Research on the Realizing Mechanism of PL-ISEE Broker Architecture

Jianli Dong
Huaihai Institute of Technology / School of Computer Engineering, Lianyungang, 222005, P.R.China.
Email: dongjl1019@sina.com

***Abstract*—The research on software product line will face two problems: one is the design of new industrialized PL-ISEE model (that is component based assembly line), another is the implementation of product line core asset and COTS component agent bus. In the paper, a new industrialized PL-ISEE model is firstly proposed by the authors. One of the main parts framed the new PL-ISEE is the core asset and COTS component agent bus. To realize the agent bus and component based software assembly line, the broker idea and architecture based on CORBA are introduced. The broker architecture and basic framework model adapted to the new PL-ISEE component agent bus requirements are designed, and its implementation mechanisms are systemically discussed. The PL-ISEE realized by the broker architecture will have more advantages on the locating transparency, dynamic updates and expansion of the core asset component servers of software product line, system platform independence and portability, interoperability and interactivity between different agent systems. These advantages are very useful to implement the new PL-ISEE and industrialization production of the component based software products.**

***Index Terms*—broker architecture, software component, software product line, PL-ISEE (product line based integrated software engineering environment), CORBA, agent bus**

I. INTRODUCTION

In the heterogeneous network environment, so as to make full use of the extensive network resources and collaboration to resolve the complex distributed computing problems effectively, the researches and applications on software component, software middleware, Internetware, software Agent etc. new technologies were born and have already made a great progress. In order to achieve the interoperability and communication between components and component-based application software development under such a heterogeneous network environment, the agent/broker technique and method have become a research focus of the current heterogeneous networks, distributed computing and communication, software product line and so on fields. This paper will mainly study the framework model and implementation mechanism of software component broker architecture in a new industrialized PL-ISEE (product line based integrated software engineering environment). The aim is to create a feasible theory and technology support for industrial assembling production of application software products in distributed network [1-3].

II. THE PL-ISEE'S BROKER ARCHITECTURE REQUIREMENTS

The research and development of software components, software architecture and software product line have brought new hope for the formation of new software engineering methodology and industrialized production of software products. Software component is a software entity or program entity with specific computing functions, it can be not only deployed and run independently, but also assembled and cooperated with the other components. The software architecture is the whole structure and blueprint of the software system development, in accordance with which, the software system will be assembled and implemented by using the software components. A software product line is an assembly line of the software products satisfied the specific application area requirements and adopted DSSA (domain specific software architecture) as blueprint and interrelated software components (including connectors) as parts. Thus, the software product line is similar to the modern manufacturing industry production line achieving the industrialized assembling production of software products, it is the revolutionary advances and leap in modern software engineering [4-5].

We have proposed a new industrialized PL-ISEE model shown as the Figure 1. This new industrialized PL-ISEE model is a multi-layer architecture model on the basis of unified product line engineering conceptual model, large-granularity reusable asset data model, component assembly behavior model, iterated evolution model of core assets developing and application software producing.

The new industrialized PL-ISEE framework includes essentially a double-development environment with core asset and COTS agent components as the software components bus. On the bus, it is the assembly line of the PL-ISEE based on product line software engineering methodology, and realizes the automated and industrialized assembling production of application software product (clusters). The core asset or COTS components needed by assembly line are provided by the

doi:10.4304/jsw.8.3.579-586

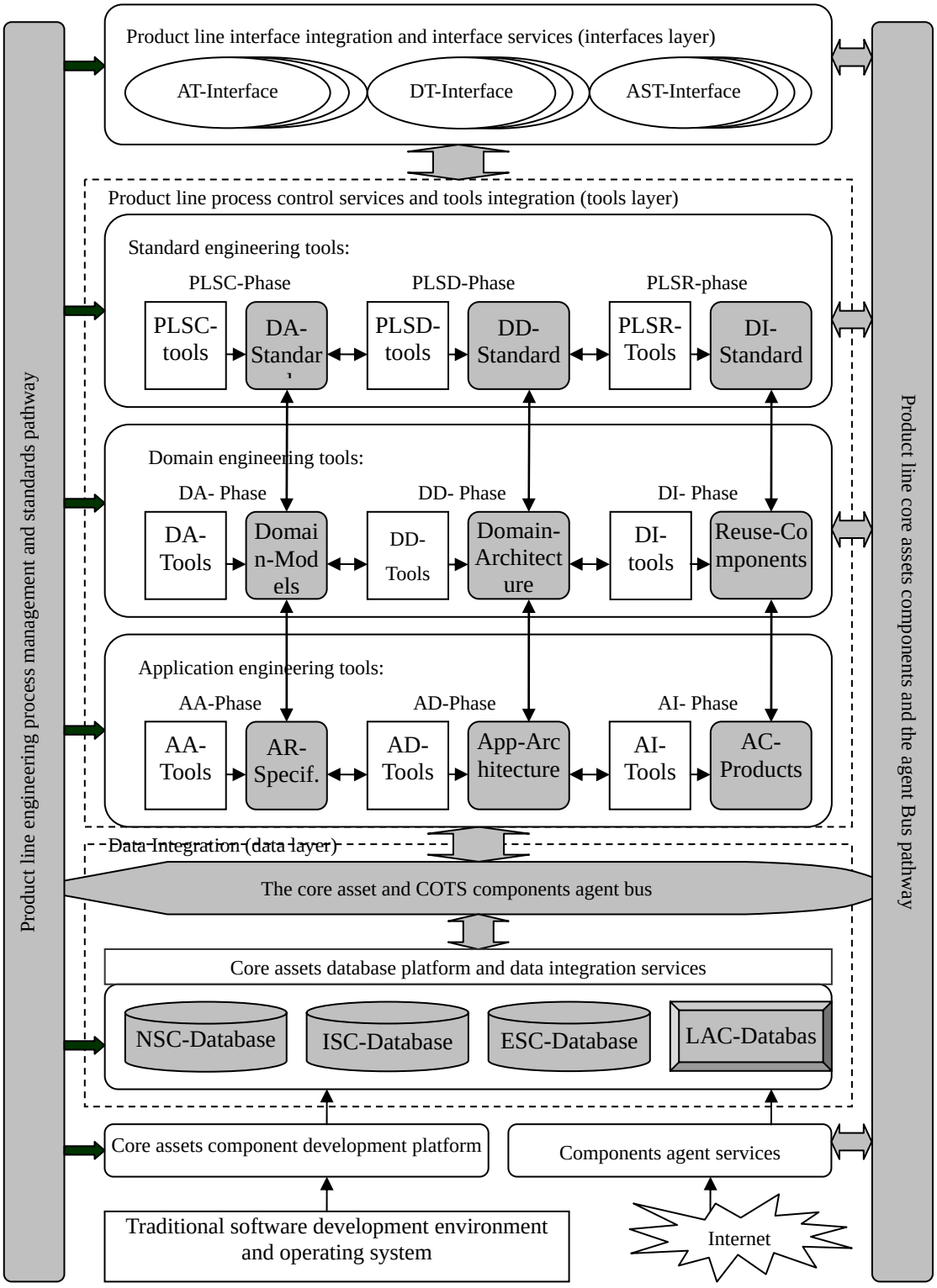

Figure 1. A new industrialized PL-ISEE architecture model

software component bus in the producing process of software products. Under the bus (or data layer), it is the traditional ISEE (integrated software engineering environment) based on traditional software engineering methodology, which support the development of the source programs and documents of the product line core asset components.

Obviously, the new model is a real industrialized PL-ISEE architecture with the software assembly production mode and process resembling the automated assembly line and management system of the modern manufacturing industry. The new PL-ISEE framework model and its realizing mechanism have been detailedly discussed in paper [6, 11].

Actually, the new model is all different from traditional ISEE model based current software development environments. The PL-ISEE realized by the new model will own an assembly line of software products based on the core asset and agent component bus. The component bus, just as a component conveyor belt, will provide all components needed on the assembly line. But, the component bus should provide the reusable COTS components, and which must consist of the broker (agent) systems. So the research on the broker system and architecture will be very important for realizing the new PL-ISEE.

The goal of software product line is to realize the industrialized production of software products. To implement the production of software system in accordance with the software product line, its essence is assembling the software process with the DSSA and software components. Therefore, the product line core assets or software components are fully regarded as a third-party products or COTS (Commercial Off-The-Shelf) software components to develop and issue, and widely used in the network environment and implement value-added. In the distributed network computing environment, it is necessary to make full use of third-party software components to implement the software products' assembly production. Therefore the research and application of software agent technology will become the basis of the software product line development [6].

Figure 1 Notes: Filled boxes represent the products of the process or tools or phases. No filled boxes represent the process or tools.

Product Line Interface Integration and Interface Services (interface layer): AT-Analysis Tools Interface, DT-Design Tools Interface, AST-ASsembly Tools Interface.

Standard Engineering Tools: PLSC-Product Line Standard Classification, PLSD-Product Line Standard Design, PLSR-Product Line Standard Release; DA-Domain Analysis Standard, DD-Domain Design Standard, DI-Domain Implementing Standard.

Domain Engineering Tools: DA-Domain Analysis, DD-Domain Design, DI-Domain Implementation

Application Engineering Tools: AA-Application Analysis, AD-Application Design, AI-Application Implementation.

Core Assets Database Platform and Data Integration Services: NSC-National Standard Components, ISC-Industry Standard Components, ESC-Enterprise Standard Components, LAC-Local Agent Components

III. THE FEATURES AND MECHANISMS OF BROKER ARCHITECTURE

The broker architecture is mainly used to build distributed software system with intrinsic isolated components under heterogeneous network. In such a distributed system, the function realization and the performance guarantee of client application system totally depend on multiple service providers (collectively referred to as the server), which are distributed in the different nodes of the heterogeneous network environment. The client application makes a request to the broker. According to the properties and requirements of the client requests, the broker sends client's requests to a service provider (server-side) in the way of remote collaboration and resource access. The service provider completes the client assignment processing, and returns the results to the client application through the broker. Thus, the broker constitutes a bridge between client application side and the remote server-side, taking charge of coordinating the communication, sending requests, returning results and abnormal information processing etc[7-8].

Thus, the development of a complex distributed application system should be divided into many separate components according to its function, and distribute each component on different nodes in the network environment, so that it can achieve the distribution and expansion of the system. This distributed application system is composed of some independent components interoperating and interacting with each other, rather than a whole application program. Obviously, such a distributed application system also has good flexibility, maintainability, and modifiability. In order to achieve interoperability between different network node components in a distributed system, the system must provide a remote communication mechanism between the component processes. Because if the components respectively handle their own communications, the system becomes dependent on the use of low-level communication mechanism, the client must know the server's location and many other complex issues. In addition, in such a distributed system, adding, updating, moving, exchanging, activation and positioning the component services are essential. As a result, the application system used server components should not know too much about the implementation details of each server component, to ensure portability and interoperability of the system in heterogeneous networks. The development of client applications only uses the service interfaces provided by the server, but implementation details and physical location of the server components are completely transparent. In the distributed applications of broker architecture, it is the broker system that takes charge of communication between the client and server components, as well as the locating operation of server.

The solution of the agent system is just introducing broker component between the client and server components to insulate client and server. The servers distributed on the network should register and publish to the brokers, and make their own services used by client's programs through certain methods and interfaces. Customers access various kinds of services through a broker to send requests. The broker's task is to locate the appropriate server, and send the request to the server and return client with results data or service status information.

The application software system built by the broker architecture has the inherent distributed nature and computing power. Through the broker architecture, the different components in the system, especially server components are distributed on different nodes of the network, and they can go across different hardware and software platforms, network environments and programming languages. Furthermore, them can be updated and expanded dynamically.

IV. The Design of PL-ISEE Broker System

A. The Framework of the PL-ISEE Broker System

According to the functions and characteristics of broker architecture above, the broker architecture generally consists of the following six parts: client, server, broker, bridge, client-side broker, and server-side broker. Broker architecture model is shown in Figure 2, in which the component parts of the role are discussed below [9-11]:

Server/component server: The server is responsible for achieving various function services of the server-side and registering the services to brokers, and then displays and provides its functions through its interfaces. The server interfaces are made up of operations and attributes. These interfaces can be acquired by the Interface Definition Language (IDL), and they generally aggregate the semantically related functions. In object-oriented approach, each service is implemented through one or more objects, and provides its functions through the property and method members of the objects. The server is divided into two types according to different tasks: one is to provide public services or public information server for many applications, the other is to achieve a specific service for specific tasks. In the PL-ISEE core asset and COTS component bus and broker architecture [12], the majority is the first server setting [13-15].

Client/developer: The client is an application program to complete specific problem solving by accessing one or more servers. In the broker architecture, the client's requests will be transmitted to the server by the broker. When the requested operations are completed by the server, the result data processed by the server will be returned the client by the broker. In the broker architecture, it must emphasis on interoperability and communication between the client and the server, which is a dynamic relation model with broker-based messenger rather than pre-defined and invariable static relation model, it is also different from the traditional client / server model. In particular, the client does not have to know the location of their access to the server. The server can be distributed in the different nodes of the network dynamically, and updated and shifted to services entities during operation. In short, the client can issued a service request only by the service interface, and the following request transmission, server location, and operation are transparent. The transparent characteristic has brought great convenience to the development of distributed application system in heterogeneous network, and ensures cross-platform portability and language independence of application system.

(local or remote) Broker: The broker is responsible for transmitting the client requests to the server side and the server result data to the client side. The broker should provide the only tag to every interoperable object or component (such as clients or servers etc.) in the system, and make the tag and the appropriate communication mechanism to locate accurate position of the information receiver. The broker should also provide the registration server and APIs called server operation for clients and servers.

When the client sends a request to a server registered on the local broker, the broker will directly transmit the requests to the server. If the server is not activated right

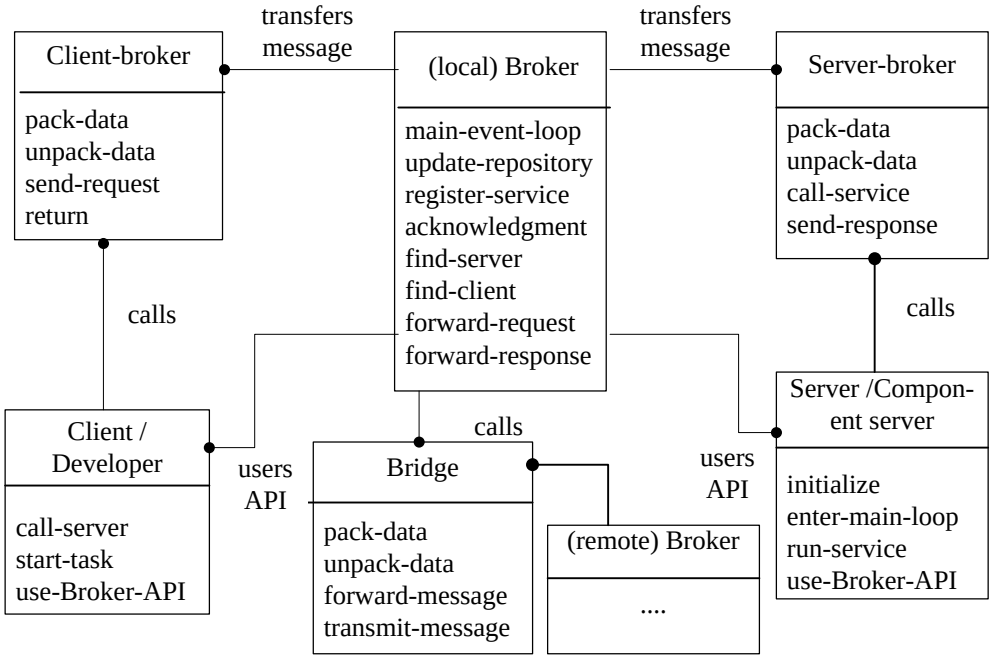

Figure 2. Broker architecture system model

now, the broker will activate it. All the results returned from the server will forward to the requesting client by the broker. If the server requested by the client makes register on another broker (that is not registered on the local broker), the local broker will find a path to reach the goal of the remote broker on the network, and use this path to send the request and returns the result information. Therefore, the broker systems will require the brokers in the network (between the local brokers and remote brokers) communicates with each other and interoperability, this requirement can use the bridge to achieve.

Client-broker: The client-broker is the isolated layer adding between client and broker, taking charge of communication between client and broker. The client broker provides service transparency, so that broker can hide realized details itself to clients. Clients don't need to know all the implemented details how to transmit, store, parameterize and package request information, as well as return the result data between the client-side and server-side, like using a local server as a remote server.

Server broker: The server broker is responsible for receiving the requests sent by the broker, analyzes the received message, unpacks parameters, locates and calls the appropriate service, completes operation requested by the client. At the same time, result data to return to the client should be packaged in accordance with the data semantics. In contrast, when the client broker received the result data returned by server and transmitted by the broker (Note: the client broker is not the same as the broker), it should unpack the result data and sent to the client. In realizing process of the broker bus of the core asset components of PL-ISEE product line and the COTS components, the server broker is very important, because it want to find and match the ideal components in the component database according to the needs of the client application. This is an important work about the development quality of the software product line application system.

Bridge: that is a communication bridge between two different network systems. When the broker system is running in a heterogeneous network environment, we can use the bridge to connect two brokers distributed in two different network systems and implement communication and interoperability each other. At the same time, we can hide implementation details about communication and interoperability of the two brokers through the bridge. The communication mode of the broker system has two ways: one is the direct means of communication; the other is an indirect means of communication. The broker simply establish a communication link between the client and the server adopting the direct means of communication, but the communication task is implemented by the direct participation of the client and server components. Not only does the broker in charge of the establishment and maintenance of the communication link, but take charge of the accomplishment of specific communication tasks adopting the indirect means of communication.

B. The PL-ISEE Broker Operation Mechanism

In the Broker architecture shown as figure 2, the processes of how the client/developer requests the server-side to provide services become the key activities of this architecture. The steps are as follows [16-19]:

1) Run the client application system (it is the application development side in the product line), and the client program is executed through remote call to the server object's interface or method.

2) Client-broker packages all call parameters and related information and generates a message packet. The client-broker will send this message packet to the local broker (that is the broker in figure 2).

3) The local broker seeks for the requested server address in its own library. Since the server is registered in the local broker (If it is not registered, see the following 9), 10), 11), 12) realizing process), therefore the local broker sends the message packet to the corresponding server broker.

4) The server broker unpacks the message packet and gets the parameters and other information, then resolves the service needed calling, and calls the corresponding server object.

5) The server object accomplishes the requested operation tasks and returns the resulting data (it may be a component) to the server broker.

6) The server broker packages the resulting data and related information and generates a resulting message packet, then sends it to the (local) broker.

7) The (local) broker sends this message packet to the client-broker.

8) The client broker receives the message package, unpacks and returns the resulting data to the client program, and then the program continues to execute.

The synchronization execution process of broker architecture is above. That is, the client waits for server-side returning the resulting data and then continues to the next step. Of course, the broker architecture also supports asynchronous calls, namely, the client has not to wait for the returning resulting data after making a request and can execute the next task.

In addition, in the above 3) step, if the local broker does not found the server registration requested by the client in its own library, so how to control the execution? This needs to implement requested services by a different broker (it is called remote broker) in heterogeneous network. The steps are as following four steps:

9) When the local broker receives the request message sent by the client program, and it doesn't find the server address in its own register library, namely this server is not registered in the local broker, but is registered on a remote broker of a node of another network system. In this case, the local broker must send the client request to the remote broker using the network bridge.

10) After the local broker sends the request message to the network bridge, the bridge is responsible for conversing this message protocol defined by the local broker into network public protocol that the two bridges can be understood in a heterogeneous network, after that

the local bridge will send the message to the remote bridge.

11) Remote bridge transfers the received request message from the network common protocol format to remote broker definition format. After the remote broker receives the request message, it will analysis and call the corresponding server services.

12) After the server completes the requested operation and returns the resulting data back to the remote broker, and then the remote broker through the bridge sends the results back to the local broker, at last, the local broker sends the results to the client program. Of course, the result data packages returned from the remote broker to the local broker are also concerns about the conversion of the data package format.

By this time, the communication and collaboration processes of heterogeneous network with multi-broker architecture system are successfully completed.

V. THE IMPLEMENTATION MECHANISM OF THE PL-ISEE BROKER SYSTEM

The implementation of PL-ISEE broker architecture should be based on broker architecture system shown in Figure 2 to design and implement the associated components which make up the system. Therefore, the design and implementation of broker architecture should include the following component implementations and the handling of their mutual relationships [20-22].

A. Determine the Client and Server Object Models:

It should abstract and model the composition objects of the broker architecture according to the distributed computing and resource sharing needs of the broker application system, such as the client object, the server object, the broker object, the client-broker object, the server-broker object. Each object model should be specified the entity according to their respective needs, such as object name, object, request, status value, exception, supported type, type extension, interfaces and operations. The basic computing model determined through the problem solving is the key to design the object models.

In the PL-ISEE architecture model shown as figure 1, the product line core asset and agent component bus can be described as broker objects, and the objects should consist of client side broker, server side broker and (local or remote) broker objects. The software product development (assembling) side in the PL-ISEE can be described as a client object. All software component and architecture products as well as component database system supporting platform can be described as server objects.

Such a design and definition of the PL-ISEE broker system can fully meet the product line computing mode and development of application software products based on the component assembly process. In this process, all the components (including the core asset and COTS components) needed by the application software developers (the client objects) can be got from the server objects (such as component and database servers) by sending requests to the component bus (the broker). The database in accordance with the request selects the components needed by the development side, and returns them to the development side through the broker bus.

B. Determine the Interoperability between the Broker System Objects:

The interoperation between the objects can be achieved by the Interface Definition Language (IDL) in the implementation process of broker systems. The IDL compiler can generate a needed programming language code by using the IDL file as input. In such a way, the IDL can obtain the support of various programming languages easily. A part of the generated programming code is used by the server to communicate with the broker, and the rest part is used by the client to communicate with the broker. The broker can maintain the current server type information through IDL rules. The broker can map the semantic concept to a suitable programming language and realize interoperation presentation when using IDL rules to define interoperability. The biggest advantage of using IDL is to realize the independent between the broker system and programming language, which provides a good mechanism for the realizing distributed computation in heterogeneous network.

C. Determine the API Functions of the Cooperation between the Broker and the Client and Server:

On the client side, the broker must provide the functions of building a client request, sending a request to the broker and receiving the response data and so on. The servers use the API functions to mainly register its implementation information to the broker. The brokers use the registration repository to maintain the server registration information. The repository can be an external file or a database system, which provides a convenient for the server to dynamically register. Since the broker should make sure the needed server while the client’s request arrives, it requires a confirmation mechanism. The broker is responsible for realizing the association from the server identifier to the server object through this confirmation mechanism. This mechanism requires that the server side API of the broker be able to generate a unique identifier for a variety of servers. If the client, the server and the broker are running as different processes, the API function needs an effective process communication mechanism between the client, the server and the local broker.

D. Hide the Lower Level Communication Implementation Details with the Broker Object:

In the broker architecture, the client broker (client side broker) and server broker (server side broker) are necessary. The client broker's responsibility is to package the client process request information into messages (packets) and send the messages to the local broker. Then it will receive result data from the local broker and return it to the client process. According to the internal message protocol, the communication and information transmission between client and local broker can be

completed. The server broker responsibility is to receive the local broker's requests and call corresponding server interface methods as well as process the requested task. And then it packages the result information made by the server into message packets and transmits these packets to the local broker according to internal message protocol. In particular, the side broker (the client or server broker) is often a part of the corresponding client or the server process, and hides the communication details by using their own inter-process communication mechanism which communicates with the local broker. They can also pack and unpack the parameters and results and convert the information into a system-independent format. This capability is very useful to implement and hide the lower level complex and multi-broker communication mechanism in the heterogeneous network environment.

E. Broker Component Design:

The broker is the communication and message passing bridge between the client and the server. When the client sends a request to the server, the broker is responsible for receiving the request, and in accordance with the request and the communication protocol, establishes a communication link between the client and the server brokers, which guarantees the implementation of communication and data transmission between two side brokers or they and the broker. Broker components design should include main works:

(1) Designate the detail online interaction protocol among the broker and client broker and server broker, then make the mapping plan from the request, response and exception to an internal message protocol; (2) The broker must be available to each machine in the network; 3) The broker must provide the client identification mechanism; 4) When the side brokers do not provide the function of data packing and unpacking to messages and results, the broker should provide this function; 5) If the system supports the asynchronous communication between the client and the server, message buffers should be stetted in the broker or side brokers as a temporary message storage area; 6) The broker should provide a directory service to associate the local server identifier with the corresponding physical address of the server.

F. Develop IDL Compiler:

If the broker system uses the interface definition language IDL to achieve interoperability, it needs to provide an IDL compiler for each supported programming language. This compiler can convert the client application and server interface definition into the programming language code, in order to hide specific implementation details of the server object and its methods.

Thus, to sum up the points which we have just indicated, the framework and realizing mechanism of the broker system of the PL-ISEE component agent bus are designed based on the CORBA architecture. According to the design idea and method, the new industrialized PL-ISEE and its component agent bus shown as figure 1 can be completely implemented by current popular BORBA technology and method. Some complex parts of the PL-ISEE component agent bus, such as the client and sever objects, broker objects, Interface Definition Language (IDL), IDL compiler, the communication rules and implementation details, etc., are all supported by CORBA system. The research idea and realizing method of the PL-ISEE can enhance the development quality and application ability of the future PL-ISEE.

VI. CONCLUSIONS

The broker architecture is an inevitable model to realizing distributed computing in heterogeneous network environments. In today's Internet application information system, many systems use the broker architecture model. For example, the Common Object Request Broker Architecture (CORBA: Common Object Request Broker Architecture) developed and promulgated by OMG, which gets widely industry support and application, is becoming a standard broker architecture and object oriented technology realizing distributed computing in heterogeneous network environment.

In this paper, a new industrialized PL-ISEE model is shown as figure 1. A core constituent part of the PL-ISEE is product line agent component bus in the data layer. How to realize the agent bus is also becoming a key problem to implement whole PL-ISEE. In the paper, the CORBA architecture and technology are employed as the broker of the product line core asset and COTS agent component bus in the new PL-ISEE model, so as to realize the complex component broker system of the integrated component-assembling environment of the software product line in heterogeneous network. The implementation of the PL-ISEE system based on the CORBA architecture has the advantages of location transparency between the client (development side) and the server (component provided side), the dynamic updating and extensibility of the component server, independence and portability of the system platform, interoperability between different broker systems. Besides, this broker system provides a solid theory foundation for the research and development of current Internetware, Mobile-Agent, Cloud Computing and Cloud Service, etc. technology system [23-25]. Of course, there is still a long way to go as for the research and improving of the broker theory and technology. They are major problems of the broker architecture system research and application, such as intelligent broker, the broker system efficiency and error detection.

ACKNOWLEDGMENT

This project is supported by Fund of Jiangsu University Natural Science Basic Research Project, Grant No. 08KJD520013 and Jiangsu Key Built Discipline Project—Computer Application Technology.

REFERENCES

[1] YANG FuQing, "Thinking on the Development of Software Engineering Technology". JOURNAL OF SOFTWARE, vol.16, no.1, pp.1-7, 2005.

[2] Zhang Xinyu, Zheng Li, Sun Cheng, "The research of the component-based software engineering", 6th International Conference on Information Technology: New Generations, pp.1590-1591, 2009.
[3] YANG Fuqing, LV Jian, MEI Hong, "Internetware technology System: A Approach centered Architecture", Science in China (Series E: Information Sciences), Vol.38, No.6, pp.818-828, 2006.
[4] Thurimella Anil-Kumar, Padmaja Maruthi T., "Software product line engineering: A review of recent patents," Recent Patents on Computer Science, vol.3, no.2, pp.148-161, 2010.
[5] LV Jian, MA Xiaoxing, TAO Xianping, CAO Chun, HUANG Yu, YU Ping, "Research on Internetware Oriented Environment Driven Model and Supporting Technology", Science in China (Series E: Information Sciences), Vol.38, No.6, pp.864-900, 2008.
[6] Jianli DONG, Ningguo SHI. "A study on framework and realizing mechanism of ISEE based on product line," Journal of Software, Vol.5, No.10, p.1077-1083, 2010.
[7] XIE Wu-ping, XUE Jin-yun, WAN Song-song. "Aspect-Based Component Model and Its Assembly and Implementation", Computer Technology and Development, Vol.19, No.4, pp.160-160, 2009.
[8] Zhijian WANG, Yukui FEI, Yuanqing LOU, Software Component technology and Application. BeiJing: Science Press, April 2004.
[9] HE Tian-zhang,WANG Pan-qing,LI Xiao-hui, "Research and Application on CORBA Component Assembly", Science Technology and Engineering, Vol.8, No.1, pp.84-86, 2008.
[10] CHANG Bing-guo,WANG Xiang-zong, "Research and development of component integration support platform", Computer Engineering and Design, Vol.32, No.8, pp.2712-2715, 2011.
[11] Jainli DONG, Ningguo SHI, "Research on the CORBA Implementation Mechanism of a New Industrialized PL-ISEE", Journal of Software, Vol.7, No.5, pp.1171-1176, 2012.
[12] Jianli DONG, Ningguo SHI, Yong ZHANG, "Research and Implementation on an Ideal Industrialized PL-ISEE and Its Database Supporting Platform", JCIT, Vol.7, No.13, pp.206-214, 2012.
[13] Jianli DONG, "Framework and Schema Design of A New Industrialized PL-ISEE Database Supporting Platform". Advances in Information Sciences and Service Sciences, accepted, 2012.
[14] MAO Yingchi, LIANG Yi1, WANG Zhijian. "Design and Implementation of Model for Heterogeneous Software Component Composition", Computer Engineering, Vol.31, No.4, pp.56-57, 2005.
[15] YE Junmin, CHEN Zhuo, LEI Zhixiang, YE Yanfeng, ZHAN Zemei, "Research on application development process based on component composition", Application Research of Computers, Vol.25, No.6, pp.1736-1738, 2008.
[16] Patrizio Pelliccione, Massimo Tivoli, Antonio Bucchiarone, Andrea Polini, "An architectural approach to the correct and automatic assembly of evolving component-based systems", Journal of Systems and Software, Vol.81, No.12, pp.2237-2251, 2008.
[17] Basem Y Alkazemi, "A Precise Characterization of Software Component Interfaces, Journal of Software", Vol 6, No 3, pp.349-365, 2011.
[18] Longye Tang, Yukui Fei, Zhijian Wang, "Service-Oriented Component Model", IJACT- International Journal of Advancements in Computing Technology, Vol.3, No.1, pp.68-79, 2011.
[19] Li Na, Wang Weizhe, "ON CORBA BASED ACCESS MIDDLEWARE OF HETEROGENEOUS DATABASE AND ITS IMPLEMENTATION", Computer Applications and Software, Vol.27, no.5, pp.162-164, 2012.
[20] YU Bin, HUA Qingyi, "Analysis and practice of distributed heterogeneous computing environment based on CORBA", Journal of Northwest University (Natural Science Edition), vol.30, no.2, pp.113-117, 2000.
[21] YAN Juan, ZHANG Lifang, CAI Xuqing, "Research and Development of CORBA", Software Guide, Vol.8, No.6, pp.41-43, 2009.
[22] HE Tianzhang, WANG Panqing, LI Xiaohui, "Research and Application on CORBA Component Assembly," Science Technology and Engineering, Vol.8, No.1, pp.84-86, 2008.
[23] Wenbin Hu, Zhengbing Hu, Yuheng Cheng, Hai Zhang, Wei Song, "Modeling and Simulation on Dynamic Allocation and Scheduling of Multi-resource problem," Journal of Computers, Vol.6, No.7, pp.1369-1377, 2011.
[24] Weidong Zhao, Haifeng Wu, Weihui Dai, Xuan Li, Fei Yu, Chen Xu, "Multi-agent Middleware for the Integration of Mobile Supply Chain," Journal of Computers, Vol.6, No.7, pp.1469-1476, 2011.
[25] Feng Liang, Shilong Ma, André Luckow, Bettina Schnor, "Advance Reservation-based Computational Resource Brokering using Earliest Start Time Estimation," Journal of Computers, Vol.7, No.6, pp.1329-1336, 2012.

Jianli Dong was born in Shanxi province, China, in 1957. He got his Bachelor of Mathematics Science in Northwest Normal University, Lanzhou, Gansu province, China, in 1988 and got his Master of Software Engineering in Beijing University of Aeronautics and Astronautics, Beijing, China, in 1995. He is now a professor at the School of Computer Engineering in HuaiHai Institute Technology, Lianyungang, China. He has published 70 papers, and completed over 8 scientific research projects, and won 6 times scientific and technological progress awards from the province and military.

Mr. DONG current research interests include software engineering, integrated software engineering environment, software architecture, engineering database system, object-oriented technology.