D E S C R I P T I O N

A method and a bridge for coupling a server and a client of different object types

The invention relates to a bridge for coupling of a client to a server computer and to a corresponding method, computer system and a computer program.

From U.S. H0001895 an application provider and method for communication is known. A method is provided for communication of an application provider between a first sub-system and a second sub-system of a common node.

For example, the application provider may be a mobile application provider, the first and second sub-systems may be a home location register and a visitor location register, and the common node may be a wireless telecommunication system. The system is implemented by means of an object request broker (ORB) such as the common request broker architecture (CORBA) defined by the object management group (OMG). The resource manager application serves as the "bridge" or "conduit" between a call processor and a resource assembly.

From U.S. 6,094,688 a system for modular application collaboration including filtering at the source and proxy execution of compensating transactions to conserve server resources is known. This system provides interoperability between applications including a plurality of connectors for communicating with a like plurality of applications and an interchange server.

The interchange server includes an application collaboration module and service module. The services module transfers messages between connecters and the application collaboration module. The application collaboration defines the interoperability between two or more applications. The interchange server service module includes a transaction service and an error service. Transactions are executed in the application collaboration module.

From U.S. 6,085,220 an enterprise interaction hub for managing an enterprise web system is known. The enterprise interaction hub includes a number of layers that interact to manage an enterprise web system.

An interaction layer receives requests to the enterprise web system and returns responsive web pages. A presentation layer is coupled to the interaction layer an generates the responsive web pages. A business layer is coupled to the presentation layer and provides business logic for use by the presentation layer in generation the responsive web pages.

An integration layer is coupled to the business layer and interfaces with existing legacy data to provide the legacy data to the business layer. A trend collection layer monitors and accumulates historical information from the interaction layer, the presentation layer, the business layer and the integration layer. The trend collection layer also stores the historical information in a trend database.

A profile database, accessible by the presentation layer and the business layer, stores profile data, including data mined from the trend database, that characterizes individual user access to the enterprise web system. The profile data is used

by the presentation layer and the business layer to provide customized dynamic content in the generated web pages.

From U.S. 6,085,030 a network component server is known. The component server architecture enables consumer nodes of a computer network to interact with heterogeneous software components and services distributed throughout the network, as well as network devices and data.

Distributed interactions between a consumer and a heterogeneous software is achieved, in part, by registering and locating the component and services. An object neutral global component registry with access controls of the architecture interoperates with the component management service to transparently ensure proper administration, of indication and run-time binding access to components offered in responds to request from applications executing on the consumer nodes.

The architecture is implemented on a component server node of the network that is configured to communicate with the consumer, i.e., client, nodes in client-server computing arrangements. That is, the component registry of the component server node responds to a consumer application request by locating a heterogeneous component for the consumer. The registry offers this component to the consumer by providing an appropriate interface between the object module of the software component. This registry is organized as a plurality of cooperating storage entities including a description repository, an offer repository, an interface adapter repository and an object factory repository.

A component server architecture is provided to interact with

heterogeneous software components and services distributes throughout the network, as well as network devices and data. Distributed interaction between a consumer and heterogeneous software is achieved, in part, by registering and locating the components and services.

An object-neutral global component registry with access controls of the architecture interoperates with a component management service (CMS) to transparently ensure proper administration, authentication and run-time binding access to components offered in response to requests from applications executing on the consumer nodes. The architecture is implemented on a component server node of the network that is configured to communicate with the consumer, i.e., client, nodes in client-server computing arrangements. That is, the component registry of the component server node responds to a consumer application request by locating a heterogeneous component for the consumer. The registry offers this component to the consumer by providing an appropriate interface between the object module of the consumer and the object module of the software component.

This registry is preferably organized as a plurality of cooperating storage entities including a description repository, an offer repository, an interface adapter repository and an object factory repository.

From U.S. 5,913,061 a modular application collaborator is known for providing inter-operability between applications including a plurality of connectors for communicating with a like plurality of applications and an interchange server. The interchange server includes an application collaboration module and a service module.

The service module transfers messages between connectors and the application collaboration module. The application collaboration module defines the inter-operability between two or more applications and includes a trigger and a transaction responsive to the trigger. The trigger is activated upon receipt of data from one or more connectors resulting in the transaction delivering data to one or more connectors for transfer to an associated application.

From U.S. 6,115,646 a dynamic and generic process automation system is known. This includes a generic object-oriented process automation engine that provides workflow management services in a heterogeneous distributed computing environment. The system consists of three major parts: a build time part used to capture and store process definitions, and to request the enactment of a process; a run time part used to schedule, execute , and monitor the requested process; a CORBA to plug-in-software applications needed to execute processes and to allow interactions among the system components. This system is event-driven and provides scheduling and resource allocations schemes.

The system can incorporate a trader service which provides a "yellow pages" for objects; it allows objects to publicize their services and allow clients to find them based on upon which services the client needs. For example, a resource allocator can use the service to find a resource that satisfies a set of agents constrains.

The above referenced prior art system and methods mostly rely on object oriented software technologies, such as CORBA. CORBA is a vendor-independent architecture and infrastructure that

computer applications use to work together over networks. In CORBA, every object instance has its own unique object reference, an identifying electronic token.

Clients use the object references to direct their invocations, identifying to the object request broker the exact instance they want to invoke. The client acts as if its invoking an operation on the object instance, but it is actually invoking on the IDL stub which acts as a proxy. Passing through the stub on the client side, the invocation continues through the object request broker, and the skeleton on the implementation side, to get to the object where it is executed on the server.

The object request broker provides a mechanism for transparently communicating client requests to target object implementations. The object request broker simplifies distributed programming by decoupling the client from the details of the method invocations. This makes client requests appear to be local procedure calls. When a client invokes an operation, the object request broker is responsible for finding the object implementation, transparently activating it if necessary, delivering the request to the object and returning any response to the caller.

The IDL stubs and skeletons serve as the "glue" between the client and the server applications, respectively, and the object request broker. The transformation between IDL definitions and the target programming language is automated by a IDL compiler. The use of a compiler reduces the potential for inconsistencies between client stubs and server skeletons and increases opportunities for automated compiler optimizations.

Another important object oriented technology is Enterprise Java Beans. Enterprise Java Beans provides a framework for components that can be "plugged in" to a server, thereby extending that service functionality. Client programs execute methods on remote Enterprise Java Beans by way of an Enterprise Java Beans object. The Enterprise Java Beans object implements the "remote interface" of the EJB component on the server.

A common disadvantage of the CORBA and Enterprise Java Beans object technologies is that corresponding applications being available in an company's intranet are not accessible through a firewall from the internet.

For this and other reasons companies recently have started to use the Simple Object Access Protocol (SOAP) as the object oriented technology of choice as (SOAP) is firewall compatible. SOAP is a protocol for exchange of information in a decentralized, distributed environment. It is an XML based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instance of application - defined data types, and a convention for representing remote procedure call and responses. SOAP can be used in combination within an HTTP extension framework.

Fig. 1 shows and example of a prior art network. The network includes internet 1 and intranet 2 which are separated by firewall 3. A SOAP client 4 is coupled to SOAP server 5 via internet 1.

Within the companies' intranet 2 a CORBA client 6 and an EJB client 7 are coupled to a CORBA server 8 and an EJB server 9,

respectively.

Because of the tight coupling between the CORBA and EJB clients 6, 7 and their respective servers 8, 9 the applications of servers 8 and 9 are not accessible through the firewall 3 from the internet 1. As a consequence the SOAP client 4 can only access the SOAP server 5 – or other servers which are not shown in fig. 1 – over the internet 1 but can not access applications provided on servers 8 or 9 in the company intranet 2. This is a serious disadvantage for offering a variety of applications to customers, in particular if existing CORBA and / or EJB systems are available.

Thus, a need exists to provide an improved method for coupling of different types of client and server objects.

According to a first aspect of the present disclosure, there is provided a bridge for coupling a client to a server object of a second type by providing access to said server object as an object of a first type,

the bridge comprising a server component of the first type representing the server object of the second type, and

the bridge comprising a client component of the second type representing said client, and

the bridge comprising a mapping component capable of mapping a request to the server object of the first type into a corresponding request to the server object of the second type, and

the server component of the first type being responsive to a receipt of a request from the client to the server object of first type by passing it to said mapping component, and

the mapping component being responsive by mapping it to a corresponding request to the server object of the second type and passing it to the client component of the second type, and

the server component of the second type being responsive by passing said mapped request to the server object of the second type.

According to a second aspect of the present disclosure, there is provided a computer system having a client computer of a first object type and a bridge in accordance with the bridge described above for establishing a communication path between the client computer and a server computer of a second object type.

According to a third aspect of the present disclosure, there is provided a method for coupling a client to a server object comprising the steps of

coupling of the client to a bridge in accordance with the bridge described above,

providing a uniform resource locator of the server object to the bridge and providing parameter data to the bridge indicative of the object type of the server object, and

coupling the bridge to the server object via the client component of the bridge having the same object type as the server object.

According to a fourth aspect of the present disclosure, there is provided a system for automatically coupling a client to a server object of a second type by providing access to said server object as an object of a first type, the system comprising means adapted for carrying out the steps of the method described above.

According to a fifth aspect of the present disclosure, there is provided a data processing program for execution in a data processing system comprising software code portions for performing the method described above when the program is run on that computer.

According to a sixth aspect of the present disclosure, there is provided a computer program product stored on a computer usable medium, comprising computer readable program means for causing a computer to perform the method described above when the program is run on the computer.

The invention allows to couple a client of a first object type, for example a SOAP client, to a server of a second object type, for example a CORBA or a EJB server. The coupling can be achieved over the internet, intranet and firewalls.

In a preferred embodiment this is accomplished by providing a bridge for coupling server and client components of different object types in order to provide a communication path between the respective client and server via the bridge. The bridge contains a mapping component for mapping of objects of a first type to objects of a second type, for example mapping of SOAP objects to CORBA or EJB objects and / or vice versa.

The invention is advantageous in that it allows to remove the requirement for using stubs / skeletons for SOAP generated objects by reusing the already existing client stubs of the CORBA and / or EJB servers and by creating a generic meta program to locate the initial contexts and homes by accessing pre- configured information. The SOAP based clients must specify the object name, the method name and the parameters.

In addition the uniform resource locator of the bridge is published together with CORBA and / or EJB application capabilities to a web directory so that web service requestors - after determining that their "real" service provider can only be accessed by CORBA and / or EJB protocols - can access the bridge which then passes on the request "in proxy" to the actual CORBA and / or EJB service. One application is to store the uniform resource locator of the bridge in a yellow pages type server.

This way the invention allows to realize distributed application services where parts of the application can be distributed over the internet and intranet being separated by firewalls. This is of particular importance for usage of existing applications which are not firewall capable in such distributed environments.

In the following the invention is explained in more detail with respect to the drawing in which

Fig. 1 is a prior art computer system having server and clients of different object types,

Fig. 2 is a computer system having an embodiment of the bridge of the invention,

Fig. 3 is the computer system of fig. 2 having a yellow pages server for locating an application service in the internet and / or the intranet, and

Fig. 4 is a flow chart of an embodiment of the invention for coupling objects of different types.

Fig. 2 shows a computer system - using like new numerals for like components as in fig. 1 - having a SOAP client 4 and a SOAP server 5 which are coupled by internet 1. The bridge 10 is contained in the intranet 2 which is coupled to the internet 1 via firewall 18. Intranet 2 further contains CORBA client 6 and EJB client 7 which are coupled to CORBA server 8 and EJB server 9 respectively.

The bridge 10 contains a SOAP server component 11 and a SOAP component client 12. The SOAP server component 11 serves to establish a communication path between the SOAP client 4 and bridge 10 via the internet 1 whereas the SOAP client 12 serves to establish a communication path between the bridge 10 and the SOAP server 5 via the internet 1 - depending on the particular application.

The bridge 10 further comprises an EJB client component 13 and an EJB server component 14. The EJB client component 13 serves to establish a communication path between the bridge 10 and the EJB server 9 whereas the EJB server component 14 serves to establish a communication path between the EJB client 7 and the bridge 10 via the intranet 2. Likewise the bridge 10 contains CORBA client component 15 and CORBA server component 16. The CORBA client 15 serves to establish a communication path between the bridge 10 and CORBA server 8 whereas the

CORBA server component 16 serves to establish a communication path between CORBA client 6 and the bridge 10.

The bridge 10 further comprises a mapping component 17 for mapping of an object of a first type to an object of a second type and / or mapping of corresponding method requests and parameters. For example the mapping component 17 can map a SOAP based method request to a CORBA or EJB method call at run-time. Likewise the mapping component 17 can map CORBA and / or EJB method calls to SOAP based method requests.

The bridge 10 has a uniform resource locator as well as the other server and client entities as depicted in fig. 2.

The usual operation of SOAP client 4 is to access the SOAP server 5 with an application request via the internet 1. In case the desired application is not a SOAP application but - in the example considered here - a CORBA or EJB application running on a server of intranet 2 the SOAP client 4 must first access the bridge 10.

This is done for example by providing the uniform resource locator of the bridge 10 to the SOAP client 4. This access request of SOAP client 4 also includes a parameter for indicating the object type of the desired application and / or further parameters for input into the application on the remote server.

The access request of SOAP client 4 based on the uniform resource locator of bridge 10 invokes the SOAP server component 11 which appears to SOAP client 4 to be a usual SOAP server. Therefore the operation of SOAP client 4 does not need to be modified in any respect. Based on the object type of the

desired application a corresponding client component of that object type is selected by the bridge 10.

For example this SOAP client 4 desires access to a CORBA type application on server 8. This is indicated by the SOAP client 4 to the bridge 10 in its access request by specifying the object type of the application. Based on this indication of the object type the CORBA client component 15 is selected and activated by the bridge 10.

The CORBA client component 15 appears to the CORBA server 8 to be a usual CORBA client; therefore the operation of the CORBA server 8 does not need to be changed in any respect for integration into the computer system of fig. 2. This way communication paths are established between the SOAP client 4 and the SOAP server component 11 as well between the CORBA client component 15 and the CORBA server 8.

A SOAP based method request of SOAP client 4 is received by bridge 10 by means of its SOAP server component 11. From there it is inputted into the mapping component 17. The mapping component 17 provides a CORBA method call having the same functionality as the inputted SOAP based method request. This CORBA method call is inputted into the CORBA client component 15 from where it is transmitted to the CORBA server 8.

This invokes the desired application on CORBA server 8. The parameter data provided by the SOAP client 4 is inputted into the application of CORBA server 8. For example the application of CORBA server 8 can be a financial services application, such as portfolio management or financial planning.

In this case the input parameters provided by the SOAP client

4 can be stock market data or the like on which the analysis of the application is based. These input parameters for the application program are transmitted from the SOAP client 4 to the SOAP server 11 and to the CORBA server 8 via mapping component 17 and CORBA client component 15.

Fig. 3 shows a further preferred embodiment having an additional yellow pages type server 19. The server 19 has a data base which contains a number of available application services and corresponding object types of the application services as well as the uniform resource locator of an application service. The database can be searched based on key words, such as by a Boolean key word search. The SOAP client 5 can connect to the server 19 via internet 1 in order to query server 19 to locate a desired application service.

A corresponding query which is inputted by the SOAP client 4 into the server 19 results in a hit list from which the SOAP client 4 selects a particular application service. By selecting the application service on this hit list the object type of the application service as well as its uniform resource locator are provided to the SOAP client 4. In addition the date base also contains a uniform resource locator of the bridge 10 for providing a communication path to a desired application service. This uniform resource locator of the bridge 10 is also provided by the server 19 to the SOAP client 4.

For example the desired financial services application is of the CORBA object type. As the CORBA and SOAP object types differ it is inferred in the SOAP client 4 that access to the required CORBA server 8 is only possibly via the bridge 10. Hence SOAP client 4 performs an access operation to the bridge

10 by means of the uniform resource locator of the bridge 10 and provides the object type and the uniform resource locator of the application service to the bridge 10. Again further input parameters for input into the application service program can be provided by the SOAP client4.

Based on the object type and the uniform resource locator of the desired application service the bridge establishes a communication link between its CORBA client component 15 (cf. fig. 2) and its CORBA server 8. This way the SOAP client can communicate with the CORBA server 8 as explained in detail with respect to fig. 2.

Fig. 4 shows a preferred embodiment of the method of the invention. In step 20 the SOAP client performs on access operation to a yellow pages type server in order to identify a desired application. In step 21 the SOAP client queries the yellow pages server in order to find the URL of the desired application and its object type. In addition the URL for a corresponding bridge results from the query for access to the application.

In step 22 it is decided whether the object type of the application is of the type "SOAP".

If this is the case the SOAP client directly connects to the application service in step 23 by entering the URL of the application service into its web browser program.

If the contrary is the case the SOAP client obtains the URL of the bridge from the yellow pages server in step 24 in order to access the bridge by means of this URL in step 25.

In step 26 the bridge is provided with the URL of the desired application service and further parameters, such as data indicative of the object type of the application and input parameters for the application.

In step 27 the bridge creates a communication path to a server containing the desired application based on the URL provided by the SOAP client requests. After this communication path is created the communication between the SOAP clients and the application server through the bridge can start and step 28.

The communication is established and maintained through the server component and the CORBA client component of the bridge by means of the intermediary mapping component of the bridge.

Likewise it is possible to establish a communication path between a CORBA client and a SOAP server or between an EJB client and a SOAP server by using the corresponding client and server components of the bridge 10. This way it is possible to build distributed application services relying on a variety of object type applications being distributed over different kinds of networks separated by firewalls.

List of reference numerals

internet	1
intranet	2
firewall	3
SOAP client	4
SOAP server	5
CORBA client	6
EJB client	7
CORBA server	8
EJB server	9
bridge	10
SOAP server component	11
SOAP client component	12
EJB client component	13
EJB server component	14
CORBA client component	15
CORBA server component	16
mapping component	17
firewall component	18
server	19

We claim:

1. A bridge for coupling a client to a server object of a second type by providing access to said server object as an object of a first type,

the bridge comprising a server component of the first type representing the server object of the second type, and

the bridge comprising a client component of the second type representing said client, and

the bridge comprising a mapping component capable of mapping a request to the server object of the first type into a corresponding request to the server object of the second type, and

the server component of the first type being responsive to a receipt of a request from the client to the server object of first type by passing it to said mapping component, and

the mapping component being responsive by mapping it to a corresponding request to the server object of the second type and passing it to the client component of the second type, and

the server component of the second type being responsive by passing said mapped request to the server object of the second type.

2. The bridge of claim 1, wherein the first object type is responsive to a request of the simple object access protocol (SOAP) type.

3. The bridge of either one of claims 1 and 2, wherein the second object type is responsive to a request of the enterprise Java Beans (EJB) type or the common object request broker architecture (CORBA) type.

4. The bridge of any one of claims 1, 2 or 3, the bridge further comprising means for coupling to the client via the internet.

5. The bridge of any one of the preceding claims, the bridge further comprising means for coupling to the server object via an intranet.

6. The bridge of any one of the preceding claims, the bridge further comprising an assigned uniform resource locator (URL) for access of the bridge from the client.

7. The bridge of any one of the preceding claims, the bridge further comprising a receiving component for the uniform resource locator of the server and for parameter data indicative of the second object type for selection of the client component of the second type.

8. The bridge of any one of the preceding claims, the bridge further comprising a receiving component for application specific parameters to be provided to the server object as input data.

9. A computer system having a client computer of a first object type and a bridge in accordance with any one of the preceding claims 1 to 8 for establishing a communication path between the client computer and a server computer of a second object type.

10. The computer system of claim 9 having a firewall component in the communication path.

11. The computer system of either one of claims 9 and 10, the client computer having means for accessing a yellow pages type server for querying the yellow pages type server in order to obtain a uniform resource locator of a desired location of the server computer.

12. The computer system of claim 11, the yellow page server providing the object type of the application and – in case of different server and client object types – the uniform resource locator of the bridge.

13. A method for coupling a client to a server object comprising the steps of

coupling of the client to a bridge of any one of the claims 1 to 8,

providing a uniform resource locator of the server object to the bridge and providing parameter data to the bridge indicative of the object type of the server object, and

coupling the bridge to the server object via the client component of the bridge having the same object type as the server object.

14. The method of claim 13, further comprising the steps of

querying a yellow page type server by the client for finding a desired application,

providing a uniform resource locator of the application and the object type of the application to the client, and

providing the uniform resource locator of the application from the client to the bridge.

15. A system for automatically coupling a client to a server object of a second type by providing access to said server object as an object of a first type, the system comprising means adapted for carrying out the steps of the method according to either one of claims 13 and 14.

16. A data processing program for execution in a data processing system comprising software code portions for performing a method according to either one of claims 13 and 14 when the program is run on that computer.

17. A computer program product stored on a computer usable medium, comprising computer readable program means for causing a computer to perform a method according to either one of claims 13 and 14 when the program is run on the computer.