Tsinghua Science and Technology, Vol. 4, No. 1, March 1999, pp.1282-1286

CORBA Based CIMS Application Integration

SHI Wei（石伟）, WU Cheng（吴澄）, FAN Yushun（范玉顺）

National CIMS Engineering Research Center,
Department of Automation, Tsinghua University, Beijing, 100084, China

CORBA(Common Object Request Broker Architecture) provides the framework and the mechanics for operation in distributed object environment. It can also be applied in CIMS application integration. In this paper the application of CORBA in CIM information environment is studied. The service requirement of the CIMS information environment is examined and the integration paradigm is proposed: the CORBA based CIM information environment architecture, application integration techniques, and the relation of CORBA to CIM application integration platform.

Keywords: CIMS, CORBA, Application Integration, Integration Platform

The distributed object computing technology has set the ground for inter-operation between objects in heterogeneous distributed systems. It also introduces a new way for CIM information systems implementation. Distributed object computing can satisfy these basic features of CIMS information system as the unification of the heterogeneous systems and the interfaces standardization. It can also be helpful to the systematically software component reuse.

The distributed object computing technologies now under extensive study include CORBA (Common Object Request Broker Architecture)[1] proposed by OMG (Object Management Group), and DCOM (Distributed Component Object Model) proposed by Microsoft Corporation. From technical point of view, CORBA is of advantage because it focuses in the general function interoperability. Under the OMG/OMA (Object Management Architecture) reference model[2], the application object sends its request for service to the ORB(Object Request Broker), ORB locates and invoke the appropriate object services, then return the operation result to the requester. The whole process is transparent on the supporting computing platform (OS, Network).

This paper discusses how to apply the CORBA technology to CIM application integration. In Section 1 we briefly discuss CORBA and its object-operation mechanism. Next we summarize the requirement of object-oriented heterogeneous distributed CIM information systems. Section 3 studies in detail how to apply CORBA to CIM integration. We present a CIM information system architecture based on CORBA and the implementation paradigm, we also discuss the relation between CORBA and the application integration platform. At last we mention the insufficiency of current CORBA technology for its application in CIM environment.

Manuscript received：1997-8-29

Supported by the National 863 High Technology Plan

1 Description of OMA and CORBA

Object Management Architecture[2] is proposed by OMG as its reference model for distributed object computing. It consists of the Object Request Broker, the Application Objects, the Object Service, and the Common Facility. Application Objects are collections of objects used to construct application systems. ORB is the agent for service request./response. It receives the object request, and invokes the appropriate server. After the server finishes the appropriate object operation, ORB returns the result to the requester. The introducing of ORB enables objects to send requests and provide services independent of platform, location, and language, so that they can coordinate in a heterogeneous distributed environment to finish the assigned task. ORB is the key component for CORBA operation. Object Service is a set of objects that utilize low level platform services to provide common services needed by most distributed object. Common Facility is a group of service provided at the application level, usually oriented to application domains.

Now we illustrate the structure of CORBA in Figure 1[1]:

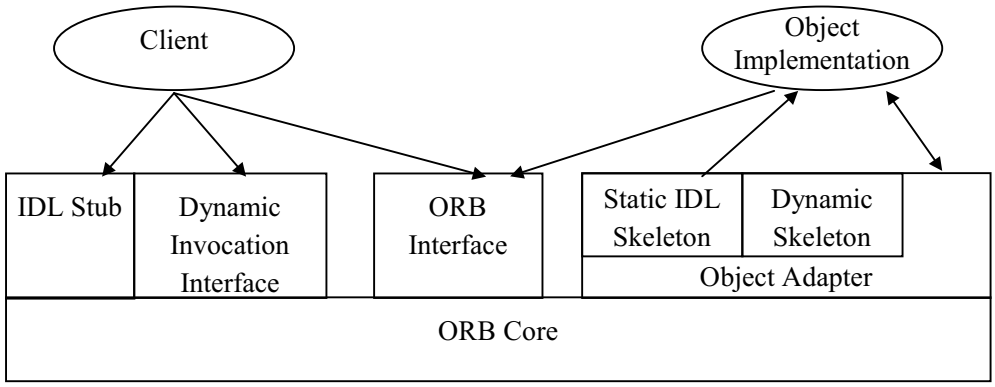

Figure 1 CORBA/ORB Structure

Clients send service request to object implementation though ORB.

Object Implementation is the server which provides service. It contains the data and the code for finishing the object service.

ORB is in response for locating the object and invoking the object implementation through Object Adapter.

Dynamic Invocation Interface provides the client with the method for constructing request dynamically. It is suitable for requesting unknown services.

Object Adapter finishes the invocation of object implementation. It passes the request to object implementation, and present object implementation with general ORB services.

IDL Stub provides the client with the static interface for object service access.

Static IDL Skeleton provides ORB with the static interface for object service invocation.

Dynamic Skeleton is the interface for dynamic object invocation.

CORBA/IDL (Interface Definition Language) is CORBA's standard definition language independent of program language. It is used to define the object request/service.

In CORBA system, there is a dynamic client/server relationship between objects. An object can be a client, a server, or both. In the operating session, the client and the server do not depend on each other, they only interact with ORB to request or provide service.

CORBA provides feasible solutions for application integration. By introducing ORB, CORBA

realizes application interoperability in heterogeneous environment, which allows object handling transparently to location and supporting platform. At the same time, the interface definition of CORBA provides mechanics for unifying the application software interface. This will greatly facilitate the integration of various software tools. In the following sections we will study how to apply CORBA to CIM information environment. We start from the function requirement of the Object-Oriented CIM information system.

2 Function Requirement of Distributed O-O CIM Information System

To meet the application requests, an O-O CIM information system must satisfy the requirement from three aspects: distributed processing, development and integration of application objects, and the support to CIM domains. From these aspects, we present the function requirement of an O-O CIM information system.

Supporting distributed processing is the basic requirement of a CIM information system. This includes:

1) supporting decomposition of functions, distributing them to several application objects,
2) object handling across different platforms (OS, Network, DBMS),
3) valid system management in distributed environment,
4) satisfactory quality of performance in distributed situation, such as response time, reliability, etc..

One of the major goal of the information system is the enabling of application development and integration. The goal raises requirements in both the development phase and the running phase, presented more accurately:

1) the system must provide standard application interface for application integration, the interface should be flexible,
2) powerful modeling and developing tools for application object building,
3) support for the integration of legacy information systems.

The orientation to CIM function domains will determine whether the information system will be convenient to use in CIM implementation. In O-O framework, the relative requirements are:

1) provide O-O data access mechanics, this is an urgent need from production data management,
2) provide mechanics for semantics integration, applications need to access data through semantics corresponding to their specific needs, not computer oriented bit flows,
3) provide workflow management services for process management and integration,
4) provide standard services for specific manufacturing domains, for example through the means of OMA Object Service or Common Facility.

3 CIM Application Integration Based on CORBA

Now we discuss how to integrate CIM applications through CORBA technology. We will present the integration architecture, the integration process, and the relevant techniques.

3.1 Integration Technology

From above discussion of CORBA architecture and CIM information environment requirement, we present our CIM integration methodology based on CORBA. First we give the integration architecture, as illustrated in Figure 2.

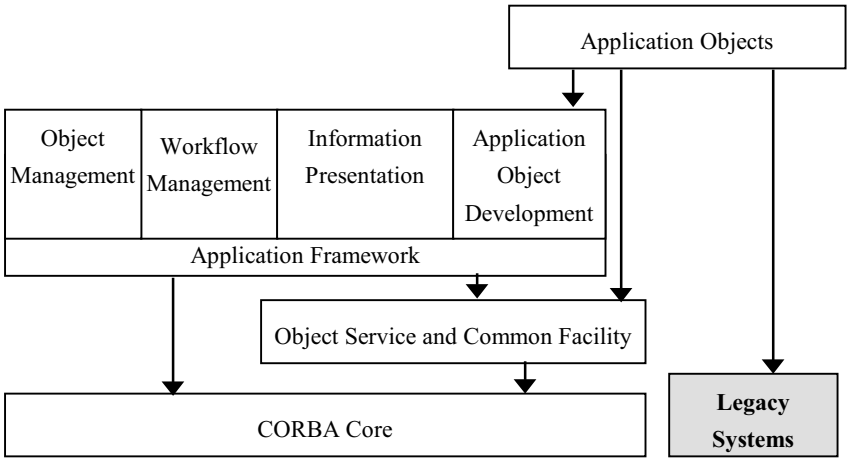

Figure 2 CORBA based CIM Application Integration Architecture

Here, CORBA core is the supporting layer of the whole architecture. It provides the basic object interoperability. Object Service and Common Facility are general system services similar as in OMA architecture. They can be accessed through object request. On the ground of CORBA Core and general service, we define the application framework, which provides such general application integration mechanism and tools as object management, workflow management, information presentation, application object development tools, etc.. Application framework acts as the supporting architecture for application development, integration and running. On the top are the application objects. They utilize the services provided by the Object Service and the Common Facility and the integration mechanism provided by the application framework. The system integration is realized through these coordinated application objects. Application Objects can also be used as to build interfaces for legacy system integration.

Next we talk about the system development process under such an architecture. The whole processes take several steps: O-O system modeling, CORBA/IDL presentation, object development and/or integration. O-O System modeling builds the entity model, the dynamic model, and the function model. There are several mature methodologies for O-O modeling, such as OMT, Booch Method. We would only mention that the model should be complete to include both the system in-built and the system existed. After the modeling step, the model can be coded into CORBA/IDL through manual or automatic work. This step formalizes the object interface. The third step is the development of the application objects. The task consists of generating stub and skeleton from IDL description, building application object adapter, developing the client and the object implementation. Application components are built and tied together through object interfaces in this step. The last step is fixing these components into the integration architecture, and integrating them through the underlying CORBA system.

We discuss several implementation technologies related to the developing process.

1) The decomposition of system function, complex system functions should be performed by the coordination of several application objects.
2) Interface description, when using IDL to describe object interfaces, the developer should be aware to separate the interface description from its implementation. The implementation detail should not be implied in the description. Apart from the static semantics provided by IDL, sometimes it is also necessary to build dynamic semantics for the interface invocation.

3) Object development. The implement policy for distributed objects should be carefully selected to ensure the feasibility of the system. Such policies as choosing appropriate object granularity and local caching will improve system performance significantly[3]. The tying of application to object implementation should be consistent in semantics.

4) For the integration of legacy systems, the key is to build its object model with clearly defined interface, with no semantic conflicts to the original system.

5) Use standard services such as those defined in Object Services and Common Facility as possible. This is helpful to ensure performance and consistence.

3.2 Services Specific to CIM Information Environment

As mentioned above, an integration supporting environment should provide a set of commonly used services to facilitate application development and integration. These services can be classified into two layers: the system layer and the application layer. System layer services are general services that are used by most application objects, such as message service, naming service. Application layer services are high level services serve to more specific usage. Seeing from OMA reference model, these two layers correspond to the OMA Object Service and Common Facility. OMG has defined some Object Service and Common Facility, but they are not abundant enough to support CIM application completely. Here we discuss some services that are required by CIM environment.

CIM information environment is inherently distributed. It must provide general mechanics for communication between distributed objects. Services needed here are naming service, message service and access control. OMG has defined naming service, event service, concurrency control service in Object Service Specification, and message service in Common Facility. In CIM environment, we also need to handle real-time object to support time critical tasks. The naming service should also be extended, and specific name graphs should be presented to meet various naming requirements in manufacturing environment.

An ideal information system should provide transparent and reliable data sharing in a global meaning. The corresponding services include persistent object supporting, distributed object duplication, concurrency control, transaction service, etc.. OMG has defined these services in Object service and Common facility. To match the current database technology availability, the interface to RDB (similar to ODBC) is also needed. The application layer service relevant to global data access are information entity modeling, data storage and retrieving, data encoding and presentation, etc..

For the control of activity and process, such services as transaction management, object status keeping, consistent time service, etc., are needed. These are services in system layer.

The information system should provide services for various domains of CIM. The typical services include: STEP data access interface for production data management, MMS/VMD service for device operation, compound document management, EDI relevant services, etc.

3.3 CORBA and Application Integration Platform

Introduction of Application Integration Platform can be found in [4]. Comparing of Application Integration Platform with CORBA environment shows that they have many features in common:

1) the communication service provided by Application Integration Platform is equivalent to CORBA/ORB, their all provide inter-operation mechanism in heterogeneous environment;

2) Application Integration Platform and CORBA system all provide unique interface format for the construction and integration of application/application objects.

3) Application Integration Platform has the Application-Dependent APIs and Application-Independent APIs that have the similar functions as CORBA Object Service and Common

Facility.

On the other hand, they have much difference. Application Integration Platform mainly provides the application domain oriented service and tools, but CORBA focus mainly on general mechanics and services. Some significant differences are:

1) Application Integration Platform provides a unique information access interface GIS (Global Information Service), which is not provided by CORBA.
2) Application Integration Platform provides application domain tools.
3) As a running support platform, Application Integration Platform has complete system management policy and tools, but CORBA provides only mechanics and some services.

Generally speaking, CORBA provides kernel services for interoperability such as ORB, BOA, etc., but places high level services in Object Services or Common Facility, which will be supplied from outside system. So it is possible to develop part or all of the Application Integration Platform modules based on CORBA. This strategy has its advantage, such as the easy implementation of application communication service, the uniqueness of interface, etc., but it also has problems in global data integration and in system performance improvement.

4 Conclusion

CORBA has its instinct advantage in system integration. It can solve the problem of the unification of the heterogeneous systems and the interfaces standardization. From the whole life cycle view, CORBA is totally open, which means that for a system implemented on CORBA, future software components conforming to CORBA specifications can be inserted into it easily. This is of great help to system updating and expansion.

At current phase, there are still some problems that obstruct CORBA application in CIM information systems. These problems are:

1) CIM domain services and components are not abundant. More reusable components mean more efficient development process and more reliable application systems.
2) CORBA supporting environments are not extensively available for practical use. Also there is the competition of CORBA and the DCOM in distributed object computing area.
3) There are problems about interoperability between different ORBs. Through OMG presented the CORBA/IIOP mechanism, the practical experience is absent.

Solving of these problems will lead to more acknowledgment and adoption of CORBA in CIM information systems.

References

1 Object Management Group, *The Common Object Request Broker: Architecture and Specification*, Vision 2.0, 1995

2 Object Management Group, *Object Management Architecture Guide, Version 3.0,* June 1995

3 James R. Kirley, William G. Nichols, *Integrating Applications with Digital's Framework-based Environment*, Digital Technical Journal, Vol.7, No.2, 1995

4 MACIP Group, State CIMS-ERC of China, *Manufacturing CIM application integration Platform, General Design and prototype development*, Research Report. 国家 CIMS 工程研究中心集成平台项目组，“制造业 CIMS 应用集成平台总体设计与原型系统开发”项目可行性论证报告，1997