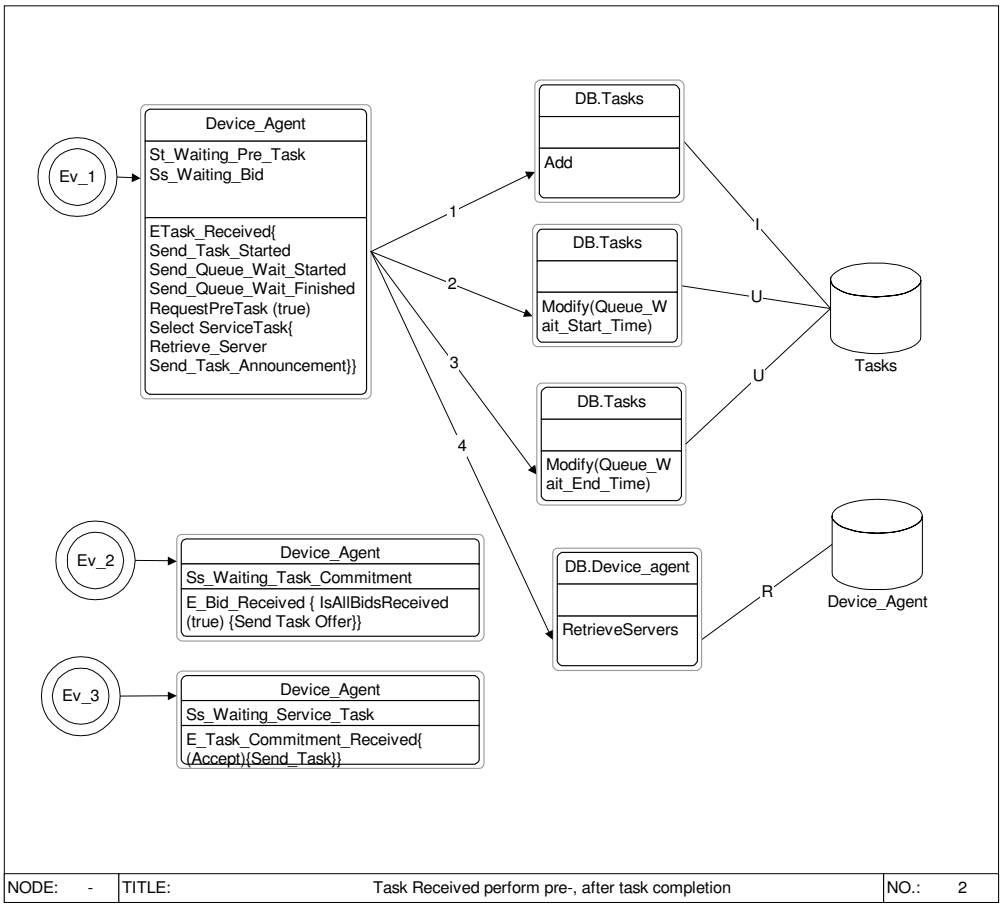

Figure C.3 Process flow of the "Perform Pre-Task" function

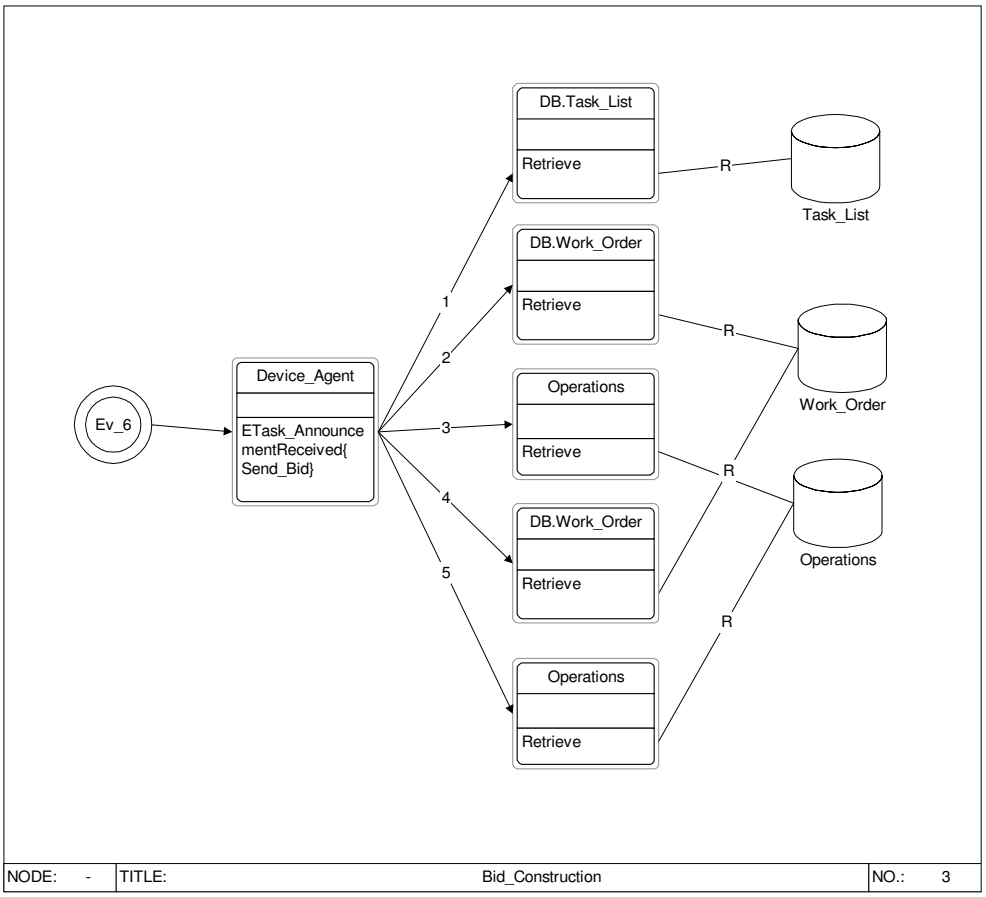

Figure C.4 Process flow of the "Bid Construction" function

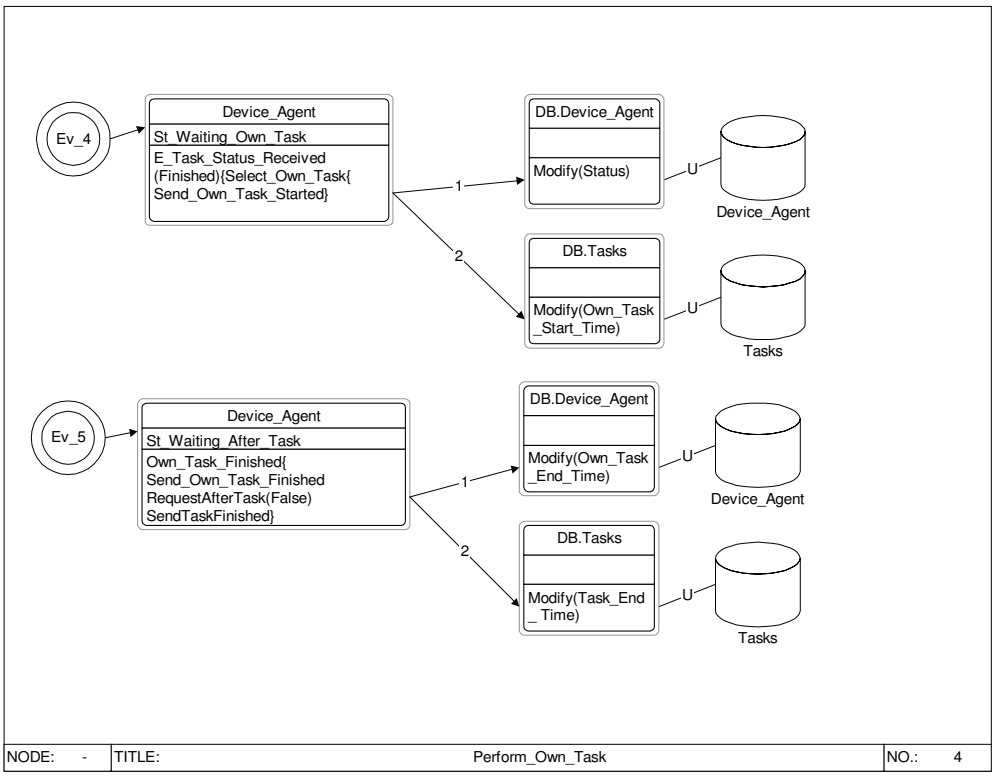

Figure C.5 Process flow of the "Perform Own-Task" function

APPENDIX D

KEYS & IDS

Device Agent, Operation, Part, Process Plan, and Tasks are related with an ID number, defining its primary key. It is more convenient to give them in tables for further reference, Agent IDs are given in Table D.1, database entry ranges of the Part, Process Plan, Sequence No, and Work Order are in Table D.2, and generic tasks are given in Table D.3. METUCIM layouts with loading/unloading positions are given in Figure D.1.

Table D.1 Agent IDs

Agent Name	**Agent_ID**	**Range**
WS_Agent	1001 :CNC Turning Machine 1002 :CNC Milling Machine 1003 :CMM 1004: Dummy CNC Turning	1000-1999
Robot_Agent	2001: Robot	2000-2999
AGV_Agent	3001: Static AGV	3000-3999
PLRD_Agent	4001: PLRD	4000-4999
Buffer_Agent	5001: Buffer	5000-5999
Part_Coordinator	6001: Part Coordinator (unique)	6001
Part_Agent	30001-39999: Becomes equal to created work order no	30001-39999
User	8001: Generic ID for User (unique)	8001

Table D.2 Database entry ranges

Table	**ID**	**Range**
Part	10001:Generic Part ID	10001-19999
Process_Plan	20001:Generic Process Plan ID	20001-29999
Sequence_No	1:Generic Sequence No	1-999
Work_Order	30001:Generic Work Order No	30001-39999

Table D.3 Generic Task IDs

Task ID	**Task Of**	**Task Description**
100001	CNC Turning	Generic Task ID for the CNC Turning
100002	CNC Milling	Generic Task ID for the CNC Milling
100003	CMM	Generic Task ID for the CMM
100004	Dummy CNC Turning	Generic Task ID for the Dummy CNC Turning
200001	Robot	Generic Task ID for the Robot
201003	Robot	from 1 to 10 (Cell loading sequence)
201004	Robot	from 2 to 3 (Machine loading sequence of CNC Lathe)
201008	Robot	from 3 to 2 (Machine unloading sequence of CNC Lathe)
201013	Robot	from 7 to 8 (Machine unloading sequence of CNC Milling)
201015	Robot	from 8 to 7 (Machine loading sequence of CNC Milling)
201016	Robot	from 8 to 9 (Machine loading sequence of CMM)
201017	Robot	from 9 to 8 (Machine unloading sequence of CMM)
201020	Robot	from 10 to 4 (Cell unloading sequence “Accept”)
201021	Robot	from 10 to 5 (Cell unloading sequence “Reject”)

Table D.3 Cont'd

201022	Robot	from 10 to 6 (Cell unloading sequence "Rework")
300001	AGV	Generic Task ID for the AGV
301001	AGV	Load a part into system
301002	AGV	Unload part to Accept
301003	AGV	Unload part to Reject
301004	AGV	Unload part to Rework
400001	PLRD	Generic Task ID for the PLRD
401001	PLRD	Go left
401002	PLRD	Go right
401003	PLRD	Change Position
500001	Buffer	Generic Task ID for the Conveyor
501001	Buffer	Load a part into 2, take it from 2 to 8
501002	Buffer	Load a part into 2, take it from 2 to 10
501003	Buffer	Load a part into 8, take it from 8 to 2
501004	Buffer	Load a part into 8, take it from 8 to 10
501005	Buffer	Load a part into 10, take it from 10 to 2
501006	Buffer	Load a part into 10, take it from 10 to 8
501007	Buffer	Unload 2
501008	Buffer	Unload 8
501009	Buffer	Unload 10
501011	Buffer	Take next empty buffer to 2, load with partID
501012	Buffer	Take next empty buffer to 8, load with PartID
501013	Buffer	Take next empty buffer to 10, load with PartID
501014	Buffer	Take next full buffer to 2,unload it
501015	Buffer	Take next full buffer to 8, unload it
501016	Buffer	Take next full buffer to 10, unload it
600001	Part Coordinator	Generic Task ID for the Part Coordinator
700001	Part Agent	Generic Task ID for the Part Agent
800001	User	Generic Task ID for the User Announcement

APPENDIX E

SAMPLE CODE

The following code is taken from the *Messenger* objects *SendMessage* method. It shows the basic use of MSMQ objects, MSMQ Queue, MSMQ Message and its properties as well as using MTS functions of SETABORT and SETCOMPLETE:

```
Option Explicit
Option Base 1
#Const usemts = True
Private mQueue As MSMQQueue

Public Function SendMessage(mPathName As String, Optional mMessageLabel
As String, Optional mMessageBody As String, Optional mAppSpecific As Long, _
Optional mQueue As MSMQQueue) As String
        '-------------------------------------------------------------------------------
        ' This function sends a message to the queue with the given parameters -
        ' Note that the queue must exist otherwise will give an error          -
        '-------------------------------------------------------------------------------

        On Error GoTo errhandler

        #If usemts Then
           Dim objcontext As ObjectContext
           Set objcontext = GetObjectContext
        #End If

        Set mQueue = New MSMQQueue
        Dim mInfo As New MSMQQueueInfo
        Dim mMessage As New MSMQMessage

        mInfo.PathName = mPathName
        Set mQueue = mInfo.Open(MQ_SEND_ACCESS, MQ_DENY_NONE)

        mMessage.Label = mMessageLabel
```

```
        mMessage.Body = mMessageBody
        mMessage.AppSpecific = mAppSpecific
        mMessage.Send mQueue

        'ReOpen the Queue for ReceiveAccess
        Set mQueue = mInfo.Open(MQ_RECEIVE_ACCESS, MQ_DENY_NONE)

        SendMessage = "OK"
        objcontext.SetComplete
        Set objcontext = Nothing
        Exit Function

errhandler:

        #If usemts Then
           objcontext.SetAbort
           Set objcontext = Nothing
           SendMessage = "SETABORT"
        #End If

        'Destroy
        Set mQueue = Nothing
        Set mInfo = Nothing
        Set mMessage = Nothing

        SendMessage = Err.Number & "/" & Err.Source & "/" & Err.Description

End Function
```