Advances in Engineering, volume 126
5th International Conference on Machinery, Materials and Computing Technology (ICMMCT 2017)

Analysis on Software Bus Architecture of Distributed Computer

Rui Wang
Liaoning Jianzhu Vocational College, Liaoyang, Liaoning, 111000

Abstract

Now the system is getting bigger and bigger, a system requires a lot of functions to complete, so software reuse and integration has an unusual significance. Component model is a new generation of software development logo. In order to improve software productivity, not hastily developing applications, developers should, as far as possible reusable software components, assembly and construction of new applications. However, component-based software development has the shortcomings of single-machine, not suited to the development of network. The distributed computer software bus architecture proposed in this paper is based on the development of components. The single-machine architecture of component development has been improved, and a new architecture model is designed for network development.

Keywords: Distributed Computer Software, Bus Architecture, Network Technology

1 Introduction

Traditional methods of application development software are often an independent holistic system. After the needs of the prior analysis and design of the software developed, its various functions and characteristics linked together in a fixed manner. However, many of these features can not be removed, upgraded, or replaced independently. For other applications, even with the same programming language, and run on the same machine, it is difficult to use the program's data and functions. Different applications, like strangers, are completely isolated.

In the development of software-based components, in order to reduce the repetitive work, the software components must be implemented in different programming languages. This presents the problem of communication and integration of software components between different language implementations. The solution to this problem can be obtained from the computer hardware bus inspiration.

The hardware bus of the computer system is a common channel between computer components, between input and output devices to transmit data information, ground information and control information. Hardware bus is a group of copper platinum wire as the carrier signal transmission line. By means of the bus, C can exchange information with main memory and input / output devices. In the computer hardware bus, you can plug a variety of types and multi-block universal standard interface module to form a powerful more perfect computer system. Such as a display interface card, a multimedia movie card, a sound card, and a printer interface card. In fact, the hardware bus is a computer CPU and the functional module components and input and output devices are interconnected with a tool. The purpose of discussing the hardware bus is to bring out the new concept of distributed software bus.

Software bus, like the computer hardware bus, is a tool, is a different from the hardware bus software design tools. In fact, the software bus is the hardware bus virtual and mapping, is an abstraction of the hardware bus. The so-called distributed computer software bus is the use of distributed technology, object-oriented for a variety of computer languages written by multiple, multi-type, type of software functional component (object) service a group of virtual data transmission line. This group of virtual data information transmission lines is software, which is a set of standard integrated software functional components that are common to distributed technology. The interface is between the computer operating system and various integrated functional components or integrated software functional components Data transmission and connection of the virtual public channel and interface interface, is a software reuse development tools. Of course, the computer hardware bus and the distributed computer software bus are substantially different.

2 Distributed object technology base

Distributed object technology is a kind of object-oriented technology developed along with the network. Previous computer systems are mostly stand-alone systems, a number of users through the online terminal to access, there is no network concept. After the emergence of the network, drink to the ient / Server computing service model, multiple clients can share the database server and print server and so on. With the further development of the network, many of the software needs of different manufacturers of network products, hardware platforms, network protocol heterogeneous environment to run the scale of the application from the LAN to the development of wide area network. In this situation, the clieot / server model of the limitations also exposed, so middleware

came into being. Middleware in load balancing, connection management and scheduling has played a significant role in the performance of enterprise applications has been greatly improved to meet the needs of key business needs. But at this stage, the client is to request services, the server side is to provide services, their relationship is asymmetric. With the further development of object-oriented technology, the emergence of distributed object technology, It can be said, distributed object technology is with the network and the development of object-oriented technology continues to improve. Early 90s CORBA1.0 standard promulgated, opened a prelude to the calculation of distributed objects.

In a distributed object computation, the computational body (the distribution object) that is normally involved in the calculation is symmetric. A distributed object is sometimes called a component. A component is a wrapper around individual code. It can be a simple object in a distributed computing environment, but in most cases it is a set of related object complexes that provide certain services. Distributed environment, to master the kernel is a number of sensitive software modules, they can be location-transparent, language independent and platform independent of each other to send messages to achieve the request service.

3 Distributed software bus architecture and services

The software bus is transparent to the user and uses the software bus. First, the user application registers its own message (just a string) with the software bus management center: Second, the application can identify any of its known Registration information Sends a message as a parameter, which can be data in any format, the interpretation of which is determined by the application.

On a software bus, if the destination program moves to a network where the application sends a message without having to know the destination end of the message, the message also arrives at the end point where it is located. There is no need to recompile the source code , Re-generation process. In this sense, the software bus on the network for the application to establish a virtual platform.

The overall structure of a common core as the center, the core includes portable object adapter, communication and transmission protocols. The upper part of the structure is the component development workstation, the left part is the component test, calculation, development and management system, and the right side is the component storage system, namely the component storage database. Interface interface is to deal with different language systems or different language components interface between the communication interface, including the IDL compiler (IDLCompiLe). Compiler (ClDI Compile). To the right of the interface interface is the application API. The left side of the common core is the secure interface of the application system and the software bus, utilizing Secure Socket Layer (SSL), and firewall technology. The right side is named service and notification service. The lower part of the common core is the various services of the software bus communication core: object transaction service OTS, continuous state service PSS, value object ObV, software bus management control, mouth

record service) and authentication authorization management. After the application development integrator has developed the system, the qualified system is placed into the application repository.

Universal core to portable object adapter P0A and communication protocol IOP protocol-based. The design of POA 11 is to enable communication between different software busses, to support objects with persistent identities, to provide support for transparent activation of component objects, to implicitly activate servants via the objectID of p0A, to allow component objects Is responsible for maximizing the behavior of the object, allowing the programmer to construct an object implementation inherited from the static program framework class and generated by the IDI compiler. Each component server can have more than one POA, each POA to provide different functions or to support different characteristics. And each pOA provides a separate object to live space, there is a set of POA strategies to determine how these objects are activated and how to build a reference object. The POA strategy is quite complex. However, in component assembly, the soft component service locator strategy is the most scalable POA strategy. The generic oRB interop structure is based on the Generic Inter-ORB Protocol (GIOP), which defines the message format for transport syntax and any connection-oriented ORB interoperability. 110P can achieve communication between components through the software bus.

The goal of the software from the software bus is to create highly integrated, well-integrated component-composite applications, with significantly different tasks for each of the components, and to create a low coupling between components in the system. This requires that the system designer extend the object-oriented principle from the object specification to explicitly represent the object dependency using the interface definition. This specification of software bus and component capabilities can be divided into several interfaces. Partitioning the component specification into multiple interfaces can make the in-component dependencies limited to individual interfaces, rather than the entire component specification. The interface to the software bus is to implement programs written in one language to communicate with other programs written in an unknown language. OMGIDJ always allows you to create inheritance-based object relationships. However, the design needs to support objects that contain multiple interfaces, which are constructed by combining rather than inheriting. Object inheritance allows you to define an implementation of this class in terms of another class, which allows you to define classes by grouping or grouping objects together. OMGIDL needs to express the ability to combine and inherit.

4 Software bus standard software components

Both reusers and component producers need to get the widgets. Component acquisition is mainly combined with re-development and utilization of existing software, in addition to re-development, there are modifications of existing components, re-engineering, reverse engineering and other means. (Which can be used but will not be reused) or overly distributed. Reusable but not easy to use)

Software knowledge is converted to available and reusable entities: the use of existing resources,

The main contents of this paper include reusable component development methodology, component model, component description language, reusable component library conceptual model, reusable component, reusable component library, component reusable component reusable component Library management system, component-based software development process and other related tools and so on. Among them, the software bus research and component model is the foundation and core, unified soft bus standards and components phase, is to achieve the basic premise of application system assembly.

5 Conclusion

The idea proposed in this paper is beneficial to improve the traditional software development in the development cycle is long, high maintenance costs, the application must rely on operating systems and programming languages and other shortcomings and modern component-based software development of the programming language Dependence and the development of stand-alone and other shortcomings. In the application of distributed object technology, it improves the limitation of application of middleware in domestic and abroad. It is the first time that the distributed technology is combined with the development and application of software technology to make the technology of software development more adapt to the development of network technology. The component development method is applied to network development, which improves the efficiency and quality of the software development, improves the software development technology and reduces the development cost.

References

[1] Zhou Yue, Wang Hong. Software bus technology and its application in enterprise ERP. Computer Development & Applications, 3(2), pp. 13–22,2006.
[2] Software bus in the East Alpine. Software Engineer , 4(1), pp. 15–17, 2004.
[3] Sun Zhian, Dou Qiang. Software bus: architecture analysis and design.Command Control and Simulation , 4(1), pp. 13–16, 2001.
[4] Li Jianhong, Zhan Chuanjie. Development of control function module in software bus architecture. Microcomputer Information, 30(2), pp. 16–18, 2009.
[5] Wu Kehe, Song Min, Zhou Jing. A software bus technology based on multi-agent system.Power System Technology, 3(1), pp. 18–31, 2007.