Proceedings of the 18th AIMTDR Conference,
S. Paul and A. B. Chattopadhyay (eds.) © Narosha Publishing House, ISBN 81-7319-275-8.

Enabling Technologies for Information Flow in Computer Integrated Manufacturing Systems

Reggie Davidrajuh and Ziqiong Deng
Narvik Institute of Technology (HIN)
Post box 385, N-8501 Narvik, Norway
Fax:+47-76966810 Email: {rd,zd}@hin.no

Summary
Conventional computer integrated manufacturing involves the internal integration of manufacturing processes within a single enterprise. The concepts like agile manufacturing extended CIM into virtual manufacturing where the integration is not only internal but also across collaborating enterprises. Building a virtual manufacturing system where applications are fast, secure and distributed depends on newer enabling technologies. This paper first makes a survey of the enabling technologies suitable for developing virtual manufacturing system that include object-oriented technology, request broker architecture and mobile agents.

This paper also proposes a framework for developing a virtual manufacturing system in a simulated environment. The framework is based on mobile agents in distributed objects platform, which we call distributed intelligent agents.

Keywords: Computer integrated manufacturing, virtual manufacturing, information infrastructure, object-oriented programming, Java, CORBA, distributed intelligent agent.

1. Introduction

Conventional Computer Integrated Manufacturing (CIM) involves the internal integration within a single enterprise of the manufacturing processes such as computer-aided design (CAD), computer-aided manufacturing (CAM), computer-aided process planning (CAPP), computer-aided quality control (CAQ), and production planning and control (PP&C) [1]. The concepts like lean production, agile manufacturing or global manufacturing extended CIM scope and scale into virtual manufacturing where the integration is not only internal but also across enterprises [2]. Virtual Manufacturing (VM) means a group of enterprises, dynamically organized (combined), cooperating with each other 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). This combination of enterprises is dynamic because it will be changing depending on product-market conditions. Collaborating enterprises may be geographically widespread and they may consist of a host enterprise combining to material suppliers and selling agencies [2].

There are four fundamental flows in the collaborating enterprises in a VM environment. They are material, work, information, and fund flow. This paper concentrates on enabling technologies for information flow in VM. When designing and constructing an enabling infrastructure for information flow among enterprises, the basic objectives Attractiveness- Quality- Agility- Leanness (AQAL) of VM must be kept in mind.

In order to achieve AQAL production, the infrastructure for information flow should provide the following two functions: 1) Facilitate information flow inside a given enterprise *'intra-enterprise information flow'* (information flow through different departments within an enterprise), 2) Facilitate information flow among the group of collaborating enterprises *'inter-enterprise information flow'*. Therefore enabling technologies for information flow should provide these two functions (see figure 1).

Both intra-enterprise and inter-enterprise information flow information is exchanged between program modules implemented often in different programming languages, on different hardware platforms, and different operating systems due to historical variance of installations from various vendors within one enterprise, or among collaborative enterprises. Therefore, the formulations of a seamless information flow for the multi-vendor's environment are a necessity for computer integrated manufacturing systems.

In this paper, a study is done to determine the characteristics that the enabling infrastructure should posses to provide the functions described above. The enabling infrastructure for information flow should contain the following characteristics:

- Provide a secure and faster flow,
- Be platform independent (works on UNIX, Windows 95/NT, Macintosh OS, etc.),
- Be network transparent (compatibility with different networks and protocols such as Ethernet, FDDI, ATM, TCP/IP, Novel NetWare, and various RPC),
- Facilitate seamless connection with knowledge-based systems and databases ('plug and play'),

- Programming language or implementation independent (also to incorporate existing tools or legacy systems),
- Facilitate exchange and use of different formats of files.

Until very recently, building a VM environment that is fast, secure and distributed was not possible due to lack of enabling technologies. We propose a modular- object oriented, platform independent, efficient, cooperating, mobile intelligent agents, called Distributed Intelligent Agents (DIA), as interfaces for realizing an integrating infrastructure of a VM environment. We present a framework for developing agent based VM system in section 3. In section 2, a survey of enabling technologies suitable for building a VM system is presented. In section 4, we present a prototype VM system under construction at VM lab, Narvik Institute of Technology (HIN).

2. Enabling technologies for information flow

Building a VM system faces many difficulties that were stated earlier; in summary, the system has to be multi-platform (different hardware, operating systems), network transparent (e.g. LAN, WAN, dial-up), programming language independent (e.g. inclusion of legacy code) and faster access to resources such as databases and knowledge bases. In this section, a survey is done on available enabling technologies that are suitable to construct VM systems. The technologies such as Object-oriented technology, Common Object Request Broker Architecture (CORBA), and mobile intelligent agent are reviewed here.

Object-oriented technology

The choice of object oriented technology is very important for larger and complex systems such as VM, because the object oriented technology allows independent construction and stepwise refinement of code that can be reused. Particularly, the object oriented programming language Java offers some unique features that include portability across platforms. Applications written in the Java language is platform independent, that means, these applications developed for a specific platform (say UNIX) will also run on other platforms (say Windows NT, Mac OS) as well. In addition to this property, Java also offers a cleaner and simpler code (than C++) and component model (Beans) [3].

Common Object Request Broker Architecture (CORBA) is based on distributed object oriented technology and is a vendor independent standard developed by the Object Management Group [4]. CORBA also offers many unique features that include access to objects regardless of their location (location transparency). Other features are interfaces defined independently of implementations, access to standard CORBA services and facilities, and access to objects written in other languages [3]. Access to objects written in other languages (e.g. legacy code) is possible, even across networks, with the help of CORBA's interface definition language (IDL).

Distributed Objects

When used together, network transparent CORBA with Java's implementation transparency yields 'distributed objects'. In addition, Transactional Java Beans based on CORBA Object Transaction Service (OTS) offers atomic, consistent, isolated, and durable (ACID) protection to the distributed objects [5]. The distributed object platform satisfies all the performance requirements of VM.

Mobile Intelligent Agent

The agent view provides a level of abstraction at which we construe of computational systems that inter-operate across networks linking people, organizations, and machines on a single virtual platform [6]. Agents typically posses characteristics such as autonomous, adaptive (learning), mobile, persistent, goal oriented, collaborative, flexible, and reactive [7,8]. Agents decide where they will go and what they will do, and they control their lifetime

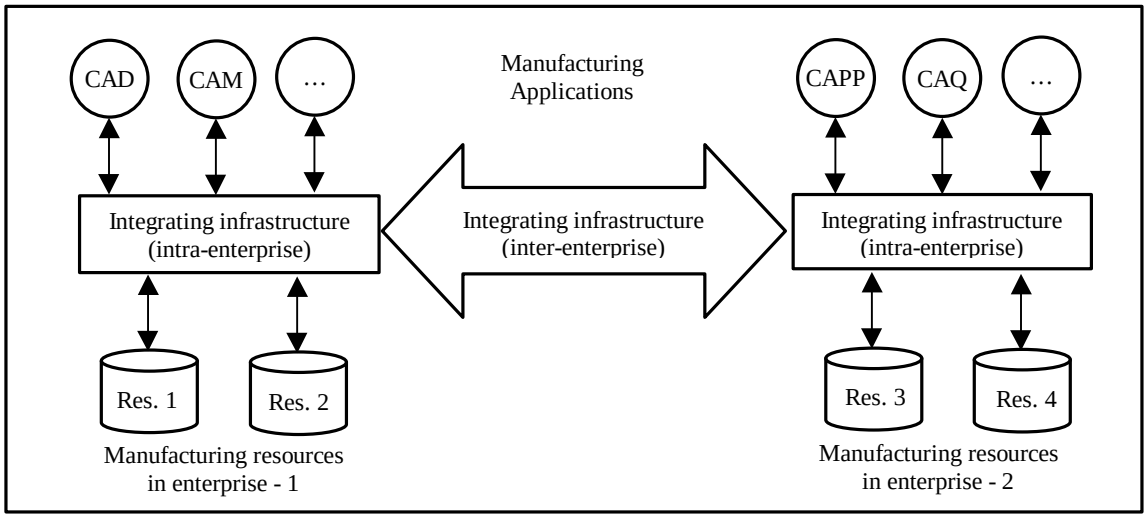

Figure 1: Extension of conventional CIM into VM.
(Though many enterprises may participate, only two are shown in the figure)

('autonomous'). They may receive requests from external sources, such as other agents, but each individual agent decides whether or not to comply with external request [9]. When a new task is delegated by the user, the agent should determine its goal precisely, evaluate how the goal can be reached effectively and perform the necessary actions ('goal-oriented', 'flexible'). An agent should also be capable of learning from past experience ('adaptive'), and should be 'reactive', so that it can sense the current stage of its environment and act accordingly [8]. Agent interacts with other agents to perform its task ('collaborative'). The Agent is 'persistent' because it is a continuously running process and it is 'mobile' because of its ability to transport itself from one machine to another [10].

Mobile agents offer a great deal of advantages over the 'static' agents (or traditional client-server paradigm):

1. Compared to the client-server architecture, mobile agents reduce the network bandwidth by moving a query or transaction from client (or remote host) to server (or local host), thus the repetitive request/response handshake is eliminated,
2. Agents reduce design risks by allowing decisions about the location of code to be pushed toward the end of the development effort when more is known about how the application will perform,
3. Because the repetitive request/response handshake is eliminated, mobile agent architecture allows applications run on low-reliable or partially disconnected networks [11].
4. Immediate reaction to incoming streams of real-time data by an agent that acts as a digital proxy for a human user,
5. Complex or larger calculations can be broken into simpler units and then assigned to agents that are forked from an "agent farm". The agents can perform the calculations on different hosts, and upon completion, the results from the agents can be summarized [12].

Because of the large amount of data, the distributed nature of the system, and because of the real-time action needed for certain type of data, the mobile agent architecture is a suitable enabling technology for information flow in VM.

Distributed Intelligent Agent (DIA)

For VM, focus is now on the type of mobile intelligent agents that collaborate with other agents, which are geographically distributed and executing on heterogeneous platform. We have seen that the distributed object architecture is a solution for heterogeneous platform. However, distributed objects architecture has to be enhanced for creating mobile intelligent agents because CORBA has no built-in support for mobile code. By adding mobility to distributed objects, a new performance paradigm shift is obtained; we call the resulting paradigm the 'distributed intelligent agents'.

3. A framework for developing VM system with DIA

In this section we present a framework for developing VM system based on distributed intelligent agents. The framework consists of three inter-related views: 1) *Control view*, 2) *Transportation* (or *mobility*) *view*, and 3) *Component-communication view*. Figures 2, 3, and 4 depict the framework for developing VM system based on DIA.

The control view has three layers of abstraction: Application interface layer, Control layer and Communication layer as shown in figure 2. The application interface layer represents the connection of the applications (such as programs, users, and resources) to the (application) agents. The communication layer represents the underlying communication infrastructure. 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 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

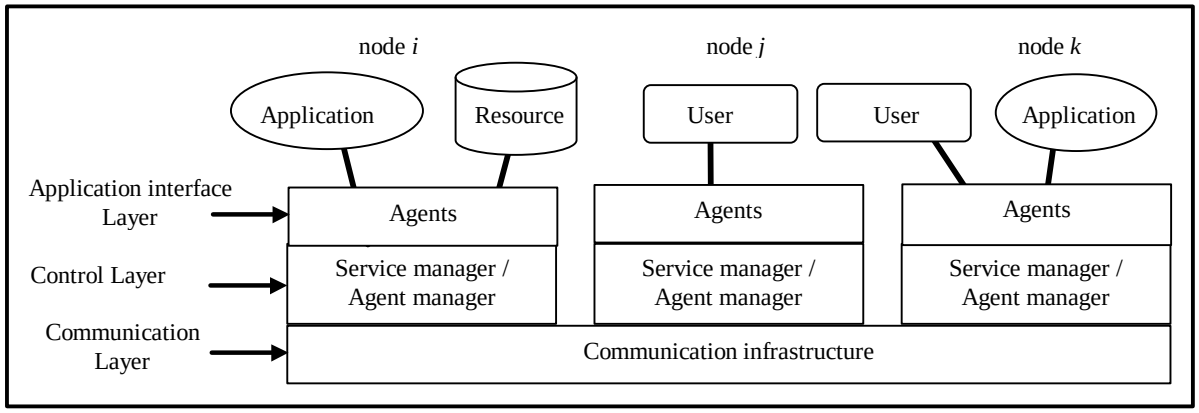

Figure 2. Control view

activity.

Figure 3 depicts the mobility view of the framework. In this view multiple resources based on heterogeneous platforms are pooled together as a homogeneous 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 request to them and response from them to the agents (resource broker) has to be transparent of where the resources are located.

Figure 4 shows the component-communication view. This view identifies the components and their communication relationship. There are several kinds of agents, and this kind of differentiation of agents strongly influences the architecture. 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 GUI that abstracts the user away from the details of the mobile agent architecture. This 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 or messages queues, which get delivery services from the post offices. For agent communication, mailboxes are better mechanism than procedure-call mechanism (Java RMI) or callback mechanism (Java AWT, Swing) [14]. 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.

4. A prototype VM system at VM lab - HIN

VM lab at Narvik Institute of Technology (*HIN*) is evolving as a research center specializing in CIM/VM area in northern Norway. VM lab can be considered as a test-

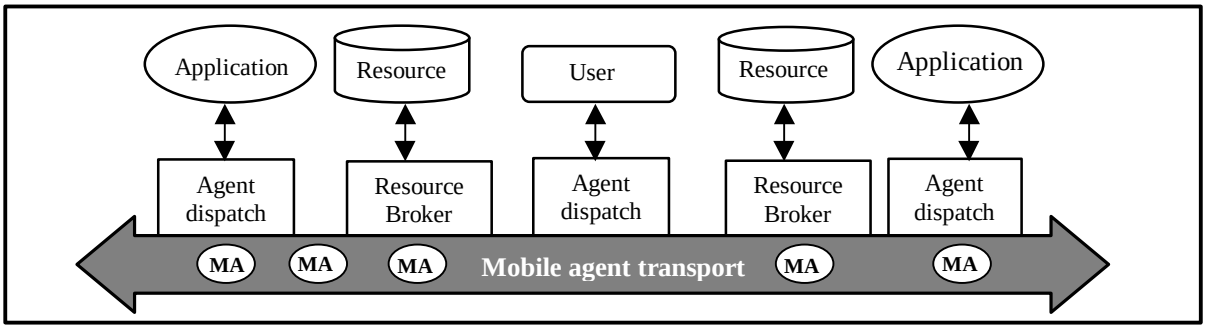

Figure 3. Mobility view

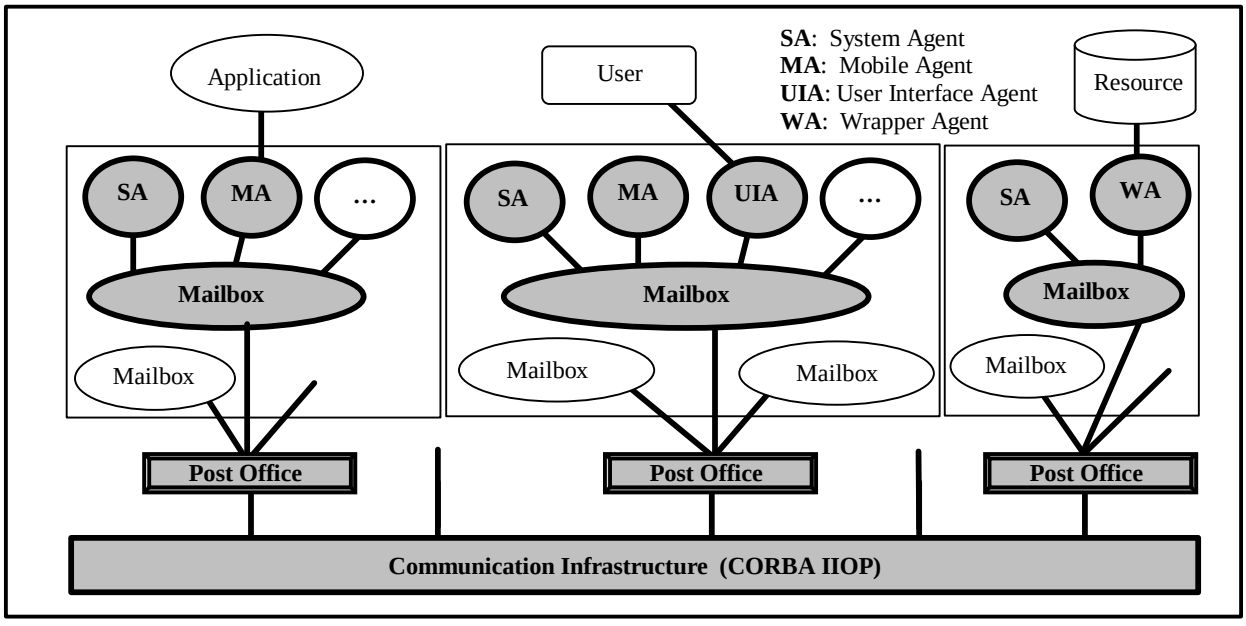

Figure 4: Component- communication view (adapted from [13])

bed for VM related R&D activities and to transfer the knowledge and experiences gained to enterprises in northern Norway so that they achieve stronger competitive position in the world market. Manufacturing enterprises in northern Norway are small so that they face difficulties in allocating resources and manpower to do R&D activities on the upcoming newer manufacturing paradigms.

At VM lab, a test-bed is under construction to simulate VM system. Figure 5 shows the test-bed. The test-bed consists of a computer park with Windows 95, Windows NT, UNIX (HP) platforms. The test-bed represent three collaborating enterprises, two of them are external industries and one is the VM-Lab itself. In the first stage, simulators for the enterprises shall be constructed on Windows NT only.

The goal of this test-bed is get experience in playing a collaborating partner role in a VM system. For example, when designing a product, the designing team (or enterprise) may not have the machine tools to check the viability of manufacturing the product. Since the necessary simulation tools are available at the VM lab, it can simulate the machine tools that are not available at a designing enterprise, thus playing a bridging role between the designer and manufacturer.

With our architectural framework's views for control, transport and communication, developing a VM system based on DIA is systemized. When choosing agent development software (middleware), the software should enable to build the VM system according to the views of the framework. In summary, the software should provide abstraction from the communication infrastructure complexities (control view), should be Java-CORBA based to provide homogenous bus for mobile agent transportation (mobility view) and should allow efficient communication mechanism (communication component view). There are many Java based and CORBA integrated agent development systems available. We found out that with the agent development system *Concordia*, building a VM system will conform to the framework we devised in the section-3 [16]. Hence, Concordia is selected as the agent development and management system for our prototype at HIN. The prototype VM system at HIN is still under development.

5. Concluding remarks

In this paper we first concluded that the Java (because of its platform independence), CORBA (location transparency) and mobile intelligent agents enable

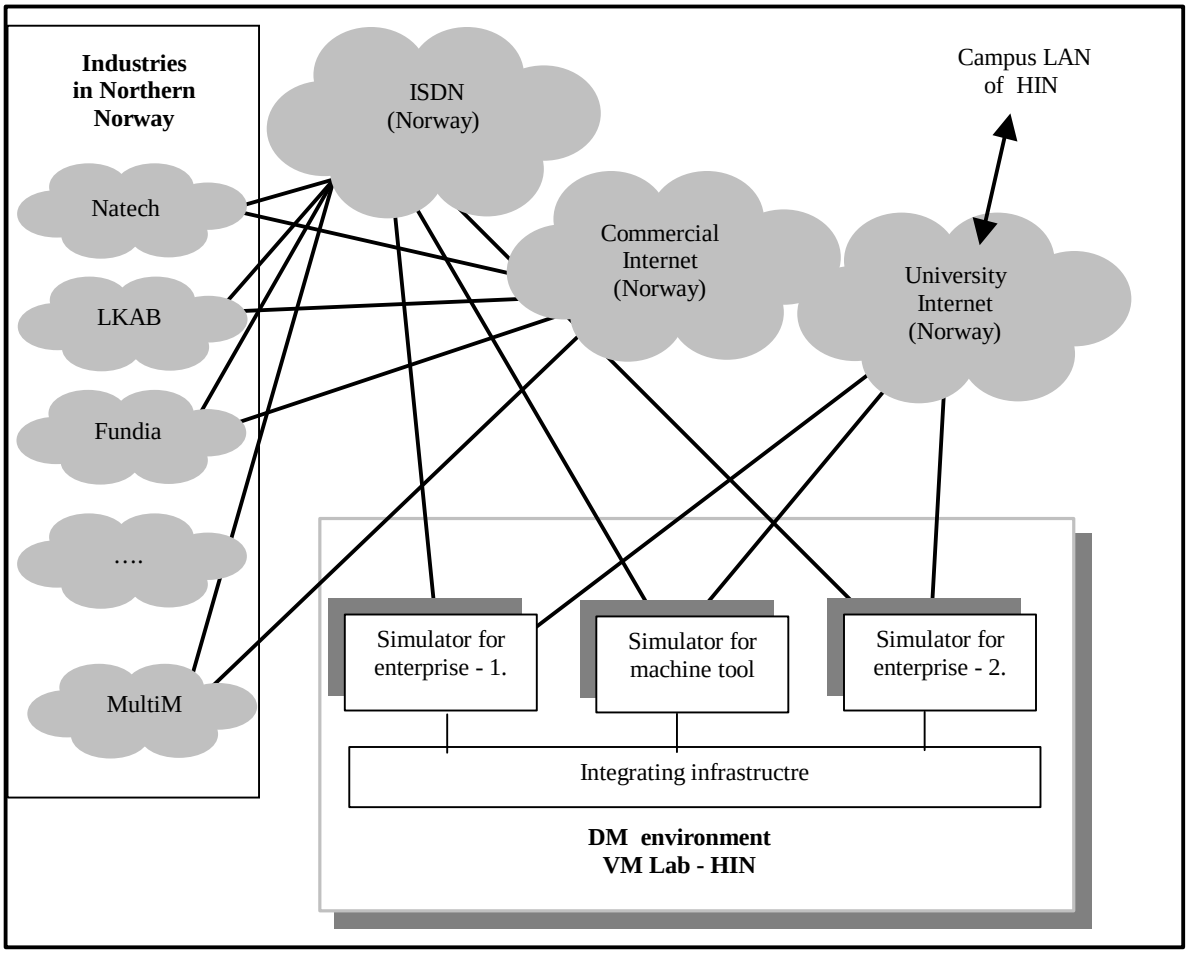

Figure 5: Test-bed for virtual manufacturing at HIN

building large heterogeneous systems (such as a VM system). We also presented a framework for developing a virtual manufacturing system based on distributed intelligent agents. As part of our future research work at Narvik Institute of Technology (HIN), we will develop a prototype of the VM system, which includes simulators of collaborating enterprises.

There are a number of problems we face when we design agent based systems. The main one is standardization problem (which is widely reported), the requirements for a number of services that are yet to be defined, such as agent definition, management, and control, models for organization, tasks, cooperation and coordination, and knowledge representation [13].

References

[1] Rembold, U. Nnaji, B. O. and Storr, A (1993) Computer Integrated Manufacturing in Engineering, Addision- Wesley.

[2] Deng, Z. (1997) A Model of Methodology Need for AQAL Production System in OE/IFIP/IEEE Conference on Integrated and Sustainable Industrial Production, Portugal.

[3] Vogel, A. and Duddy, K. (1998) JAVA Programming with CORBA, 2 ed., John Wiley

[4] Object Management Group http://www.omg.org

[5] Orfali, R. Harkey, D. and Edwards, J. CORBA, Java, and the Object Web Byte, October 1997

[6] Barbuceanu, M. and Fox, M. S. (1996) Coordinating Multiple Agents in the Supply Chain. WET ICE'96.

[7] Sundsted, T. An introduction to agents. Java World, June 1998

[9] Venners, B. The architecture of aglets. Java World, April 1997

[10] Franklin, S. and Graesser, A. (1996) Is it an Agent, or just a Program? in Proceedings of the 3rd Workshop on Agent Theories, Architecture, and Languages. Springer-Verlag.

[11] Sundsted, T. Agents on the move. Java World, July 1998

[12] Sommers, B. Agents: Not just for Bond anymore. Java World, April 1997

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

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

[15] Farley, S. R. Mobile Agent System architecture, Java Report, May 1997.

[16] Framework for agent development- Concordia: http://www.meitca.com/HSL/Projects/Concordia