Title

A Platform

Technical Field

The invention concerns a platform for integrating an enterprise.

Background of the Invention

A Service Oriented Architecture (SOA) is a collection of services. These services communicate with each other. The communication can involve simple data passing or it could involve two or more service coordinating some activity. Connecting the services to each other is required.

A service is a function that is well defined, self-contained, does not depend on the context or state of other services. A service is the endpoint of a connection and it therefore abstracts out the underlying implementations of the process. An SOA defines how services are used to implement a business process.

Not every executable code is a service, the majority are not. A service will always interact with a requestor and has certain attributes. These attributes include:

- a requestor must be able to invoke a service remotely. This means that they cannot assume they have a shared computing environment;
- There is no shared state information (for example, in a shared database). The only information passed will be in the invocation process (typically in a message sent from the requestor);
- a service must be able to describe and advertise itself to potential requestors. The description must be sufficient to allow the requestor to dynamically invoke the service. This implies a shared protocol at this level even though protocols for service implementation may vary;
- The description defines the service and also the Quality of Service (QoS) provided by the service and the Service Level Agreement (SLA). It also defines the format(s) and protocol(s) supported by the service (a service may mandate a single format and protocol); and
- The requestor must assume an asynchronous interaction with the service to provide a reliable service. The service may offer a real-time synchronous communication.

The SOA defines how these attributes are implemented. The platform enables use of services and service orientation.

Web services management tools help businesses manage their web services. However existing tools suffer from many problems.

Clients consuming web services need to search via Universal Description Discovery and Integration (UDDI) protocol for the endpoint information. Once this is known, it will bind the web service and consume the available services. The binding process will have a problem if the web service endpoint changes. In this case, the client must update its programs in order to continue to consume the services. This not only increases the maintenance costs but also breaks interoperability.

Most web services only have a single instance running on the server. This causes a bottleneck in performance if numerous requests arrive. These requests are queued and serviced by the web service on a first come first serve basis. As a result, there will be a long delay in response time.

Web service availability is not guaranteed. If the server is down, the web service is unable to service the requests.

Enterprises which need to implement web services must invest in many systems and servers. They include the UDDI registry, SOAP server and other systems. There is not a single platform for the enterprise to manage all these servers. Administrators not only have administer these servers but also ensure they can communicate with each of them in order to operate.

Summary of the Invention

In a first preferred aspect, there is provided a platform for integrating an enterprise, the platform comprising:

a service registry to enable client applications to search for registered web services;

a routing engine to route the web services selected by the client applications to end points according to at least one configurable routing rule; and

a management engine to manage the web services after they have been invoked;

wherein the web services are abstracted from the client applications to enable platform independence and the at least one configurable routing rule is dynamically configurable.

The configurable routing rule may be a rule selected from a group consisting of: priority based queues to meet Service Level Agreement (SLA) standards; load balancing comprising load, fastest response time, or round robin algorithms; geographical proximity to enhance performance; and content based routing based on header content of a message.

The management engine may perform security, version control, access control, end point translation, data schema translation and load balancing for the web services. The management engine may also handle security, recording audit logs and caching.

The platform may define a strategy to implement adaptive web services.

Agents may be provided by the platform to integrate WS-security into the client applications.

In a second aspect, there is provided a method for selecting an optimal web service for a client application, the method comprising:

checking a service level agreement to obtain minimum service levels for a plurality of web services; and

instructing a routing engine to use a load balancing process to select the optimal web service from the plurality of web services according to the quantity of service instances deployed and maintains a predetermined level of quality of service.

The method may further comprise the step of determining the geographical proximity of the client application to the routing engine to increase the performance of the web service.

The method may further comprise the step of categorizing geographical locations of web services in the service registry.

Brief Description of the Drawings

An example of the invention will now be described with reference to the accompanying drawings, in which:

Figure 1 is a high level architecture diagram of the platform; and

Figure 2 is a message flow diagram of requests and responses passing within the platform.

Detailed Description of the Drawings

Figure 1 and the following discussion are intended to provide a brief, general description of a suitable computing environment in which the present invention may be implemented. Although not required, the invention will be described in the general context of computer-executable instructions, such as program modules, being executed by a personal computer. Generally, program modules include routines, programs, characters, components, data structures, that perform particular tasks or implement particular abstract data types. As those skilled in the art will appreciate, the invention may be practiced with other computer system configurations, including hand-held devices, multiprocessor systems, microprocessor-based or programmable consumer electronics, network PCs, minicomputers, mainframe computers, and the like. The invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote memory storage devices.

Referring to Figure 1, a platform 10 for integrating an enterprise is provided. The platform 10 comprises: a service registry 15, a routing engine 20 and a management engine 25. The service registry 15 enables client applications 50 to search for registered web services 100. The routing engine 20 routes the web services 100 selected by the client applications 50 to end points according to at least one configurable routing rule. The routing rules are dynamically configured. There may be more than one routing rule used in combination by the routing engine 20. The management engine 25 manages the web services 100 after they have been invoked. This allows the web services 100 to be abstracted from the client applications 50 and achieve platform independence.

The routing rules include:

1. priority based queues to meet Service Level Agreement (SLA) standards;
2. load balancing comprising load, fastest response time, or round robin algorithms;
3. geographical proximity to enhance performance; and
4. content based routing based on header content of a message.

The capabilities of the management engine 25 include performing security, version control, access control, end point translation, data schema translation and load balancing for the web services 100. The management engine 25 also handles security, recording audit logs and caching.

The platform 10 is a single platform which enables registration, searching and discovery of web services 100 within the enterprise, amongst partners or by the general public. The platform 10 performs web service performance, load balancing, error recovery and data transformation, web service schema management and routing. Data transformation involves transferring data into standard schemas, for example, HIPAA or HL7. The schemas are validated according to a validation regime. The platform 10 guides enterprises to implement technology from a business standpoint. Business processes are exposed as web services 100 and when business processes change, the web services 100 are updated. The platform 10 manages versioning and prevents the web service consumers from breaking down.

The platform 10 responds quickly to changes in business processes. The platform 10 enables enterprise applications and software to interoperate and exchange data. The platform 10 empowers staff to monitor and manage web services 100 easily.

1) Service Registry

The registry 15 allows web services 100 to be discovered through the use of standard search categories. The platform 10 provides the selection of appropriate web services 100 through its classification into various categories. Additionally, it allows multiple implementations of a single interface with standard method calls as defined by an enterprise's data and/or service policies. By this aspect, the implementations and endpoints of these implementations are abstracted away from

the invoking client systems and applications 50, enabling an even more loosely-coupled approach by only binding to the service interface. Endpoints of these services can be discovered at design-time and/or run-time, improving flexibility and reliability to the development of the enterprise applications by ensuring that there is no failure should any of the endpoint addresses change for any reason.

The registry 15 is a web-based business directory and a technical specifications library. Registration allows:

a) search web service interfaces of business partners and potential partners;
b) discover technical details on working with other web services 100 and post these details.

The directory specifies a standard way for business and services to list themselves on the Internet and discover one another. The registry creates a platform independent, open framework for describing web services 100, discovering businesses, and integrating business services using the Internet as well as an operational registry that is available today.

2) Routing

The routing engine 20 manages routing of web services 100 to different end points based on a set of rules. This allows web services 100 to be configured dynamically to adapt easily to business changes. Web services 100 may be routed according to any one of the following rules, in any combination: priority based queues to meet Service Level Agreement (SLA) standards; load balancing comprising load, fastest response time, or round robin algorithms; geographical proximity to enhance performance; or content based routing based on header content of a message. In one embodiment, the routing engine 20 is a core module in the management engine 25.

3) Web Service Management

The management engine 25 handles end point translation for web services 100, load balancing of web services 100, performance monitoring, error handling and recovery, policy and security enforcement.

Business processes are exposed as web services 100. When these web services 100 are written and tested, they are registered with the service registry 15. The web services 100 are made available to the public, to a selected list of partners or

secured only for use within the company. Client applications 50 search the registry for relevant services and register to use these services. The usage of services is subject to various licensing arrangements including QoS, and SLA between the service provider and consumer.

To use the web service 100, the client queries the registry 15 from the client application 50 to request for the web service 100. The platform 10 checks if the client is a registered end user, and then checks the licensing agreement with the client. Based on these parameters, the best service 100 is selected and a reference end point is returned to the client. The client 100 uses this end point to invoke the service 100. The platform 10 accepts the request and performs the following:

a) if there are multiple instances of the web service 100, each instance conforming to a different SLA, it selects the best service 100 which is the most appropriate;
b) if there are multiple instances of the web service 100 that are load balanced, it selects the best option based on various load balancing algorithms;
c) if there are multiple versions of the same service 100, the most appropriate service 100 is selected;
d) it then queries its service status monitor to check the status of the service 100. This includes the performance of the service 100, whether the service 100 is operational, and whether the service 100 meets the criteria set in the SLA. In the event that the minimum criteria is not met, an error message is sent to the client enabling systematic termination;
e) once the quality of the service 100 is determined to meet the required standards, the request is then examined based on the requirements of the service 100;
f) the data that is sent on the request may need to be transformed to conform to a different schema. This is provided by the system;
g) the WS-security and WS- policy requirements of the web service 100 are then verified. If the request does not meet the criteria set by the service 100, an error message is sent to the client and the program terminates;
h) if the service requires the request to be routed to a different service 100 or end point, the routing engine 20 handles the routing;
i) after the request and the service 100 are subjected to the verification, the request is forward to the actual web service 100. The response is collected from that service 100 and subjected to verification;

j) the response is then returned to the client.

The platform 10 abstracts the service 100 from the client application 50. By doing this, new versions of the service 100 can be created or the service 100 can be moved to a different location without breaking the client's application 50.

The platform 10 allows the service provider and client to agree upon an SLA and monitors if the SLA has been breached. The platform provides load balancing of multiple web services 100.

The platform 10 provides support for WS-security and WS-policy. The platform 10 has reporting capabilities to provide activity, performance error statistics of the services. The platform 10 monitors service status to enable smooth termination of any service requests. The platform 10 is able to support new standards when they are available for WS-security, WS-addressing and WS-policy.

Maximise re-use of existing systems and services

The platform 10 facilitates a simple transition of silo-deployed services of an organization to a fully integrated service-oriented architecture where optimal re-use of existing services 100 is endeavoured. This is achieved by allowing existing systems and services 100 to register themselves into the registry 15 so that they can be found and discovered by others using open industry-accepted standards. All messages passing via the server are inspected to ensure that they meet all requirements and conform to the policies of the enterprise.

Existing systems and services in the enterprise are left in tact and the platform 10 serves as a main message bus for these systems and services 100 with industry-accepted and open standard protocols. The service registry 15 allows all services 100 to be registered and categorized according to an enterprise's category scheme or taxonomy. Searching these services 100 is facilitated with a rich graphical user interface of the registry portal 15 or through a standard set of Application Program Interface (API) calls. Accordingly, the platform 10 enables services 100 to be easily discovered and once they are discovered, they are easily re-usable. Services are developed with no single specific client application 50 implementation. The following are to be abstracted from the invoking client application 50 to the extent possible:

a) service implementations;

b) endpoints of these service implementations;
c) evolving web services standards; and
d) enterprise policies

The return on investment for an enterprise adopting web services 100 is significant if they adopt a service-oriented attitude. The enterprise is able to move from an application-specific approach towards a more service-oriented approach.

The platform 10 acts as a central, abstracted and encapsulated message layer that is able to integrate heterogeneous systems. The platform 10 is the central messaging layer in the enterprise. This requires it to be capable of functioning under constant heavy load conditions and therefore the platform 10 is designed for high availability. The availability of the services 100 registered with the system can also be increased based on the QoS levels and SLAs.

Turning to Figure 1, any implementation of a service orientation has three major participants: the service layer 7, the messaging layer 6 and the application layer 5. The messaging layer 6 represents the platform 10 while the service layer 7 and the application layer 5 are the participants. The service layer 7 includes organizations that share their processes through web services 100 and the application layer 5 includes organizations that consume these services 100 in their applications 50. For example, Organisation A, B and C may be implemented within a single organization and represent business units or departments.

Referring to Figure 2, all requests and responses are directed through the platform 10. The platform 10 manages the request and response of message and associated functions.

The service registry 15 supports the use of taxonomies to group services 100. This allows users to easily search and find web services 100. All web services 100 registered with the platform are registered within the service registry 15. The service registry 15 also stores information on the Web Services Definition Language (WSDL) interface, the data schema, the policies, the owners of the service 100, and the number of instances of the services 100 that are available.

Applications 50 consuming these web services 100 do not directly bind to the web services 100. They provide the service registry 15 with a binding key using what

the service registry 15 determines is the best service instance to invoke. The loose-coupling enforced by the service registry 15 enables service providers 15 to move their web services 100, add or remove instances without affecting client applications 50.

WS-security describes how to attach signature and encryption headers to SOAP messages. In addition, WS-security describes how to attach security tokens, including binary security tokens such as X.509 certificates and Kerberos tickets to messages.

Agents for the platform 10 are installed in participating organizations to seamlessly integrate WS-security into their applications. The WS-security module along with the agents provides the following features:

a) authentication. While WS-security provides an infinite number of authentication mechanisms. This architecture addresses three methods – username/password, Public Key Infrastructure (PKI) through X.509 certificates and Kerberos;
b) digital signature. When a message is signed, it is nearly impossible to tamper with the message. Message signing does not protect the message itself from external parties seeing its contents. Using the signature, the receiver of the SOAP message knows that the signed elements have not changed en route.
c) encryption. Using the WS-security specification, the platform 10 provides encryption services using XML encryption. Both symmetric and asymmetric encryption is supported.

The agents handle signing, encryption and decryption of messages. This frees applications 50 from handling these functions at an application level. Based on the policy requirements, WS-policy, of the service provider and the consumer, message may be signed and/or encrypted using a private key that is known only to the agents and the platform 10. For example, a policy requires applications 50 to encrypt request messages and services to encrypt response messages, it is handled by the agents located both at the service and consumer side. When a request for a service is sent to the platform 10, it is intercepted by the consumer side agents. These agents check the policy and discover that the request message needs to be encrypted. The agent encrypts the message and forwards it to the platform 10. The platform 10 forwards the request to the service 100, which is

again intercepted by the agents on the service side. The service side agents decrypt the message before sending it to the service 100 for processing.

Before applications are able to consume services 100 through the platform 10, they have to provide a unique key to the agents for authorization. This unique key is generated using a utility that is provided by the platform 10.

The platform 10 enforces service providers to submit WSDL schema and data schema documents. The platform 10 uses the WSDL document to ensure service implementations conform to specification. The data schema verifies whether the incoming data and outgoing message conform to specification. Performing these validations at a central location allows applications 50 and services 100 to concentrate on their business processes and business logic.

Certain legacy systems and other enterprise software are unable to conform to the format specified in the data schema. Rather than customizing the data transformation for every application 50, a data transformation module allows applications 50 to handle this at a central location. A designer is provided in the Organisation Administration Console to create transformation rules for incoming data and outgoing messages. The console allows transformation into various industry standards.

All incoming and outgoing messages, authentication, request and response failures and changes made by at the administration console are logged and audited by the system. Administrators are able to view usage data, the performance statistics and other information of their services. The system also allows administrators to generate reports on these audit logs. The audit log also provides information on whether services 100 maintained their minimum performance requirements as specified in the SLA.

Access control is supported at two levels: the service level and data schema level. At the service level, administrators are permitted to use their administration console to allow/disallow specific organizations from accessing their services 100. At the data schema level, organizations may elect to share information depending on the requesting organization. For example, if the request is from organization A, then all information is shared. However, if the request is from organization B, only partial information is shared. The administrator uses the administration console's

designer to specify the rules. The rules are dynamically configured and do not require system downtime or application level customization. Thus, deployment, updates and maintenance can occur in real time. Responses sent to organizations are verified against the rules specified and the access control module filters any information that is not to be sent to the requesting organization.

The platform 10 provides a version management module. Service providers can introduce a new version of an existing service 100 without breaking existing client applications 50. Service providers can specify rules to enable client applications 50 to move from one version to another transparently. If two versions of a web service 100 co-exist, administrators can specifically allow this to happen using the administration console.

To address performance and latency, the platform 10 supports caching at three levels: real time, central cache and local cache. Real time does not involve any caching. Data is always served in real-time. In central cache, data is cached to a central database. All requests for data are served from the central database through the use of the central web services 100. In local cache, data is cached to organizations through the use of batch jobs. Organisations then internally access their own local cache. Administrators can switch from one caching model to another using the administration console.

The platform 10 enables enterprises to adapt seamlessly to business changes and evolve as business paradigms change. By shifting the focus to business processes and providing a centralized management architecture, the platform 10 fosters reusability, extensibility and a blueprint for organisations to derive true value.

It will be appreciated by persons skilled in the art that numerous variations and/or modifications may be made to the invention as shown in the specific embodiments without departing from the scope or spirit of the invention as broadly described. The present embodiments are, therefore, to be considered in all respects illustrative and not restrictive.

WE CLAIM:

1. A platform for integrating an enterprise, the platform comprising:

a service registry to enable client applications to search for registered web services;

a routing engine to route the web services selected by the client applications to end points according to at least one configurable routing rule; and

a management engine to manage the web services after they have been invoked;

wherein the web services are abstracted from the client applications to enable platform independence and the at least one configurable routing rule is dynamically configurable.

2. The platform according to claim 1, wherein the configurable routing rule is a rule selected from a group consisting of: priority based queues to meet Service Level Agreement (SLA) standards; load balancing comprising load, fastest response time, or round robin algorithms; geographical proximity to enhance performance; and content based routing based on header content of a message.

3. The platform according to claim 1, wherein the management engine performs security, version control, access control, end point translation, data schema translation and load balancing for the web services.

4. The platform according to claim 1, wherein the management engine handles security, recording audit logs and caching.

5. The platform according to claim 1, wherein the platform defines a strategy to implement adaptive web services.

6. The platform according to claim 1, further comprising agents to integrate WS-security into the client applications.

7. A method for selecting an optimal web service for a client application, the method comprising:

checking a service level agreement to obtain minimum service levels for a plurality of web services; and

instructing a routing engine to use a load balancing process to select the optimal web service from the plurality of web services according to the quantity of service instances deployed and maintains a predetermined level of quality of service.

8. The method according to claim 7, further comprising the step of determining the geographical proximity of the client application to the routing engine to increase the performance of the web service.

9. The method according to claim 7, further comprising the step of categorizing geographical locations of web services in the service registry.