Proceedings of the 15th International Conference on Computer-Aided Production Engineering, CAPE'99
P. G. Maropoulos and J. A. McGeough (eds.) © Univ of Durham, School of Engineering, Publishers.

REALIZING INTEGRATED INTELLIGENT MANUFACTURING SYSTEM WITH DISTRIBUTED INTELLIGENT AGENTS

Reggie Davidrajuh[1], Ziqiong Deng[1] and Kesheng Wang[2]

[1] Department of Computer Assisted Production Engineering
Narvik Institute of Technology, Norway
{rd, zd}@hin.no

[2] Department of Production and Quality Engineering
Norwegian University of Science and Technology (NTNU)
kesheng.wang@protek.ntnu.no

ABSTRACT

The advances in information, telecommunications, and computing technologies enable us to investigate newer manufacturing paradigms to design, develop, and produce qualitative products faster, and cheaper. In this paper, we explore a new manufacturing paradigm called Integrated Intelligent Manufacturing System (IIMS) where manufacturing resources (databases, graphics programs, multimedia systems, etc.) are integrated with the manufacturing applications (CAD, CAM, etc.). The integrating infrastructure in IIMS is not necessarily within an enterprise- it may be inter-enterprise, connecting collaborating enterprises, spreading over different networks and platforms. Integration of resources with manufacturing applications is realized with intelligent interfaces. We propose modular- object oriented platform independent, mobile intelligent agents, called Distributed Intelligent Agents (DIA), as interfaces for realizing integrating infrastructure of IIMS.

In this paper, we present a framework for designing an IIMS based on DIA. We show how the framework enables us to design the components of the integrating infrastructure. We then show how to develop an IIMS, by presenting a prototype. Problems in the design of IIMS and ways to tackle these difficulties are also discussed.

KEYWORDS

Mobile intelligent agent, object-oriented technology, Java, CORBA, Databases, and knowledge bases.

INTRODUCTION

Modern manufacturing systems are larger and complex therefore effective techniques should be employed to develop, design, and produce a class of products to response the market with the attractive and qualitative products, flexible and fast responsiveness (agility), and lowest price (leanness) [1]. These basic objectives (Attractiveness-Quality-Agility-Leanness) of modern manufacturing may well be achieved through integrated software and hardware architectures that generate decisions based on information collected from the manufacturing systems environment and knowledge developed by the experts in manufacturing process, and execute these decisions across communications networks. This will be a large knowledge integration environment, consisting several symbolic reasoning systems, numeric computation systems, neural networks, data base management systems, computer graphics programs, and multimedia. This kind of manufacturing environment is called (IIMS). That is, IIMS is a manufacturing system where manufacturing processes are integrated and knowledge intensive (Figure 1).

Figure 1 shows the concepts of IIMS, where the basic manufacturing applications such as computer aided design (CAD), computer aided manufacturing (CAM), computer aided quality control (CAQ), computer aided process planning (CAPP), and production planning and control (PP&C) are not intelligent. In an IIMS, integration of these manufacturing applications with knowledge bases is realized by intelligent interfaces. The integrating infrastructure for IIMS is not necessarily within an enterprise; it may be well inter-enterprise, connecting collaborating enterprises, through different networks, and platforms.

In IIMS environment, the intelligent interfaces have to solve the following problems:

- Provide secure and faster information flow,

- Be platform independent (works on UNIX, Windows NT, etc.) and network transparent (compatibility with different network protocols),
- Facilitate seamless connection with knowledge-based systems, databases, and incorporation of legacy systems, and
- Facilitate efficient management of different intelligent systems.

Until very recently, building an IIMS that is intelligent and distributed was not possible due to lack of enabling technologies. We propose Distributed Intelligent Agents (DIA) as interfaces for realizing integrating infrastructure of IIMS environment. DIA is a mobile intelligent agent written in platform independent object-oriented Java language, and enhanced with the distributed object (Common Object Request Broker Architecture - CORBA) capabilities for network transparency.

With the integration of knowledge, the manufacturing activities such as CAD, CAM etc. become intelligent applications of manufacturing, and can be classified as (Figure 2): Intelligent design, Intelligent process planning, Intelligent operation, Intelligent production planning, Intelligent testing and assembling, and Intelligent maintenance [2]. Advantages due to these intelligent applications are added flexibility in the manufacturing functions such as configuration management, decision support, and automatic failure detection and avoidance.

In this paper, we present our vision for the development of IIMS based on DIA. The scope of this paper is limited to design of DIA suitable for interfacing manufacturing resources such as knowledge bases, databases and multimedia. The remainder of the paper is organized as follows. In the next section, a framework for designing IIMS based on DIA is presented. Section 3, a prototype of IIMS is developed. In the conclusion section, we give some problems encountered in the design and development of IIMS and also present some directions for further work.

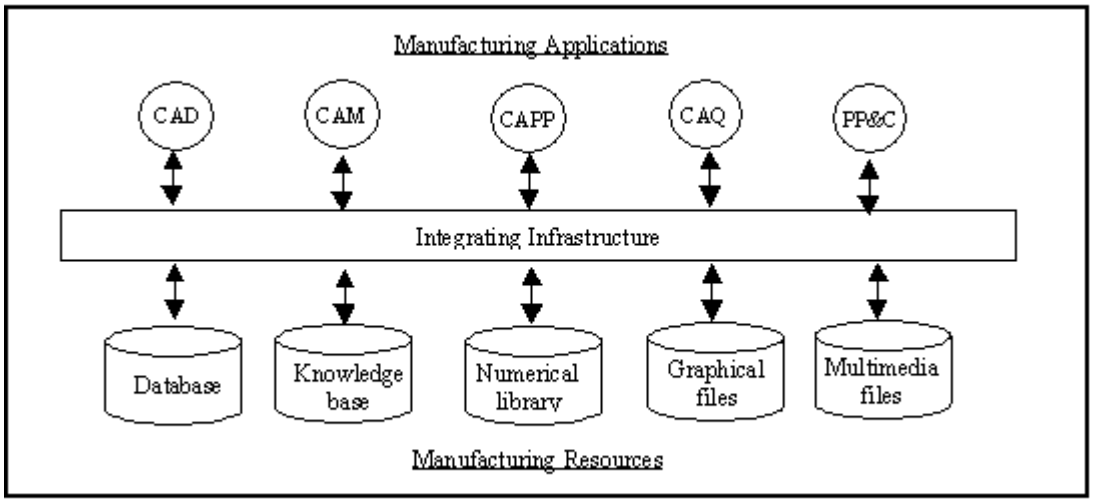

Figure 1. Integrated Intelligent Manufacturing System

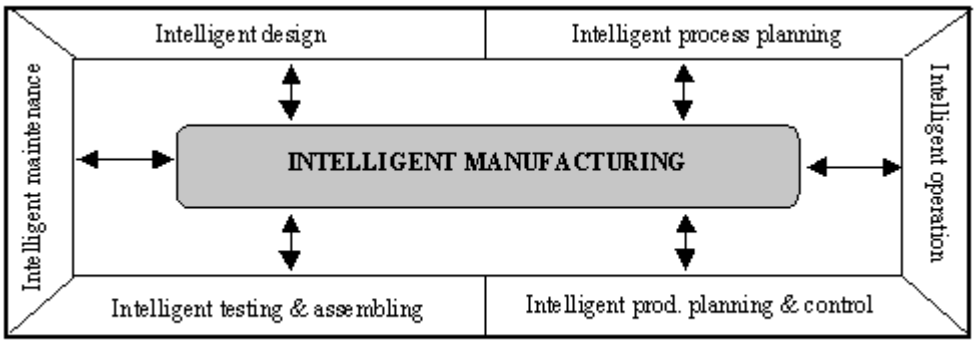

Figure 2. Integration of intelligent manufacturing applications

DESIGN OF IIMS

The Framework

In this subsection we first present a framework for designing and developing an IIMS based on DIA.

The framework that is presented here starts with a simple view of the information infrastructure. There are three major actors dealing with the information infrastructure:

1) The manufacturing resources and applications.
2) Networks for mobility of information, and
3) Control or supervisory mechanisms for routing and control of information.

Therefore our framework too is divided into three inter-related (and overlapping) views (see Figure 3). This division is to reduce complexity and to allow independent and parallel development of components under different views.

The three views are 1) Control view, 2) Mobility view, and 3) Component-communication view. Figure 3 shows the formation of views of the framework. Figures 4, 5 and 6 show the detailed exposition of the views.

The control view has three layers of abstraction: Application interface layer, Control layer and Communication layer as shown in Figure 4. The application interface layer represents the connection of the applications and resources to the integrating infrastructure through agents. The communication layer represents the underlying communication infrastructure (the hardware and the net operating system). The middle layer is the control layer, where two central controllers are situated to take basic control actions on agents and to offer fundamental services to the agents; these two controllers are the *Agent Manager* and the *Service Manager*. The agent manager controls the

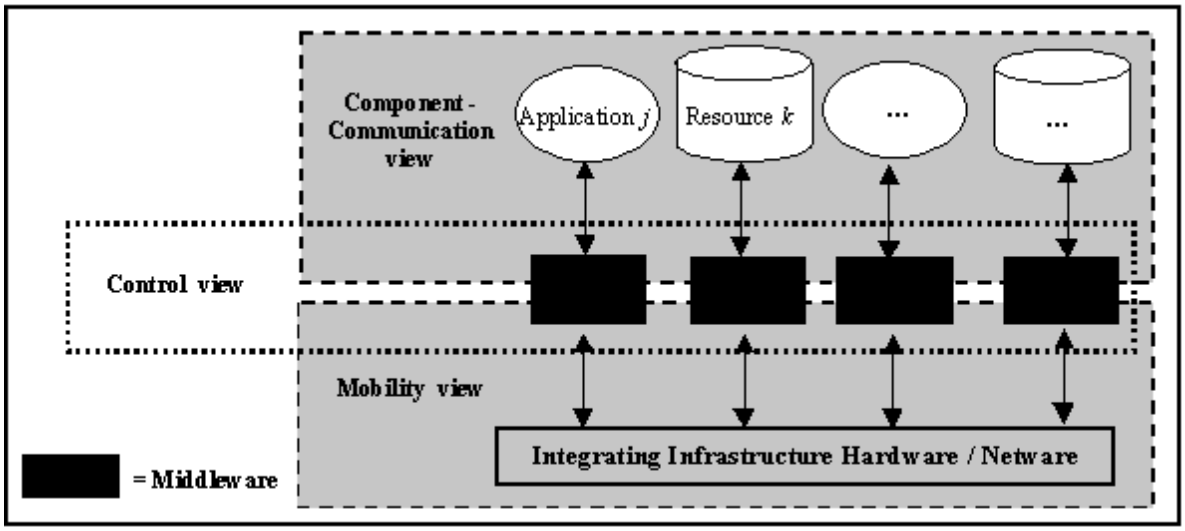

Figure 3. Formation of the three views of the framework

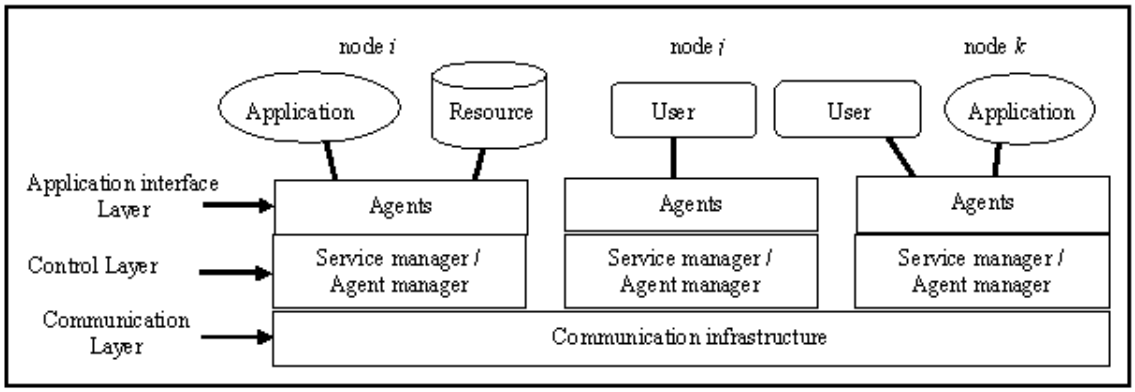

Figure 4. Control view

life cycle of the agents and their actions, and offers relevant services relating to this area. Agent startup, execution context, inter- communication, and persistency are under the agent manager control. The service manager offers system services to the agents such as security checks, access permission, performance monitoring, system failure- reliability reporting, queuing of agents, and mobility and transportation of agents. There is strong interaction between these two controllers for all the agent activity.

There may be also a third kind of controller the *Utility Manager* or utility services manager (not shown in the Figure 4) which offers utility services (services and control actions that are not offered by agent manger or service manager) to the agents.

system to serve the mobile agents seeking data from these resources to satisfy the applications which started them. To enable this, resources have to be encapsulated, and the request to them and response from them to the agents has to be transparent of where the resources are located. With CORBA technology the kind of common bus for mobile agent transportation as shown in Figure 5 can be created. To do this, different nodes where diverse applications and resources are attached, should use CORBA object request broker (ORB). Local resources attached to the individual nodes are encapsulated by CORBA interface definition language (IDL), and their presence is let known to all the nodes by registering in the local interface repository.

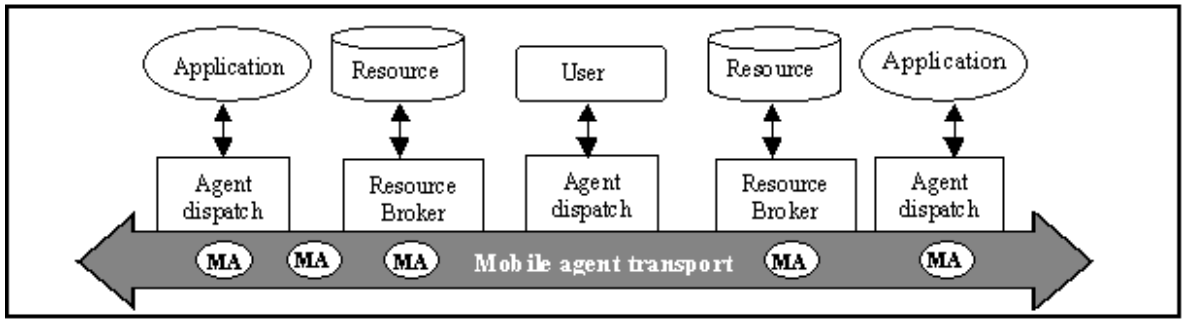

Figure 5. Mobility view

Figure 5 depicts the mobility view of the framework. In this view multiple resources on heterogeneous platforms are pooled together as a homogeneous

Figure 6 shows the component-communication view. This view identifies the components and their communication relationship. There are several kinds of agents, and this kind of

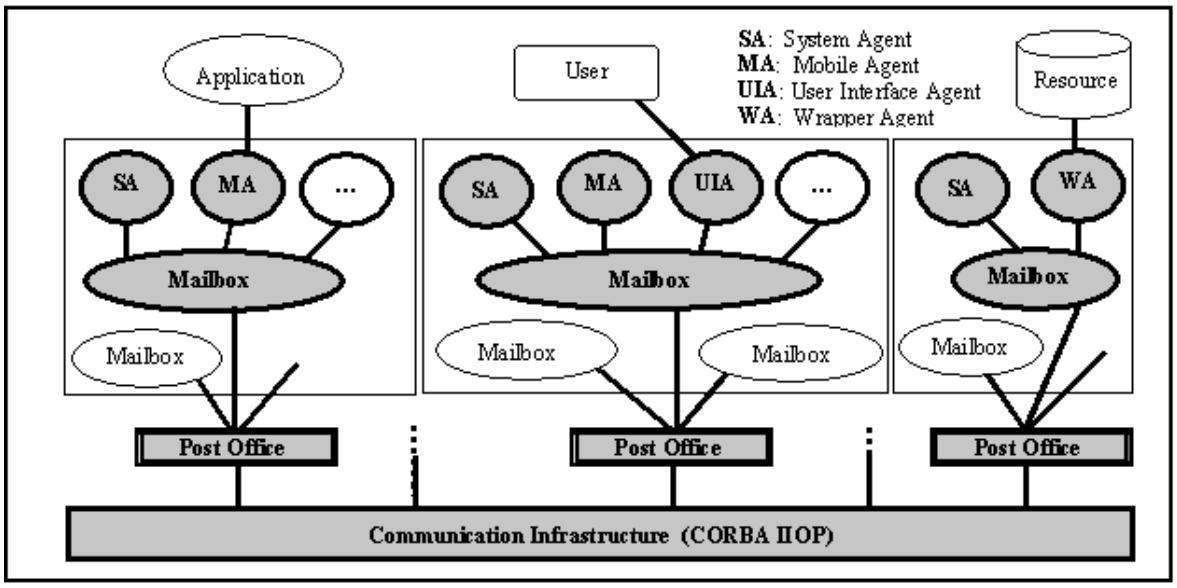

Figure 6: Component- communication view (adapted from [8])

differentiation of agents strongly influences the framework. In addition to the mobile agents, there are *static agents* too. The *system agents* are static agents that provide infrastructure-level services to other agents. A *resource agent* or *wrapper agent* is a static agent that provides a level of abstraction between a resource and requesting mobile agent. The resource agent understands how to access the resource and the permission structures associated with the resource and wrap-up legacy code for transparent cross-platform access. The third kind of static agent is the *user-interface agent*. The user-interface agent is probably a graphic user interface (GUI) that abstracts the user away from the details of the mobile agent architecture.

The component-communication view also shows the agent inter-communication mechanism and their relevance to the communication infrastructure. Agents communicating with each other through communication channels that are linked to mailboxes (messages queues), which get delivery services from the post offices. For agent communication, mailboxes are better mechanism than procedure-call mechanism (Java remote method invocation - RMI) or callback mechanism (Java abstract window toolkit - AWT, Swing) [3]. For communicating across networks, the messaging protocol CORBA-GIOP (General Inter ORB Protocol) is used because of its simplicity, scalability and generality. IIOP (Internet Inter ORB Protocol) is the TCP/IP version of GIOP.

DEVELOPMENT OF IIMS

In this section we present a prototype of IIMS under development. This prototype illustrates an application accessing STEP (standard for the exchange of product model data) files in a buffer database attached to a different platform. Though this example is not intended to represent fully the concepts of IIMS, it shows the basics of building an IIMS.

Description of the scenario

Consider the scenario of changes (e.g. update, deletion) in the buffer database containing product data 'STEP' files. Whenever this is done, there is a need to inform the manufacturing applications that use these files. Otherwise, the applications that have already used the files will find inconsistency. In this kind of situation which are triggered by some internal or external events (file deletion, updating) a pool of DIA help to maintain the working condition of manufacturing applications and resources integration, by offering the following services:

- Monitoring the internal and external triggering events and record keeping,
- Inferencing for routine tasks, early detection of potential problems (due to file change, in our scenario),
- Coordinating with the manufacturing applications, e.g. manufacturing application interfacing, and
- Coordinating with the manufacturing resources, e.g. database.

The Prototype

Current version of our prototype under development is:

1) Based on Windows NT & UNIX(Sun Solaris),
2) No Real- Time restrictions are imposed,
3) Based on network protocol TCP/IP, and
4) Communication between agents are similar to that of Java RMI.

The abstract architecture of this prototype containing basic components is shown in Figure 7.

The agent development system

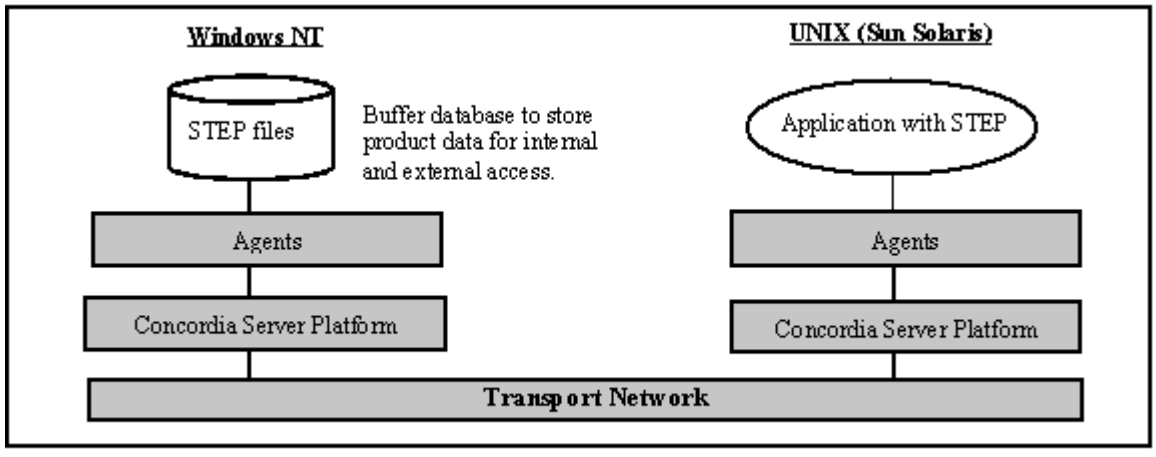

Figure 7. Abstract architecture of the prototype.

Based on the framework given in the previous section, we chose Concordia [4] as the middleware upon which we developed agents. Concordia is an agent development system that hides the complexities of application program mobility from programmers, so that we could concentrate on solving the problem rather than solving the programming complexities.

Types of agents

The prototype system consists of five agents. They are application interface agent (AIA), broker agent (BRA), coordinating agent (COA), event monitoring agent (EMA), and wrapper agent (WRA). See figure 8. BRA is a mobile agent. The other agents, AIA, COA, EMA, and WRA are static agents.

Application Interface Agent
Whenever the application is launched application interface agent (AIA) becomes active to serve the application. To supply the data required to the application, AIA request BRA to get it from the resource. In the case of inconsistency, AIA consults COA in association with EMA.

Broker Agent
Broker agent (BRA) is responsible for obtaining the data from the resource and supplying it to AIA. Being a mobile agent, BRA travels to the resource site, gets the data from the resource through WRA and returns to the application site.

Coordinating Agent
The idea of coordination agent is to function as specialist agent on the site and help the other agents in achieving their goals. COA contains solutions for primary (or critical) problems such as what to do when the AIA finds that the requested file from the database was not found or the database is not available. COA maintains a narrow knowledge base for this purpose. In short, COA performs the inferencing and early problem detection as well as advisory functions.

Event Monitoring Agent
Whenever an event makes changes to database, this triggers the event-monitoring agent to register about the changes made. This information will be made available to agents (such as COA, BRA) seeking info about the changes made to the database.

Wrapper Agent
Wrapper agent (WRA) wraps the database using CORBA-IDL, expressing the interfaces to access the database. Thus WRA is to make the database appear as any distributed (CORBA) object.

CORBA - Mobile Agent Integration Strategy

In our scenario, a mobile agent (BRA) travels from the site where an application running on Windows NT, to the site where a UNIX machine is supporting a database, to seek information from the database (Figure 8). Though it is much easier to encapsulate Java database connectivity (JDBC) driver as the WRA, the database has been wrapped into a CORBA object instead, just to make it fit our theoretical framework which employ DIA, which actually means CORBA and mobile agent. Making database as a CORBA object also satisfies our intention to represent the database as any (legacy) CORBA object, not just a database.

In our prototype, mobile agent - CORBA

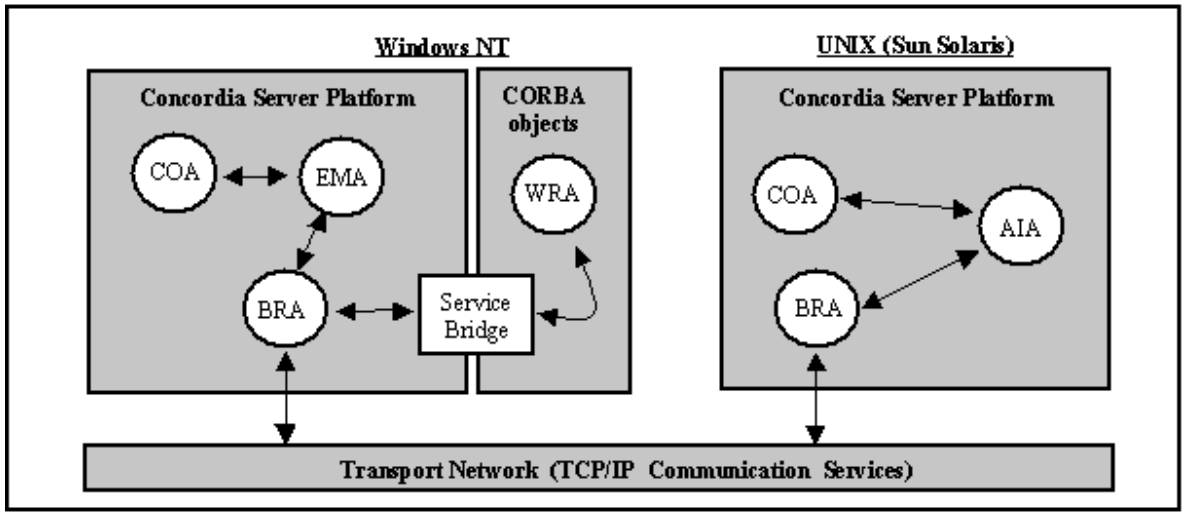

Figure 8: The agent prototype.

integration is achieved by using the following strategy: First the mobile agent BRA travels to the resource site. On the resource site, the resource (database) is encapsulated with CORBA-IDL and thus made as a CORBA object.

On the resource site, Concordia Service Bridge is used to encapsulate the ORB specific calls to the CORBA object (see Figure 9). When the mobile agent (BRA) wants to access the database (CORBA object), the service bridge would locate the CORBA object, retrieve a reference to it, and then delegate method calls down to the CORBA object (Figure 9).

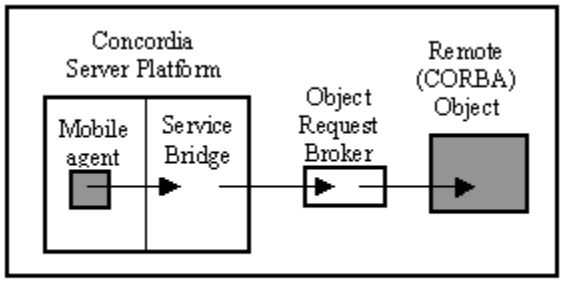

Figure 9: Integrating mobile agent with CORBA using the Concordia Service Bridge.

There are many reasons for using this strategy (mobile agent travel to a Concordia Server located on the same computer as the CORBA object):

1) Most importantly, this strategy satisfies our agent-CORBA integration need.
2) By this approach, the agent needs to know nothing about the specifics of the CORBA API. From agent's point of view, it is just accessing services using standard Concordia mechanisms.
3) When both agent and the CORBA object are located on the same computer (as in our prototype), and if the ORB has a "same server" optimization then it may bypass the TCP/IP stack and use shared memory, named pipes, or other inter process communication mechanisms. This will yield much better performance.

CONCLUDING REMARKS

After presenting the framework of the IIMS in the previous section, we give below some more detailed work that is needed to integrate the manufacturing resources like knowledge bases, databases, and multi-media systems into IIMS.

Further work on Real-time operations

When dealing with certain kinds of manufacturing resources and applications, such as multimedia, there should be allowances for real-time operations. In a heterogeneous system where applications and resources are tied up with CORBA, providing real-time or dynamic mechanisms becomes a severe problem because CORBA has no built-in support for real-time (or dynamic) activities. The framework we propose must incorporate enhanced components to provide event-triggered (or real-time) approach for predictable agent (and application) start-up time, arrival time, execution time, deadline constraints, etc.

Appropriations for real-time operations can be incorporated in our framework with the help of a scheduler. In the control layer (see figure 4) the agent manager employs the scheduler to allocate resources to the agents and to control their real-time execution in order to guarantee predictability. To perform these duties, the scheduling component of the agent manager make uses of the hardware clock and dispatching services of the standard real-time operating system that resides in the communication layer, the layer beneath the control layer. If the operating system does not offer standard real-time services, then these services may be added to it with the help of a 'plug-on' shell.

Further work on Knowledge-base integration

For IIMS to become truly 'intelligent', inclusion of different kinds of knowledge bases is vital. When including different kinds of knowledge bases, to solve the problems of heterogeneity, a knowledge base and its access mechanism has to be divided into three parts [5]: 1) 'Problem-solving method', which is about how to accomplish the query task. 2) 'Knowledge base', which has the data/information about the domain that is necessary for a method to operate. Knowledge base is method independent; it can serve as source of information for more than one method. 3) Wrapper agents to adapt the methods to different knowledge bases.

With this partitioning, problem-solving methods and knowledge bases can be reused across different applications, applications apply a method to different knowledge bases, and each knowledge base used by different methods. However, developing wrapper agents in this situation is a very difficult task; Normally a specific wrapper agent is required for connecting a particular method to a particular knowledge base. This means $m \times n$ number of different wrapper agents required for m number of methods and n number of knowledge bases. There are research efforts going

on to reduce the different number of wrapper agents needed, like using Open Knowledge Base Connectivity (OKBC) application programming interface (API) to wrap-up different knowledge bases thus make them look like generic knowledge bases [5].

Conclusions

This paper presents a framework for developing an integrated intelligent manufacturing system. Since the scope of this paper is focuses on the overall framework, the other main topics relevant to DIA such as internal architecture, agent control activities, agent coordination mechanisms and cooperation issues, inter agent communication approach, etc. not described in-detail here.

Being network transparent and platform independent, DIA is a very useful technology for using manufacturing applications and resources across collaborating enterprises. In both intra-enterprise and inter-enterprise, applications and resources are implemented in different programming languages, on different hardware platforms, and operating systems due to historical variance of installations from various vendors within one enterprise, or variant installations among collaborative enterprises. Therefore, enabling a seamless connection of applications and resources within the multi-vendor's environment is a basic necessity for successful cooperation of collaborating enterprises, to which DIA can be successfully used.

When designing agents for IIMS environment, we face two problems: 1) Allowances for real-time operations, and 2) Tackling the heterogeneous nature of the resources. For the real-time operations, further work is needed on in-built scheduling component of the agent manager. By this we add real-time functionality to the proposed framework, which is otherwise not given by the (static) CORBA. The other way is to employ a 'Real -Time' CORBA with all the real-time features in-built. However, real-time CORBA compliant is still at research stage [6, 7].

IIMS not only offers the advantages due to the 'intelligent manufacturing' (such as flexibility in configuration management, decision support, automatic failure detection and avoidance, and self-calibration), but also overall reduction of high cost of software development and maintenance. This is due to wrapping up of resources (different kinds of knowledge bases, databases and legacy code) that were collected over the time due to historical variance of installations from various vendors within one enterprise, or among collaborative enterprises, and reusing them.

Acknowledgements

We would like to thank Tom Walsh of Mitsubishi Electric ITA, Horizon Systems Lab, US, for his useful comments regarding Agent (Concordia Server Platform) - CORBA interactions.

REFERENCES

[1] Z. Deng, A Model of Methodology Need for AQAL Production System, in: *Proceedings of the International Conference on Integrated and Sustainable Industrial Production*, Lisbon, Portugal, 1994.

[2] P. Jeyakumar, *A Design Approach to integrate intelligent systems in a manufacturing environment*, Ph.D. Thesis, Norwegian University of Science and Technology (NTNU), Norway, 1995.

[3] T. Sundsted, Agents talking to agents, *Java World*, September 1998.

[4] http://www.concordia.mea.com/

[5] J. H. Geannari, Reuse, CORBA, and Knowledge-Based systems, SMI, Stanford University, 1998.

[6] V. F. Wolfe, Real-Time CORBA, in: *Proceedings of the Third IEEE Real-Time Technology and Applications Symposium*, Montreal, Canada, 1997.

[7] D. C. Schmidt, D. Levine, and S. Mungee, The Design of the TAO Real-Time Object Request Broker, *Computer Communications, Special Issue on Building Quality of Service into Distributed Systems*, Elsevier Science, Volume 21, No. 4, April, 1998.

[8] Farhoodi, F. and Fingar P. Developing Enterprise Systems with Intelligent Agent Distributed Object Computing, Nov. 1997