International Journal of Technology and Engineering System (IJTES)
©gopalax-Publisher of International Journals
ISSN: 0976-1345 pp. 7-11
available at : www.ijcns.com
Vol 3.No.1 Jan- March 2012

Service Oriented Architecture using ISO RM-ODP with respect to Engineering and Technology Viewpoints

C. Madana Kumar Reddy[1], **Dr. A.Ramamohan Reddy**[2]
[1]Associate Professor of M.C.A, Annamacharya Institute of Technology and Sciences,Rajampet, Kadapa(dt), AndhraPradesh,c_mkr@yahoo.co.in, Mobile Number-91+9441105151
[2]Professor and Head of CSE Department, Sri Venkateswara University College of Engineering, S.V.University, Tirupati, ramamohansvu@yahoo.com

Abstract—**The proposed Service Oriented Architecture using ISO Reference Model for Open Distributed Processing is a high performance technique for providing effective services to the users. This architecture consists of five different viewpoints namely Enterprise viewpoint, Computational viewpoint, Information viewpoint, Engineering viewpoint and Technology viewpoint. This paper discusses the details about the Engineering and Technology viewpoints. Engineering viewpoint gives the details related to web services composition, service chains, choreographies, exception handling, service clustering and patterns in Service Oriented Architecture. The main concerns of this viewpoint are communication, computing systems, software processes and the clustering of computational functions at physical nodes of a communications network. In *SOSA* several services need to cooperatively process one message, the failure of one service results in a system failure. The patterns address common security problems related to Web services. It also gives the various problem situations and its solutions. The Technology Viewpoint focuses on the technology aspects related to the system and its environment. It describes the hardware and software components used in the distributed system together with the infrastructure which allows distributed components to communicate.**

Index Terms—performance, Engineering viewpoint, choreographies, exceptions, Service, Technology viewpoint

1. INTRODUCTION

These viewpoints describes the main concerns. These concerns include how the different web services are composed, service chains, choreo-graphies, exception handling, service clustering and patterns in Service Oriented Architecture. Each service implements only a part of the overall functionality, in a similar fashion to database systems. In which it is necessary to execute several services as one single logical operation. This is realized by means of transactions. Choreographies are realized through message routing. Technology viewpoint describes the hardware and software components used in the distributed system for this Architecture.

2.ENGINEERING VIEWPOINT

According to the RM-ODP, the Engineering Viewpoint describes the system as a network of com-puting nodes. The main concerns of this viewpoint are communication, computing systems, software processes and the clustering of computational functions at physical nodes of a communications network [1]. The computational nodes in our system are security services. This section will describe how these services can be combined, how they communi-cate with each other and how they can be distributed.

A. Web Services Composition

Web services composition (aggregation) refers to the composition of multiple Web services in a process flow. Such processes are described in terms of exchanged messages, business logic and the order of execution for interactions. They can range from simple ones (e.g., one Web service calls another) to very complex ones which span several applications and organizations and result in long lived, transactional, multi-step interactions. There are two models for aggregating services [2]: choreography and orchest-ration. The difference between the two is that in orchestration the execution is controlled centrally from a single entity. Whereas, choreographies are more collaborative in nature, and each party involved in the process controls a part of the execution. Because each of the approaches has its advantages, both of them are valid ways for composing security services.

B. Service Chains

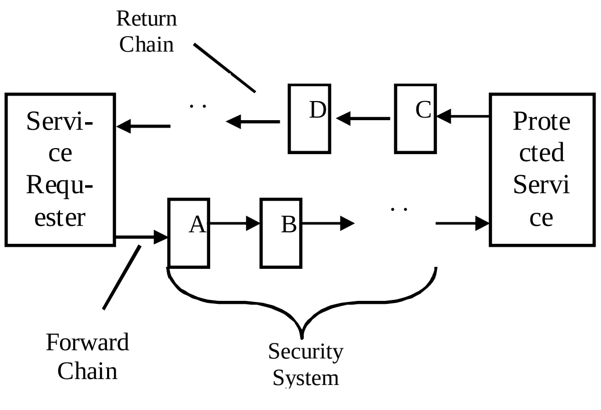


Figure 1. **Service Chains**

Messages originating from the requester are travelling along the forward chain to the protected service. After processing the request, the service generates a response message which travels along the return chain back to the requester. The forward and return chain may also contain orchestrations which are exposed as services.

C. Realization of Choreographies

Choreographies are realized through message routing. The two patterns that can be used for the realization of choreographies are itinerary routing and content-based routing.

Itinerary Routing

In itinerary routing a routing slip is attached to the message describing the route that the message should follow. The routing slip fully describes the choreography. In an execution environment, after a service has processed the message, it is the responsibility of the messaging middleware to dispatch the messages according to the routing slip. A gateway service would receive the message from the requester, inspect it and attach a routing slip describing both the forward and the return chains. In this way, the protected service is aggregated together with the security services into one choreography. The advantages of this approach are that the choreography is centrally described at the gateway service which is the one entity attaching the routing slip. This is very convenient for the management and administration of the whole system.

Content-Based Routing

In the case of content-based routing, after a message is processed by a service, it will be inspected and, depending on its contents, the next service along the chain is determined. In networking, this kind of routing is called next-hop routing, as each node determines the next destination of the message, based on some internal routing table. Here the choreography is specified through routing tables which are distributed (each router has its local routing table). The choreography is managed in a distributed fashion.

Mixed approaches

In this case parts of the choreography are centrally described through itineraries, while the rest is specified through routing tables. This is a good way to combine multiple choreographies together or to handle exceptions.

D. Transactions

Each service implements only a part of the overall functionality, in a similar fashion to database systems. In which it is necessary to execute several services as one single logical operation. This is realized by means of transactions. Atomic operations can be implemented by means of transactions which are supported at the middleware layer. Because security services run on top of the middleware, they can make use of transactions. The framework contains two specific coordination types: WS-Atomic Transaction for short duration [3], ACID transactions and WS-Business Activity for longer running business transactions [4].

E. Exception Handling

Security services may throw exceptions if they cannot fulfill their tasks. Because exception messages are distinctly marked, the messaging middleware can take special action. The simplest strategy for exception handling is to return the exception messages back to the requester. Because they contain information regarding the failure reason and details about it, the requester can fix the problem and resend its request. However, more complex strategies can be applied such as routing exceptions to a central exception-handling service, that implements exception recovery strategies, releases any resources associated with the request, unrolls any transactions pending and further processes the exception message (through either enrichment or filtering) before sending it back to the service requester.

F. Service Clustering

In *SOSA* several services need to cooperatively process one message, the failure of one service results in a system failure. It is most probable that some services will spend more time in processing a message than others. This results in some services being performance bottlenecks for the whole system. To address these issues, and still profit from the *SOSA* model, solutions available for server clustering can be deployed. In such a design, several services are configured to appear as one single logical service. Show in Figure 2.

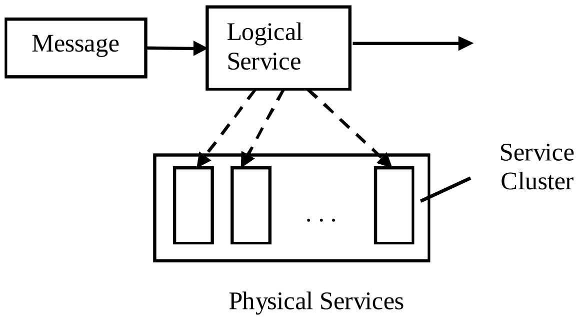


Figure 2. **Service Clustering**

This technique can be used for enhancing availability, scalability or both. Clustering is usually supported both at the networking layer as well as at the middleware layer. The security services can easily take advantage of these features.

G. Patterns in SOSA

The patterns address common security problems related to Web services. The purpose of this is to document how security services can be designed and coupled together in order to build a *SOSA* security solution. Because of the loosely-coupled nature of the system, it is easy to assemble together several patterns and build complex security solutions. Each pattern has described in terms of the context, problem, solutions, variations, and benefits.

i. The Gateway Pattern

Context: External applications require access to one or more Web services which are deployed inside the private network. The access to these services is restricted to authenticated users. External applications should not be able to access or determine the existence of services and other resources deployed inside the private network.

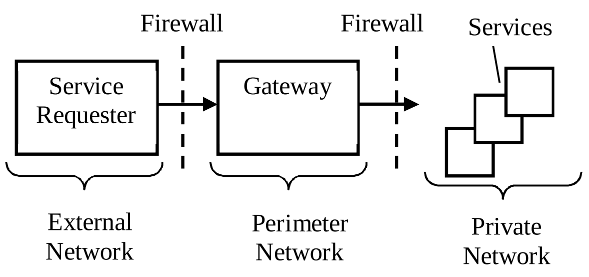


Figure 3. **Gateway Pattern**

Problem: How to make Web services in the private network available to external applications, without exposing the other resources?

Solution: The solution involves a Gateway service located at the perimeter of the network, in a demilitarized zone. This pattern builds on the perimeter security idea.

The Gateway Service is the only entity visible from the external network. The protected service together with the rest of the security services are hidden from applications running in the external network. The tasks of the Gateway service are as follows:

- Transform the message into the canonical message format used by the security services with annotations.
- Verify security relevant information provided by the transport protocol (i.e. IP, SSL, HTTP, etc).
- Determine the forward and return chains for the given request message.
- If the security system is implemented using asynchronous messaging and the protected service requires a request-response message exchange pattern, the Gateway must correlate the request messages with the responses and deliver the response to the requesting party.

Benefits:

- *Concentrate Security Administration*: The rules describing which security services need to be invoked for which type of requests are centrally managed by the Gateway service. The Gateway Service acts as a single point of administration for the security policy.
- *Multiple Transport Protocols*: It provides the same service on a variety of transport protocols. By this, the Web service logic and the security implementation are decoupled from the transport protocol on which the Web service is being offered.

ii. The Federation Pattern

Context: A protected Web service must accept users which are registered in several security domains. Each domain contains a user management system which stores attributes about the registered users. These attributes are required during the security check process. External applications send requests to the protected service that contain some kind of authentication credentials. However, from the credentials, it is not possible to know the domain where the requester is registered.

Problem: How to retrieve the attributes of the requester from the home domain?

Solution: Because the user's home domain is not known, all identification services must be invoked. In order to optimize this operation, the splitter-aggregator pattern is used (Shown in figure 4). A splitter broadcasts the message to all identification services. The execution of these services is done in parallel, thus optimizing execution time and minimizing failure. An aggregator service joins the messages together by simply dropping the messages containing failures.

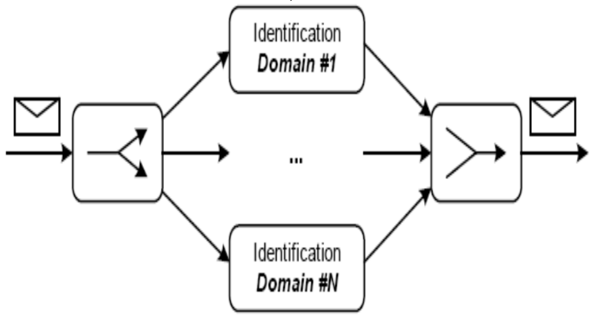
Figure 4. Federation pattern - without knowing user's home domain

Explicit Federation: In the case of an explicit federation, it is possible to determine the user's home domain from the provided information; a content-based router can be used instead of the splitter (shown in fig. 5). The router determines the home domain and routes the message to the associated identification service. Because the message is not split, there is no need for an aggregator service.

Figure 5. Federation pattern - knowing user's home domain

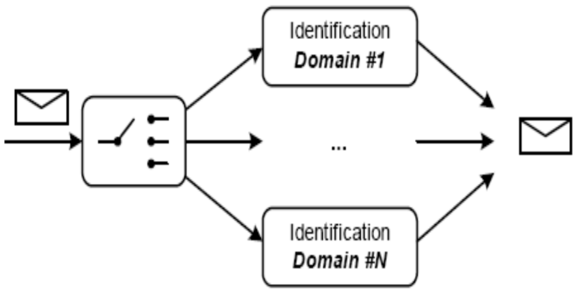
Benefits:

- Several domains are aggregated together in a manner totally transparent for the user.
- The solution is optimized in terms of time (execution is parallel) and fault tolerance (if the identification service of one domain is down, the users in the other domains are not affected).



- Several authentication methods (password, digital signature, etc.) can be supported by delegating the verification process to different verification services.
- The identification service receives the whole message, including the service request and the schedule of the message. It is possible to build identification services that inspect the service request and the schedule of the message and, based on this information, respond with different attributes or attribute values. Such services are able to enforce privacy on behalf of their users.

iii. Sequential Decision Making Pattern



Context: The decision as to whether or not a given Web service can be accessed in a certain context cannot be taken in a single place or by a single entity. One justification for this could be the fact that the access

permissions are not all stored in a single place, and in order to grant the access, the complete set of permissions is required. Another scenario could be one where the resource is under the authority of several security domains and, in order to take the access decision, the cooperation of several entities is required.
Problem: How to have messages authorized by multiple entities?

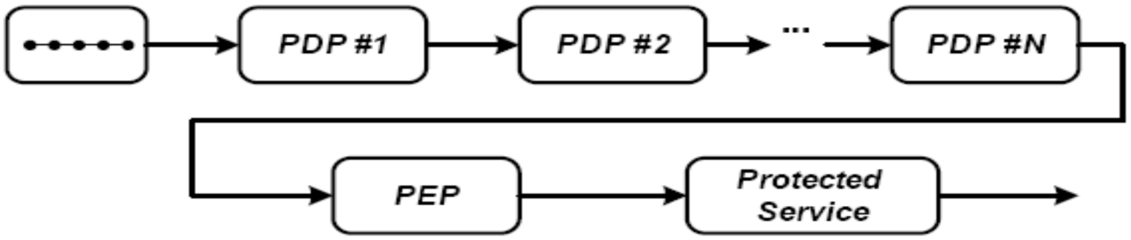


Figure 6. **Sequential decision making pattern**

Solution: Several Policy Decision Point (PDP) services are deployed, one for each place where messages need to be authorized. A single Policy Enforcement Point (PEP) is deployed, because the enforcement should be as close as possible to the resource being protected. The PDP services, the PEP and the Protected Service are chained together by means of schedule routing. Messages are routed such that they sequentially pass through each of the PDP services. The PDPs inspect each message they receive and annotate it with their decision and obligations.
Benefits:

- Several entities can collaborate in taking an access control decision;
- Enforcement is done in a single place, close to the resource;
- Through obligations, requirements can be specified. These requirements can refer to decisions taken by other PDPs or the action of other security services.

3. TECHNOLOGY VIEWPOINT

According to the RM-ODP [10], the Technology Viewpoint focuses on the technology aspects related to the system and its environment. It describes the hardware and software components used in the distributed system together with the infrastructure which allows the distributed components to communicate [11].

Because SOSA is a security system for Web services and because it is an architectural design, several implementations are possible. The purpose of the Technology Viewpoint is therefore not to present the implementation of the system, but rather to demonstrate how such a system can be built and to show how the most important aspects regarding SOSA can be realized by means of Web services technologies.

A. Standards

There are several possible realizations for SOA, for messaging systems and also several possible realizations for Web services. An implementation for SOAP messaging was considered desirable as it offers interoperability with other platforms.
SOAP All messages, including the communication with the service requester, the service provider and the communication internal to the security system are SOAP messages. Message routing is implemented based on the processing model defined by SOAP (SOS!e 1.0 and 2.0).
WS-Security: Security tokens are encoded and attached to messages by means of WS-Security.
BPEL: Service orchestrations are described in BPEL and executed through BPEL runtime-engines (***SOS!e* 1.0** & **2.0).**
WS-Addressing: This specification provides enhances SOAP messaging with addressing capabilities that are independent of the transport protocol. These are leveraged by the ESB middleware in SOS!e 2.0 to deliver messages to services accessed through different transport protocols.

B. SOS!e 1.0

SOS!e 1.0 was derived from SOSIE(Service Oriented Security - an Implementation Experiment).
Software Platform, Libraries and Tools
The security framework was implemented in Java using open-source software. Apache AXIS 1.x was chosen as Web services container as it was the state-of-the-art implementation at the time. Subsequent versions of this software were used: 1.2, 1.3 and finally 1.4. This had several drawbacks, among them the fact that AXIS 1.x has limited support for asynchronous messaging.

Several other open-source libraries are also used by the implementation, the most important to mention being the following: Xerces 1.4.4 parser (http://xerces.apache.org/xerces-j), Xalan 2.7.0 for XPath evaluations (http://xml.apache.org/xalan-j), WSS4J 1.5.1 as the WS-Security implementation(http://ws.apache.org/wss4j), OpenSAML 1.1 as the SAML implementation (http://www.opensaml.org), Log4J 1.2.9 and Commons-HttpClient 3.0.
Security Services Realization and Deployment
The implementation provides a middleware layer upon which the security services can be built. Figure 7 shows how the security services are running inside the SOS!e middleware, which is deployed as a Web service inside Apache AXIS, which instead runs as a servlet inside the Tomcat servlet container.

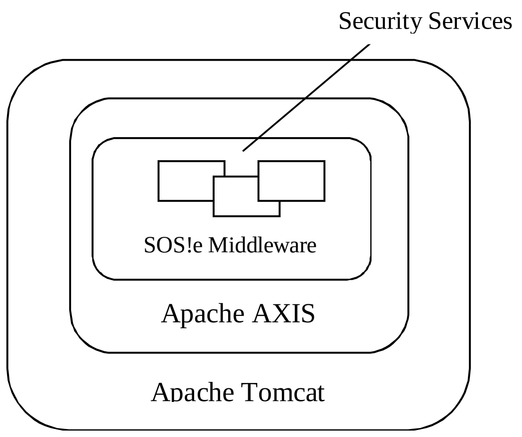


Figure7. Realization and deployment of security services

C. SOS!e 2.0

After experimenting with SOS!e 1.0 , it was decided that a complete rewriting of the framework is necessary in order to test some new concepts and implementation possibilities. The major driver for this was the development of Enterprise Services Bus (ESB) software - several implementations, both commercial and open-source, made their way into the market. As the part of it SOS!e 2.0 was developed.

Developing Custom Security Services in SOS!e 2.0

Several security services were implemented on top of the SOS!e 2.0 framework. These implement common security tasks such as authentication by means of user-name password against an LDAP directory, simple authorization, audit, accounting or charging by means of PayPal.

As far as developing custom security services, the mechanisms built in SOS!e 2.0 are similar to the ones from the first version of the framework. An AbstractService class is defined which must be extended by all security services. This class implements the functions for processing annotations (retrieval, creation, deletion, modification).

4. CONCLUSIONS

This architecture is a high performance, Service Oriented Architecture to support the geospatial data with scientific applications. The scope of this paper is only up to the details of the Engineering and Technology viewpoints of the Reference Model for Open Distributed Processing. It described the various patterns which are possible in this architecture along with the service chains, transactions, exception handling and the clustering.

REFERENCES

[1] OGC Reference Model. Open Geospatial Consortium (OGC), 2003. Version 0.1.3, http://www.opengeospatial. org/specs/?page=baseline.

[2] Chris Peltz. Web services orchestration and choreography. Computer, 36(10):46– 52, October 2003.

[3] Web Services Atomic Transaction (WS-AtomicTransaction). IBM, BEA Systems, Microsoft, Arjuna, Hitachi, IONA, August 2005. http://www-128.ibm.com/developerworks/library/specification/ws-tx.

[4] Web Services Business Activity (WS-BusinessActivity) IBM, BEA Systems, Microsoft, Arjuna, Hitachi, IONA.

[5] Hashimi, S., Service-Oriented Architecture Explained. 2004, O'Reilly

[6] Peng, Z.R. and M. Tsou, Internet GIS: Distributed Geographic Information Services for the Internet and Wireless Networks. 2003: Wiley.

[7] Madana Kumar Reddy,C. (2009). Operating Systems Made Easy. New Delhi: University Science Press.

[8] ISO/IEC 2382-01: Information Technology - Open Distributed Processing - Use of UML for ODP system specifications. International Standards Organization (ISO), committee draft v02.00 edition, May 2005.

[9] Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, and Henrik Frystyk Nielsen, editors. SOAP Version 1.2 Part 1: Messaging Framework. World WideWeb Consortium (W3C), June 2003. Status: W3C Recommendation.

[10] ISO/IEC 10746-1: Information technology - Open Distributed Processing - Reference Model: Overview. International Standards Organization (ISO), 1998.

[11] OGC Reference Model. Open Geospatial Consortium (OGC), 2003. Version 0.1.3, http://www.opengeospatial. org/specs/?page=baseline.