INFORMATION INTEGRATION PLATFORM FOR CIMS

Felix T. S. Chan
Department of Industrial and Manufacturing Systems Engineering
The University of Hong Kong, Pokfulam Road, Hong Kong
E-mail: ftschan@hkucc.hku.hk

Jie Zhang
School of Mechanical Science and Engineering
Huazhong University of Science and Technology, China
E-mail: zhangjie@hust.edu.cn

H.C.W. Lau and A. Ning
Department of Manufacturing Engineering
Hong Kong Polytechnic University

ABSTRACT

A new information integration platform for computer integrated manufacturing system (CIMS) is presented, which is based on agent and CORBA. CORBA enhances the system integration because it is an industry-standard for interoperable, distributed objects across heterogeneous hardware and software platform. Agent technology is used to improve intelligence of integration system. In order to implement the information integration platform, we use network integration server to integrate network, design a generic database agent to integrate database, adopt multi-agent based architecture to integrate application, and utilize wrapper as CORBA object to integrate legacy code.

Keywords: Agile Manufacturing; Agent; CORBA; Application Integration.

1. INTRODUCTION

Today's marketplace is under rapid, continuous change. To survive, a competitor must not only adapt to change, but must drive change and thrive on change. Agile manufacturing can be defined as the capability of surviving and prospering in a competitive environment of continuous and unpredictable change by reacting quickly and effectively to changing markets, driven by customer-designed products and services [1]. Under the paradigm of agile manufacturing, Implementing CIMS pays more attention to process integration and enterprise integration, but information integration is essential. Therefore, CIMS must integrate all of the support systems for product and process specification, production planning and scheduling, material handling and tracking, etc. In today's manufacturing environment, there are heterogeneity of hardware and operating systems, complexity and non-harmonization of application programming interfaces, diversity of communication protocols, and user interfaces and databases involved in the applications. A distributed and open information integration platform becomes essential for CIM to integrate information in enterprise network. In this paper, a new information integration platform for CIM is presented. The platform is based on agent and CORBA so as to integrate information covering all stages of a product's life cycle and various applications.

2. INFORMATION INTEGRATION PLATFORM

2.1 Applying Agent Technology and CORBA

According to the requirement for information integration platform, its design is based on agent technology and common object request broker architecture (CORBA). These foundations were used to develop the overall infrastructure.

Some researchers have made great efforts on using multi-agent technology in manufacturing systems [2-5]. These studies have shown that agent-based architectures can offer many advantages over traditional centralized systems. Agent-based architectures are robust and dynamic; they can quickly react to unexpected events, and adapt to changing conditions. They are inherently distributed and scaleable: more agents and more computers can be added as necessary to increase the performance or

the capacity of a system. Therefore, agent-based architecture can be used to construct an information integration platform. Implementing agent-based application requires many infrastructure services, such as mechanisms for establishing communication among agents. CORBA provides most of the services needed for building agent-based applications.

CORBA technology has got more attention from research and system developer in manufacturing enterprises [6-9]. CORBA provides the best technical solution for integrating information; it is open, robust, heterogeneous, multi-platform, and multi-vendor supported. CORBA allows applications to communicate with one another no matter where they are located or who has designed them. CORBA is composed of the four elements of this object management architecture: (1) Object Request Broker (ORB); (2) Common Object Services; (3) Common Facilities; and (4) Application Objects. Object management group (OMG) uses Interface Definition Language (IDL) contracts to specify a component's boundaries and its contractual interfaces with potential clients. IDL provides operating system and programming language independent interfaces to all the services and components that reside on a CORBA bus. It allows client and server objects written in different languages to interoperate across networks and operating systems [10].

CORBA provides excellent mechanisms for integrating, sharing information, and interoperating application, and could meet the users' demands for integration and distributed-processing. Thereby, it is greatly prospective to use the CORBA in information integration for the modern enterprises.

2.2 Architecture

Integration platform is an easy to use and maintenance software platform based on network and database. It supports heterogeneous and distributed environment and integrates information seamlessly. The main objective of integration platform is to achieve information share and application integration in CIMS. Applying agent and CORBA, the architecture of the application integration platform can be constructed as shown in figure 1. This architecture consists of three levels from bottom to top, including infrastructure level, integration level, and application level.

At the bottom of figure 1 is infrastructure level. It is a heterogeneous and distributed computing environment.

At the middle of figure 1 is integration level. CORBA provides common object system services (such as naming, events, persistence, concurrency, etc.) and common facilities (such as user interface, information management and exchange, system management, task management, etc.). Because CORBA provides implementation transparency allowing a client to access an object through its interface defined in IDL, independent of the hardware and software environment in which the object resides. This solves the platform heterogeneity problem. On the other hand, CORBA provides location transparency that allows clients to access objects using their object identifiers independent of their location and communication protocols between the client and the object. This property of CORBA solves the communication level heterogeneity problem.

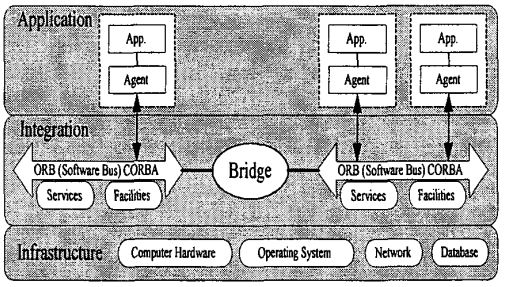

Figure 1 The architecture of the application integration platform

At the top of figure 1 is application level, which consists of application in different domains, such as product and process development domain, business operation domain, management and office automation domain, and monitor and control domain. Each of them is presented by a corresponding agent used to charge interactions with outsides. Such an agent is an autonomous software component encapsulated as a distributed CORBA manufacturing object, then all of these CORBA objects are regarded as a collection of CORBA application objects. The agents in this architecture are linked up physically each other by networking in the distributed form, thereby they can be intercommunicated by passing messages in a common language, an agent communication protocols. Due to the ability of providing us with the basic mechanisms and many services that we need, the CORBA distributed-object standard is as the basis for agent communication. Moreover, in this level, each distributed CORBA manufacturing object, i.e. an agent on the behalf of a manufacturing sub-system, has an interface or multiple interfaces specified in OMG Interface Definition Language (OMG IDL) that is purely a descriptive language used to describe the interfaces to CORBA-compliant distributed objects. An interface, a description of a set of possible operations and attributes with regard to a CORBA object, is the only way by which any CORBA object interacts with others. A CORBA object satisfies an interface if it can be specified as the target object in each potential request described by the interface. In manufacturing enterprise, some legacy codes wrapped by defining interface with IDL may be well integrated in the platform architecture.

3. INTEGRATION PROCEDURES

The integration platform in heterogeneous environment is to integrate different network and its protocols, integrate varied databases, and integrate various applications. The implementing procedure of the information integration platform is divided as below: integration of network, integration of database, and integration of application.

3.1 Integrating Network

With the explosion of World Wide Web (WWW) and Internet, the concept of Intranet, Extranet, and Infranet was born. In fact, Infranet is the short for infrastructure networks, i.e. control network. In this connection, some new manufacturing systems based on Internet/Intranet/Extranet/Infranet have been developed. Internet is essential to implement global manufacturing and network manufacturing. Intranet is the mainstream in enterprise informatization. Extranet built between enterprises is used to realize virtual enterprise. Infranet is utilized to achieve a blend of control network and data network. Because the integration of information is divided into two kinds: integration in an enterprise within an Intranet, and integration between enterprises on the Extranet. Hence, it is needed to support Internet/Intranet/ Extranet/Infranet.

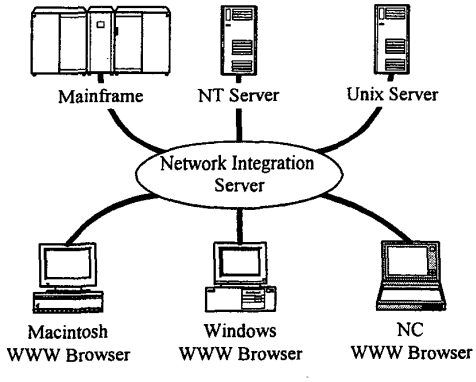

Figure 2 Network integration server

As we known, it is common that there are monolithic mainframe, client/server, and browser/ server systems in an enterprise environment. In order to solve this problem, network integration server can be used to integrate them into an integrated network, bringing each advantages of them into play, overcome each disadvantages, and at last provide uniform and consistent access way. This can be described in figure 2. The CORBA is the core to network integration server. And CORBA is used to process the transparency of network and support heterogeneous distributed environment.

3.2 Integrating Database

In integrating database implementation, the main problem is the heterogeneity that basically exists at four levels: platform level, communication level, database system level, and the semantic level. Database systems reside on different hardware and operating systems and use different communication protocols. CORBA solves the platform and the communication level heterogeneity problem. The third level of heterogeneity is among the database management systems based on different data models and query languages. Finally, semantic conflicts are likely to be present among independently designed databases. This includes schema conflicts and data conflicts. The last two levels of heterogeneity in integrating databases on a platform can be handled by developing a generic database agent which has been implemented as a CORBA wrapper communicating with other CORBA objects. It includes: adapter, knowledge base, query processor, and local database interfaces, as shown in figure 3.

The adapter is a standard interface to be responsible for interaction with other CORBA object through an ORB, and dispatching lower-level operation to query processor. A client knows only the interface definitions of a generic database agent.

The knowledge base contains the essential data and knowledge required by the generic database agent to perform its activities.

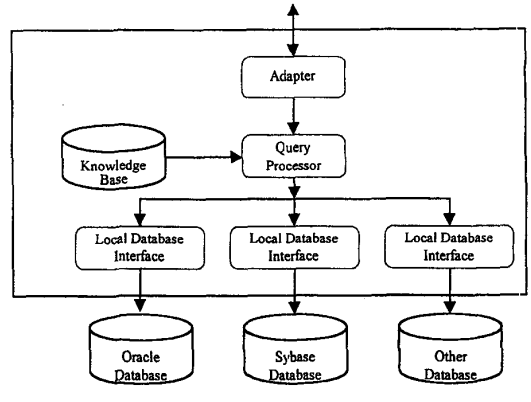

Figure 3 A generic database agent

The query processor is responsible for parsing and decomposing the information queries obtained from adapter according to knowledge in the knowledge base. A global query is decomposed, the global sub-queries are sent to the involved local database interface objects. To implement decomposing the queries, Microsoft's Open Database Connectivity

(ODBC) and Java Database Connectivity (JDBC) are used.

Local database interfaces would provide access to different database management systems (DBMS). Each of the local DBMS has a local database interface. It provides an interface to the local DBM and is responsible for maintaining export schemas provided by the local DBMS represented in the canonical data model, translating the queries received with ODBC/ JDBC to the local query language, and implementing operation for the local DBMS. The derived features of the local database interface depend on the responsibilities fixed for the kinds of database.

3.3 Integrating Application

For implementing application integration, firstly, the manufacturing system function is decomposed. The complex function should be performed by several coordinating application agents. An application agent is an autonomous software component encapsulated as a distributed CORBA manufacturing object, then all of these CORBA objects are regarded as a collection of CORBA application objects. Secondly, agent model is established. From software engineering point of view, agents can be thought of as a natural extension to object-oriented programming. Thereby, besides being built the entity model, the dynamic model, and function model of agent according to object-oriented system modeling methodology, the message flow and coordinative model between agents must be constructed. After the modeling step, agents can be coded in CORBA/IDL manually or automatically to formalize the CORBA object interface. Note that only those attributes/operations of several agents that are allowed to be accessed/invoked by the outside clients need to be defined in the IDL. Lastly, the application agents are developed and are plugged into the proposed integration platform architecture, then integrated through the underlying CORBA system. In this step, the task consists of generating the stub and skeleton from the IDL description, building the application agent adapter, and developing the client and the object implementation. The application integration is realized through these coordination application agents.

3.4 Integrating Legacy System

In CIM, it may be necessary to integrate legacy system. CORBA helps to combine many kids of distributed services that are prepared in CORBA into their application and import legacy applications by wrapping the input and output with a CORBA interface. This can be done through the encapsulation of these systems into several business objects. The key to integrating legacy system is to build the object model with well-defined interface that has the same semantics of the original system. Not all of the application objects should be encapsulated as CORBA objects. Only those application objects that share and exchange data with other applications need to be encapsulated. Therefore, we only concern about the outside interface and format of application. A wrapper is presented the interface of a legacy application as a CORBA object. Figure 4 shows the wrapper architecture of legacy system.

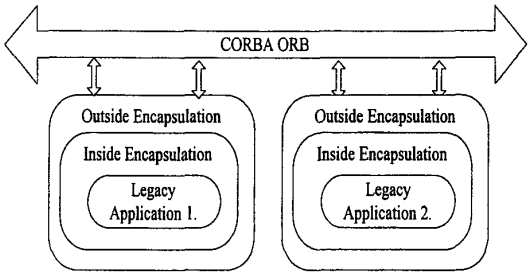

Figure 4 The wrapper of legacy application

The wrapper is composed of outside and inside encapsulation. Outside encapsulation is closer to the ORB. It accesses data and requests operation through inside encapsulation. In addition, it is responsible for translate the operation request of client into the method invocations. A wrapper enables applications that do not have CORBA interfaces to communicate with other distributed components that have CORBA-compliant interfaces. For example, we had a simulator of shop floor scheduling system. Its codes are written in FORTRAN. It doesn't execute because the I/O required an old graphics terminal. CORBA provided a solution to make this simulator code accessible again, without having to rewrite the code. First, any I/O from the source code will be stripped out, and created a library, which could be called from C or C++ code. Second, a wrapper is defined which provides functionality for putting data into the simulator code, executing any shop floor scheduling functions, and extracting data from the simulator code. This could be C++, Java, or Smalltalk. Last, the wrapper will be converted into code "stubs" by an IDL compiler. The output of the IDL compiler will be dependent on the programming language in which the wrapper is to be written. By being used the wrapper, this simulator component then becomes available over the network to clients developed in many ways. A new graphical user interface to the code can be developed easily without changing the engineering functionality in the simulator code. These legacy systems are integrated in the new architecture of distributed agents, and will be reused, encapsulated

and transformed into a distributed architecture so as to implement information integration.

4. CONCLUSIONS

In this paper, a new information integration platform based on agent technology and CORBA is presented. In order to implement information integration for CIMS, information integration platforms provide services for sharing data and synchronizing the operation of applications. It hides the heterogeneity of hardware or operating systems, offers simple application programming interfaces and hides the diversity of communication protocols used to access data, supports for the integration of legacy system. Moreover, the platform is open, distributed, and modular to enable the user to adapt the content of the platform to his requirement. Therefore, the development of information integration platform satisfies the requirements of agile manufacturing: rapid response to changing requirement; reduction in both time and cost of the product realization process, and integration within a heterogeneous, wide-area networked enterprise.

REFERENCES

[1] H. Chao, M. Jung and M. Kim, "Enabling technologies of agile manufacturing and its related activities in Korea," **Computers and Industrial Engineering,** vol. 30, no. 3, pp. 323-334, 1996.

[2] K. H. Kim, "A negotiation based scheduling for items with flexible process plans," **Computers & Industrial Engineering,** vol. 33, no. 3-4, pp. 785-788, 1997.

[3] G. Rzevski, "A framework for designing intelligent manufacturing system," **Computer in Industry,** vol. 32, pp. 211-219, 1997.

[4] P. Gu, "Bidding-based process planning and scheduling in a multi-agent system," **Computers & Industrial Engineering,** vol. 32, no. 2, pp. 477-496, 1997.

[5] R. Sikora, "Coordination mechanisms for multi-agent manufacturing systems: application to integrated manufacturing scheduling," **IEEE Transaction on Engineering Management,** vol. 44, no. 2, pp. 175-187, 1997.

[6] R. A. Whiteside, C. M. Pancerella and P. A. Klevgard, "A CORBA-based manufacturing environment," **Proceedings of the Thirtieth Hwaii International Conference on System Sciences,** Volume: 1, pp. 34-43, 1997.

[7] E. Niemela and M. Holappa, "Experiences with the use of CORBA," **Proceedings of the 24th Euromicro Conference,** Volume: 2, pp. 989–996, 1998.

[8] M. Lei, X. H. Yang, M. M. Tseng and S. Z. Yang, "A CORBA-based agent-driven design for distributed intelligent manufacturing systems," **Journal of Intelligent Manufacturing**, vol. 9, no. 5, pp. 457-465, 1998

[9] C. Paolini and M. Vuskovic, "Integration of a robotics laboratory using CORBA," **Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics: Computational Cybernetics and Simulation,** Volume: 2, pp. 1775 –1780, 1997.

[10] R. M. Soley and C. M. Stone, **Object management architecture guide.** Third edition, Wiley, NY, 1995.

 0-7803-6652-2/2000/$10.00©2000 IEEE