2011 Fourth International Joint Conference on Computational Sciences and Optimization

# Software Reuse and Distributed Object Technology

Xichen Wang
School of Computing,
Communication University of China,
Beijing, 100024, China
E-mail: wxc@cuc.edu.cn

Luzi Wang
Department of Mathematical Sciences,
Ball State University,
Indiana, 47304, US
E-mail: wangluzi_1216@yahoo.com.cn

***Abstract*—Distributed Object Technology employs object resource, which distributes on the website joint points, to handle data, following by the expanding applied range, the development process of distributed applied software are more and more complex. It is the software reuse that is one of the most important ways to reduce complexity of software development, as a result, much overlapping happened between distributed object and reuse technologies, moreover, distributed component concept is the consequence of the two mutual promoting and combining, this paper will discuss this problem by CORBA standard.**

***Keywords- Distributed object; software reuse; component***

## I. INTRODUCTION

Software Reuse means the process that use "designed software for reuse" again and again. By reusing, we can control complexity of software development, shorten the circle, advance quality of produce. According to the various models of software development, the reuse methods are not all the same, in which, the component-oriented reuse is the mainline technology universally certified by academia and industry circles, comparing with other, it is more available and practical.

Distributed object technology mainly handles interoperability in heterogeneous environment, with the increasingly mature of middleware technology, high level development of distributed object is more and more important, consequently, the introduction of component concept is nature, which breaks the status that only senior programmers can compile distributed implicated programs. Due to the introduction of middleware, moreover, the individuality among components is stronger, and supplies heavy support for component assembly, especially the operating-degree. Currently, this kind of research is very actively around the world, each company and research institution has large investment, CWC (Component Ware Consortium), which is a special organizations already established, use for studying the common problems together

After 1990s, internet technology improves rapidly, and object-oriented technology language is increasingly mature, in addition to their internal complementary and coordination, promote distributed object technology. The distributed object technology researches how the different joint points coordinate, and accomplish the special job together which distributing on website. The core of distributed object technology is the interoperation among the objects, especially in heterogeneous environment, by the support of middleware, the physical dispersed computing resource logically become a whole. The key problem of interoperability is to make a standard, which is independent from hardware platform, operation system and program language, two standards exist now: CORBA Standard of OMG Organization and DCOM Standard of Microsoft. CORBA (Common Object Request Broker Architecture) is the public structure of object request broker (ORM) defined by OMG, base on OMA (Object Management Architecture)[1]. ORB is a object-oriented middleware, it effectively separates the two interoperating parties: customer and server objects only comply with common interface defined by IDL (Interface Definition Language), and customer object can transparently active a remote server object, like active the local one, it's nothing to do with the object position, program language, detailed operation system, or other information irrelevant to object interface. The details of CORBA refer to figure 1.

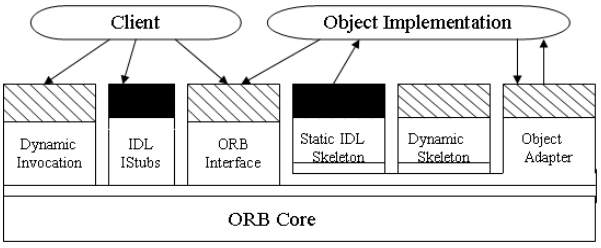


Figure 1 CORBA Structure

COM is the component standard and realization supplied by Microsoft. Supported by COM communication mechanism, the developers can assemble components provided of different manufactures, the corn of COM is a set of Application Program Interface (API), which supplies the method of building component and team-training component. DCMO is the result of expanding COM for Microsoft supporting website environment, whose target is familiar with CORBA, they support the interoperation among objects realized by different joint points, different operation systems, and different realizing languages. Contrasting to ORB, the bottom mechanism of COM can only work at the phase when customer build request relationship to help orienting server object's location, while the special adjusting, reflecting processes directly work between customer object and server object[2].

978-0-7695-4335-2/11 $26.00 © 2011 IEEE
DOI 10.1109/CSO.2011.243

## II. DISTRIBUTED OBJECT-ORIENTED COMPONENT TECHNOLOGY

Improvement of complexity of applied program needs us to abstract object system on a higher level, and reuse the past design achievement on some larger extent. Component technology satisfies the mentioned request very well. Because CORBA can perform distributed object technology better, the paper will discuss below combined by CORBA standard [3].

### *A. .Component Feature*

Component is the software entity for reuse. In according to the different reuse phases and reuse methods, the corresponding performances are different, for instance, analysis file, detailed design, code realization, etc [4]. In distributed system, the initial identification of component is the program units, which distributes on each joint point, mutual individual operation, as well as mutual cooperation to accomplish certain target. Any operation between the two components is processed by customer/server. The one who sends the request is customer, another one who accepts, implies and handles is the server. The two are mutual individual, only when customer sends the request during operation, customer and server will connect. From the customer's perspective, as a result, the server is already existed, because we need the certain special function of server, which is named customer "reuse" server, they have to connect with server. Take the widely recognize three-level customer/server structure (see figure 2) as an example, it is composited by final customer program, applied server and data server, they accomplish the special job designed by customers together.

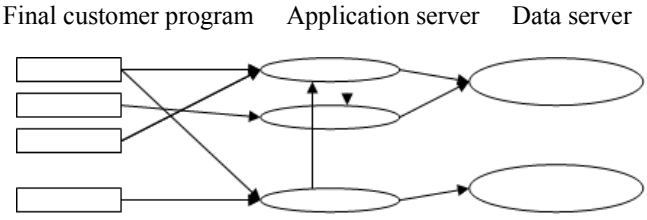


Figure 2 Three-level Customer/Server Structure

From the above analysis, it is not hard to see that most of the distributed components are object code components, which don't need to be modified and complied when we reuse this kind of components, as a result, the reuse method is the most directly method. Actually, it is the past way that application program did for operation system, compiling system. Reuse people cannot open them by this kind of black-box using method, as well as modify by their own demands, however, it is still the most important component in the supported components, COM, JavaBean support it also. Comparing with COM, JavaBean, however, the distributed components put more attention on the connection among components. COM, JavaBean connect with static object codes at building phase, which is suitable for the local components, but not distributed components, because they locale at different processors. Besides adopting common static method to build, connection among distributed components can be built during operating, that is, only when customer components actively send request to server, it build connection relationship. The connected components can be static, and droved by correspondingly adaptors when reuse people connect it; they can be dynamic, namely, operating all the time, this kind allows reuse people to designate share-based reuse or private-based. Under the share-based, different reuse people share one component entity, among which even can build relationship by reused people; under the private condition, when different reuse people connect component, they need to clone one operating component, the consequence is that many reuse people corresponds to different components respectively.

### *B. Component Description*

A component must provide information about oneself to reuse people. Because the target of CORBA initially be the interoperation problems during processing, it only provides the description of operation in IDL of objects, and lack of design process information, for instance, the dependent and interaction relationships among components, etc., the key that distributed components are different from other object-oriented codes is the flexible connection method among components[5]. The following contents are the must if reuse people wanted to finish the use of distributed components:

- Attribution. It means that the components can be directly operated from the outside, and the value can be read and modified. Additionally, in order to keep the consistency of internal and external, the modification of some special attributions can notice to the external as an event of system, and trigger the operation of related components, and then, becoming a organic whole.
- Functional Interface. This is the availably operational method that components transfer to the outside, as well as an identification of components' functions and activities. Obviously, this is the identified content of IDL in CORBA standard.
- Dependence Relationship. This kind of relationship refers to the depending components or interfaces when this kind of component is instantiated, which is the necessary part before components finishing their functions, not only can be other components, but also the special interface of others[6].

The mentioned description of components is usually called the meta-data of components, each one has its own. Components must be browsed, namely, Meta-data of each reused component can be listed by tools, and the reuse people can understand the necessary information. When reuse people get these data, the corresponding component doesn't have to be instantiated, because currently people haven't decided whether use it or not, it is no need to build the example of component at this time. Two methods can be used for storage meta-data: the more direct way is that save it into objected realized description region, as the supplement of it; another one is more stable, that is putting them into a

component description region (CDR) (which is similar with interface region: IR), in order to strengthen the management of meta-data.

The description language of meta-data is Component Description Language (CDL), it is the superset of IDL.

### C. Component Assembly

#### 1) Component Connection

For the source-code component, whose connection method is relatively simple, what you need to do is analyzing the source code first, and adding the call statement of called person at the proper location of call person. Under this kind of condition, however, it is hard to adjust after building the connection relationship, because the connecting information is compiled in the object code, if we want to adjust, it should modify the source code and compile again. On the other side, connections among distributed components must figure out how to add connection information without changing object code, as a result, its connection mechanize is more complex. But once we realize it, the method will become more flexible, especially the strengthening functions of dynamic configuration, user can add new component or replace old one during system operation without affecting the normal running.

For reusing the realization of one object, we usually hope that replace an old realization by a new one. In C language, function pointer can be transferred as a common variable, and the programming people can call different functions, but in C++ and Java languages, they don't supply the corresponding object pointers. The distributed components adopt component interface reference mechanize to realize this function, which is similar with pointer in function, the two can be used for locating operated people, reference, however, supported by other mechanics (such as ORB), can cross computer, operate system and other different operation spaces, to locate components and interfaces exactly.

The target of component connection is to build call relationship among components, which need to fill the reference of called people into calling people. This process is based on the event, the change of component situation of call people can be considered as a kind of event (such as change of attributed value and so on), due to the design demand, if the occurring this event affected other components, it should be spread into other corresponding components, this process is the operation for calling other components. The key to realize the process is a cited list involved in the calling people, for recording reference of each called person. While the components, which want to monitor this event, actively register event source---calling people, and put the own reference into the list, and note the type of event. Its structure refers to Figure 3. When the event happened, calling component check the component which interest to this event from citing list, and callback the corresponding method.

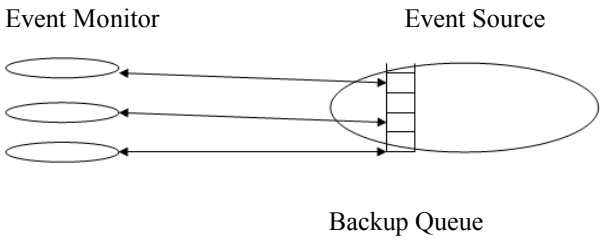


Figure 3 Event-base Component Connection Structure

#### 2) Component Assembly

The target of component reuse is that create operated application program by currently available components, or bigger one. Process of component assembly is adding the current components into a frame, and then, connecting the added components by events. This process not only can reflect a static component design behavior, but also a dynamic component configuration behavior.

Building connection among components is the important process of component assembly, we have introduced its basic principle briefly before. During actual assembly process, due to the existing layer structure, it make more complex when spanned different layers, and should be operated by the support of locating function of involving people. Additionally, the types of corresponding interface of event source and event monitor may have some certain inconsistencies, we usually introduce an adapter to make necessary coordination for the two parties, and the communication will continue. Adapter can be responsible for differentiate trigger methods of event as well, for the registered monitors of one event, there are many trigger methods: event source triggers all the monitors at the same time, or only one monitor at one time, or several.

The feature of mentioned component assembly mode is that use callback list and inlay different components together, it is what we call inlaying assembly mode, moreover, there still has another bonding one, it adopts script language to assemble various components. Script description can bond different components, which not only can call input way of components randomly, but also can be triggered by happening events of components. Because script compiling is open and more flexible, we can adopt bonding assembly mode to complete assembly among components when inlaying assembly mode will be hard to realize or need to pay a higher price.

#### 3) Component Assembly Tool

Component assembly tool is an important part of component technology, and has functions of browsing components, graphic components and graphic operation components, and other:

*a)* Component Browsing Assembly tool must browse the information of reuse component at first, including component level, component attribution, component distribution and so on. This kind of information is usually saved in component region, and according to design demands, in which, the developer find the satisfied components. This kind of demand may be a functional one, use another word, the key is on some certain attribution of

component. For the satisfied components, we can instantiate it complying with distributed information of component, and express it in assembly space. Because the dependent relationship may exist among components, we can instantiate its dependent components first, and then, the original follows.

*b)* Graphic expression of component. Although most of the distributed components are invisible during running, and in order to operate, we should express it by graphic methods in component assembly space when design. The graphical expression of components means the same with text expression (CDL description) of components, however, the graphical can focus on the most important information of developers concerning due to the demands, and avoid the unimportant ones, which is helpful to reduce the complexity of development process.

Another advantage of graphical components is that it can adopt "pull", "push" types to install component into another component of including person, and finish the including operation among instantiations at the same time with pull and push processes.

*c)* Component Relationship Event relationship is another key content of component assembly, event source and monitor connect events by a directed line, event source reflects a certain attribution of component, and monitor does certain operation of another components. While it finishes, event monitor will complete registration operation from oneself to events.

## III. CONCLUSION

Distributed component is a special type, which should comply with common component model, but expand their content in some aspects, such as the dynamic connection and so on, consequently, this kind of research will promote the study of component skill. On the other hand, the rapid development of internet skill advance the demands of distributed software, and the internet software development which is based on distributed components is consider to be a promising direction.

## ACKNOWLEDGMENT

The paper is quality demonstrational program and supported by 211 main project of Communication University of China, project No: 21103050115.

## REFERENCE

[1] The Common Object Request Broker: Architecture and Specification, Revision 2.2, OMG Document Feb.1998.

[2] Robert Orfali, dau Harkey, Jeri Edwards, The Essential Distributed Objects Survival Guide, John Wiley&Son, Inc.

[3] CORBA Components, OMG Draft Document, orbos/97-11-24.

[4] Wang Qianxiang et al, Distributed Sharing Objects Semantic and Realization, Nanjing University Journal, 1995.10

[5] Pan Feng Application Research of CORBA-based Distributed Object Technology, Wuhan University of Technology, 2002

[6] S. X. Guo etc, Credible research of Software Components,On Computer Science 2007,34 (5) .