Modeling Security as an Aspect in AO4BPEL for Service Oriented E-Vehicle System

P.C.Dinesh
Assistant Professor
[1]Computer Science and Engineering, Kalasalingam Institute Of Technology – KIT, Virudhunagar, India
[1]pc.dinesh5@gmail.com

***Abstract*—Service Oriented Architecture (SOA) is a framework to enhance the use of existing web service into the application. Business Process Execution Language (BPEL) is an XML language used for web service composition and implementations. However, BPEL exhibit some limitations in providing security in web service composition. Companies will get satisfied only if their requirements of authentication, confidentiality and integration are fulfilled. In this paper, the main process is to weave security dynamically as an aspect in AO4BPEL, an aspect oriented extension to BPEL. Here, the Service Oriented E-Vehicle system is taken as an example for implementing security aspect during runtime.**

***Index Terms*— SOA, AO4BPEL, AOP, WS-Security.**

I. INTRODUCTION

Web service composition [1] is a collection of web services that are published by various service providers. The services are composed according to clearly defined rules in such a way that they provide us with a more complex service. The composition languages follow valid work-flow rules, whereby they are defined as a work flow process. These rules determine the web services that participate in the composition and the order of control-flow. The data is transferred between the process activities (data-flow).

Aspect Oriented Programming (AOP) aims on enhancing the modularity and maintainability of the system, and consequently influencing the evolution of the software. It helps in separation of core concerns and cross-cutting concerns. AOP provides separation of cross-cutting concerns by introducing an aspect that cross-cut the other module. AOP is a programming methodology that is part of the overall paradigm of Aspect Oriented Software Development. It is possible to introduce AOP in Business Process Execution Language [2] to overcome the drawback of capturing web service composition in a modular way and for dynamic change.

An Aspect Oriented Extension to BPEL (AO4BPEL)[2] is the most popular language for web service composition which has capability to accept widely in the industry. It captures web service composition in a modular way and the composition becomes more open for dynamic change. BPEL allows the specification of interactions among the web services that participate in the composition according to various control-flow patterns. It provides three activities for web service interaction: <invoke> for invoking an operation on a partner web service, <reply> for sending a response to a client, and <receive> for blocking until a client request is received. These activities produce a message-based interaction between the composite web service and its partners. They also specify the functional logic of a composite web service. These activities are done in XML process such that it affects the composition logic, these XML file contains the above said interaction process. However, beyond the functional aspects, non-functional composition properties such as security, reliability, and quality of service are of importance for web service composition languages to keep their promises.

In this paper, we present a framework for integrating the specification of security features into the specifications of web service compositions. The proposed framework consists of security service, a process and a deployment descriptor. The security service is an aspect based web service that provides operations to secure the interactions of the BPEL[3] process with its partners and clients. This service provides functionality that is used by the web service orchestration engine to authenticate, give integrity, prevent repudiation, or make BPEL activities confidential. Apache ODE (Orchestration Director Engine) software executes business processes written following the WS-BPEL standard. It talks to web services, sending and receiving messages, handling data manipulation and error recovery as described by your process definition. It supports both long and short living process executions to orchestrate all the services that are part of our application.

WS-BPEL (Business Process Execution Language) is an XML-based [4] language defining several constructs to write business processes. It defines a set of basic control structures like conditions or loops as well as elements to invoke web services and receive messages from services. It relies on WSDL to express web services interfaces. Message structures can be manipulated, assigning parts or the whole of them to variables that can in turn be used to send other messages.

The security process which wraps BPEL processes provides transparent security the users of the composed web service. The process container is implemented as a set of aspects that are specified in AO4BPEL, which is an aspect-oriented extension to BPEL. The XML-based deployment descriptor specifies the requirements of the process activities along with the function of security features.

Signcryption security [5] technique is implemented and dynamically weaved in a SOA application. This security service is non-functional component that can be plugged or unplugged during runtime. During process deployment time, the deployment descriptor file has to be specified in addition to the BPEL configuration.

The remainder of this paper is organized as follows. Sec.2 explains the System Architecture of E-vehicle System. In Sec. 3, we briefly describes the scope of aspect service in E-vehicle System. Sec. 4 gives a short overview of Security framework design and its implementation. We conclude the paper and outline areas for future work in Sec. 7.

II. System Architecture

The web services that are created initially are composed in the BPEL process using the work flow model and executed in Apache Orchestration Directory [6] engine. The additional functionality implemented in services can be modeled as an aspect and weaved into the existing BPEL process [7]. This is achieved using run time using aspect weaver. The Aspect Oriented BPEL process is created, providing an efficient modularization of core concerns and cross-cutting.

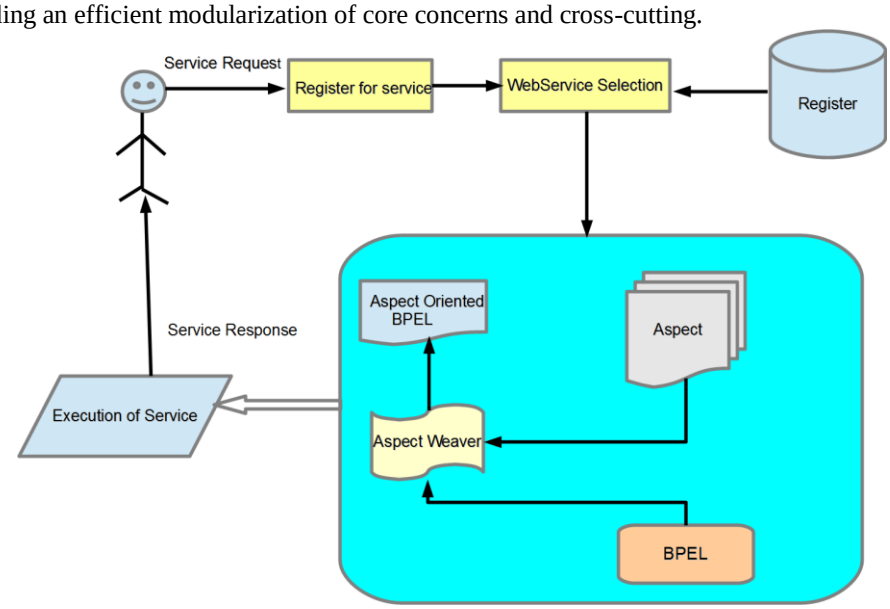

Fig. 1. System Architecture.

It also supports changes of composition logic during the run time of the BPEL[8] process. The AO4BPEL process is done by the promoters in web service composition, they will also work with public relations personnel to increase public awareness of the company and the products it offers. Often, there are a number of projects in operation simultaneously, with some promotions being long-term and general while others are short-term and specific. The work-flow for implementing this task, is clearly depicted in Figure 2.

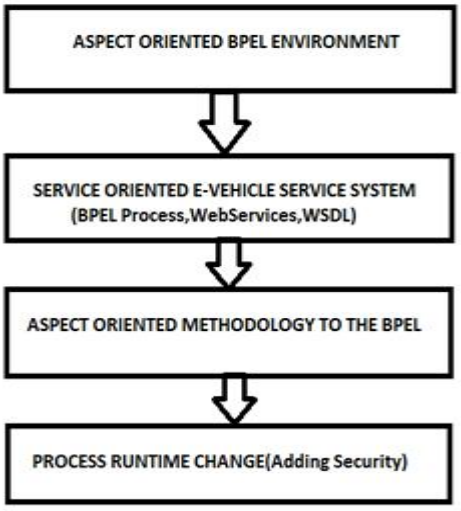

Fig 2: Work flow for implementing the task

Dynamic reconfiguration of web service is implemented in an Aspect Oriented Environment. In Service Oriented E-Vehicle system the offered web services are composed according to the promoters need. The additional functionalities or new services are implemented as an aspects and weaved at run time, this information can be obtain in the registry where the promoters add aspects based on the needs of the Fig. 2. Work flow for implementing the task consumers. They compose web service along with the aspect dynamically in the composition[9]. The woven aspects can be unwoven at any time by the promoters based on the changes in the work-flow process.

Adding Security features in a SOA application, Where security is a non functional components which may not needed in this application even though some clients needed this Security web service so this service is provided with Aspect features.

III. Application Scenario

The effective communication between customer, mechanic and service station plays a vital role for providing good vehicle service for the customer vehicle. Currently the communication mechanism is based on an elaborate paper work which is not efficient in the modern era of electronic book keeping for processes. The problems with this kind of communication mechanism are,

- Service station may not understand the parts written by the Mechanic.
- The mechanic may not have clear history information about the service of the vehicle done.
- If anyone who knows the URL of the service or finds it in a UDDI registry they can place orders or perform bank transfers.

In order to provide the effective communication between the trio(customers, station, mechanic), the concept of Service Oriented e-vehicle system is introduced to the domain.

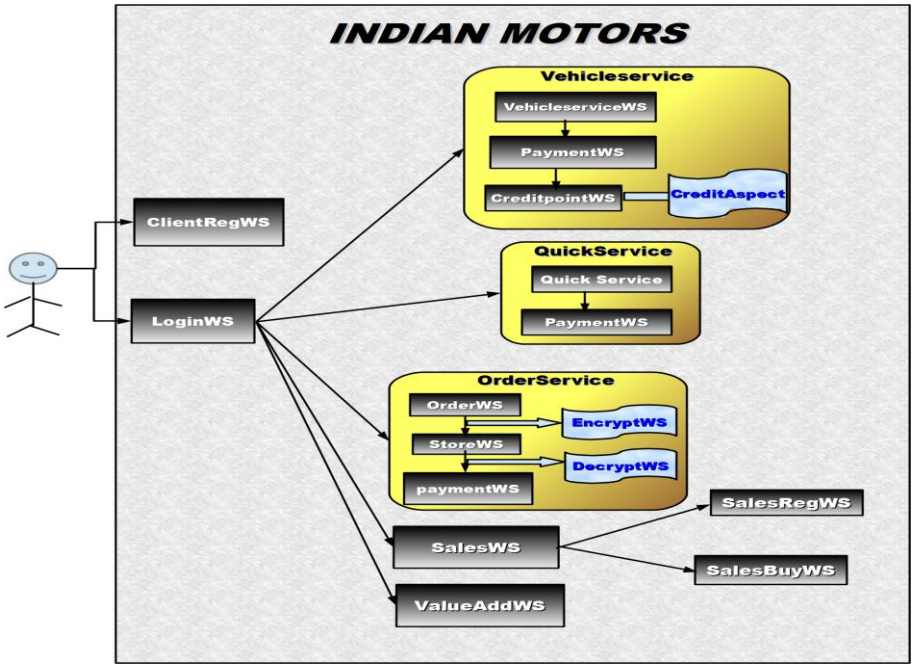

Fig. 3. Web Service Composition in E-Vehicle Service.

In the system, the registration, appointment, availability of services, billing are all done electronically. An UniqueID is generated for each customer, who is registering his credentials with the system. Security is provided by extending the features of WS-Security and WS-Trust protocols to set of services offered in the proposed system. Security features are modeled as an aspect and dynamically recomposed during the runtime of the application.

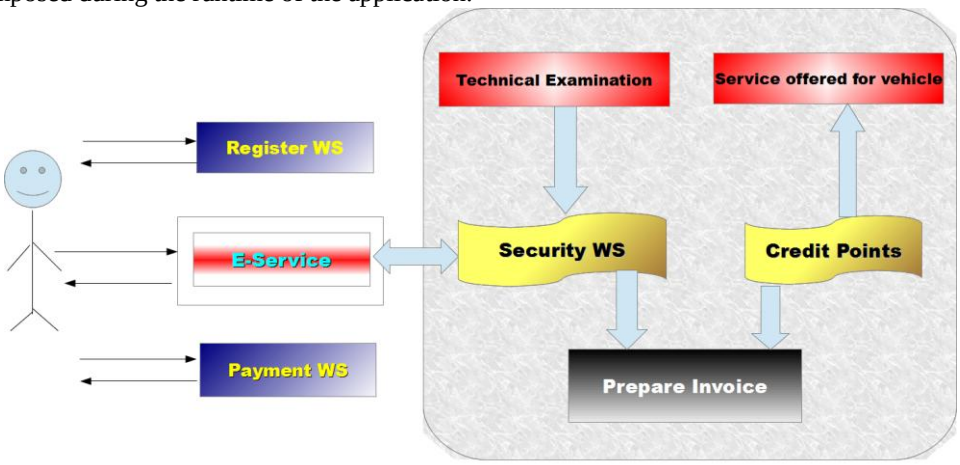

Fig. 4: Aspect based Service Oriented E-Vehicle Service system

The task involved in the project are, web service creation, web service composition and deployment of aspects. These tasks are implemented using specific modules in the application.

Web Service Creation

The following are the list of services that were created for e-vehicle service system:

a) RegistrationWS - This Service registers the customer details and provides unique Id to the customer.
b) NeededPartsWS - This service helps the mechanic to suggest the parts to the customer and sends back control to the Repair Service. It also returns the payment made by the customer to mechanic.
c) SurveyPartsWS - This service returns the best Manufacturer for the given parts.
d) StockVerificationWS - This service checks the availability of the given parts and returns the status(available or unavailable in stock)of the parts.
e) OrderIdWS - This service returns the ordered which helps in transferring the control to the new branch.
f) RepairINWS - This service generates the rate for each of the parts suggested by the mechanic.

g) PrepareInvoiceWS - This web service obtain the payment data from the mechanic and list from the Repair Service. Based on the list and needed parts obtain from the mechanic, a bill is generated and calculated as a payable amount by the customer.

h) VehicleAdminWS - This sends notification regarding the retrieval of the Net Pay by the Service Station.

Web Service Composition

BPEL is used in composing the web service and deployed in Apache Orchestration Directory Engine. In this project, composition is done namely E-RepairService and E-VehicleService. The services used in the composition of E-RepairService are,

- StockVerificationWS
- OrderIDWS
- SurveyParts
- RepairInWS

The Service involved in E-VehicleService are

- NeededpartsWS
- RepairInWS
- SurveyParts
- PrepareInvoiceWS
- E-RepairService
- VehiclesAdminWS

Deployment Of Aspect

Apache ODE is considered for weaving the aspect into BPEL. In order to provide runtime process change, the security functionality is injected in the result of the OrderWS. It can be (un)plugged at any point of time without modifying existing BPEL process.

IV. SCOPE OF ASPECT SERVICE IN E-VEHICLE SERVICE SYSTEM

For the security requirements of web service composition, we consider e-vehicle service as an example scenario. The service providers define a BPEL process for the purpose of composing set of basic services. The process orders items from the supplier and consequently invokes the bank's payment web service to pay the transaction amount. The operations of the partner web services require authentication since it is not acceptable that anyone who knows the URL of the service or finds it in a UDDI registry can place orders or perform bank transfers. The supplier and bank web services must be accessible only to business partners with appropriate credentials. This means that the web service composer has to know the security policy of the partner service before writing the BPEL process that invokes the partner service.

With authentication mechanisms, the partner web service can be sure of the identity of the caller. The next step is to decide what the caller is allowed to do, which is the focus of authorization. Furthermore, it is also important that the factory which passed the order to the supplier cannot deny having done so (non-repudiation) and that nobody can claim the misuse of identity. Digital signatures and signature verification can be used for implementing the security. A further requirement is data integrity; both parties need appropriate support for integrity, i.e., if the factory orders one hundred items then the security infrastructure must make sure that nobody can tamper with the data on its way to the target web service and change the order position. Appropriate security mechanisms [3] are also needed to avoid replay attacks, i.e., if some malicious third-party copies the message for ordering car parts from the registry and resends it later, then the order should not be accepted for the second time. This is usually implemented by means of timestamps. When invoking the bank's payment web service, it is essential that nobody can see the sensitive information transferred from the composite service to that partner (confidentiality). Both parties have to negotiate and agree on the mechanisms (usually key-based encryption and decryption) used to ensure confidentiality.

The security requirements of BPEL process focused on the<invoke> activity during interaction of partner web service. The <receive> and <reply> activities have slightly different requirements on security specifications.

A BPEL orchestration engine that executes the <receive> activity waits until a client request matching that <receive> arrives and then starts the process interpretation. From the client's perspective, the composite web service looks like any other web service with its WSDL interface. From the security perspective, it should be clear who is allowed to invoke the composite web service, what kind of authentication, encryption, or signing mechanisms are required by the composite service. The security policy of the composite web service can be expressed using the WS-Policy specification; this policy will have to be enforced by the orchestration engine. Unfortunately, current BPEL engines do not yet support policies. For the <reply> activity, the orchestration engine on which a composite web service runs has to secure the response as required by the security policies of both the composite web service and its client. The security framework should make sure that the response is encrypted if the client requires an encrypted response and that the orchestration engine supports one of the encryption algorithms specified in the client policy.

The issues mentioned so far are not covered by BPEL. There is no means to express the security capabilities and requirements of the process or of a given activity. However, this is not a limitation of BPEL itself because non-functional concerns should be addressed by other specifications for a better separation of concerns and for more modular composition specification. If we extend BPEL with new constructs for each non-functional concern of the composition, it would evolve into a very complex language, which in turn would limit it acceptance. Furthermore, mixing the specification of the core logic of the composition with specifications of security features and other non-functional concerns into one unit would make the composition specification too complex and hard to maintain and evolve.

In this paper, we present a framework for securing web service compositions based on AOP. The logic needed to ensure security features is plugged into the composition logic using a set of aspects. These aspects are generated from a generic aspect library at deployment time according to the deployment descriptor, which specifies the security requirements of BPEL activities along with the required security parameters such as keys and certificates. A web service is used by the orchestration engine to add security to certain <invoke>, <reply>, and <receive> activities. The integration of the BPEL process and the security service is tackled by the generated aspects.

Composition with BPEL

BPEL is a work-flow based web service composition language. It specifies the composition as a process, which declares the web services participating in the composition (partners), data containers (variables), and a set of activities with specific patterns of control and data flow. The building blocks of BPEL processes are activities. There are primitive activities such as <invoke> and <assign> and structured activities such as <sequence> and <flow>. Structured activities manage the order of execution of their enclosed activities. BPEL processes can run on any BPEL-compliant orchestration engine. The engine orchestrates the invocations of the partner web services according to the process specification. For illustration, we present a skeleton of the BPEL process that corresponds to the car manufacturer scenario through the code given below. We omitted some unessential constructs due to restriction of space in the page.

```
<process name="OrderProcess"/>
<partnerlinks>
<partnerLink name=supplier".../>
<partnerLink name="bank".../>
<partnerLink name="factory".../>
</partnerlinks>
<variables>
<variable name="clientrqst"
messageType="orderInMT"/>
<variable name="clientrspse"messageType=
"orderOutMT"/>
<variable name="payrequest"
messageType="payInMT"/>
----
</variables>
<sequence name="Main">
<receive partnerlink="factory"
operation="order"
variable="orderqst"
createInstance="yes" --/>
<invoke partnerlink=supplier"
operation="putOrder"
inputvariable="supplyrequest" --/>
----
<invoke partnerlink="bank"
operation="pay"
inputvariable="payrequest" --/>
----
<reply partnerlink="factory"
operation="order"
variable="clientrspse" --/>
</sequence>
</process>
```

WS-Security and WS-Policy

Signcryption[5] is a new paradigm in public key cryptography that simultaneously fulfills both the functions of digital signature and public key encryption in a logically single step, and with a cost significantly lower than that required by the traditional signature and encryption approach.it uses two Schemes Digital Signature Public Key encryption it uses RSA cryptosystem where the Signcryption costs on average 50 percent less in computation time. 91 percent less in message expansion. This Signcryption Algorithm [5] can be implemented using: ElGamals Shortened Digital Signature Scheme.

Schnorr Signature Scheme ensures that the message sent must not be forged and the contents of the message are confidential and non-repudiation Computation involved when applying the Signcryption, Unsigncryption algorithms and communication overhead is much smaller than signature-thenencryption schemes Receiver can only obtain the message m by decrypting it using his private key Xb Any changes he makes to the message m will reflect in the next step of Signcryption one-way keyed hash function on the message m will not match the value r. An attacker has all three components of the Signcrypted message: c, r, s. He still cannot get any partial information of the message m. The attacker have to also know receiver's private key, p and q (known only to Sender and Receiver).

Implementation of AO4BPEL

The pointcut language of AO4BPEL is XPath. That is, XPath expressions are used to select the activities where the advice code should be executed. Point-cuts can span several processes. An advice in AO4BPEL is a BPEL activity that specifies some crosscutting behavior that should execute at certain join points. Like AspectJ [10], we support before, after and around advice. That is, the behavior defined in an advice can be executed before, after or instead a join point activity. The around advice allows replacing an activity by another. The activity of integrating aspects into base functionality is called weaving. A weaver is a tool that integrates a base program's execution with aspects. In the case of AO4BPEL, the base program is the BPEL process. AO4BPEL supports dynamic weaving, i.e., aspects can be deployed or un-deployed at process interpretation time. We have implemented AO4BPEL as an aspect-aware orchestration engine for BPEL.

Cross-Layer Pointcuts

To implement the security framework, it was necessary to extend AO4BPEL with pointcut designators that in addition to capturing BPEL activities such as <invoke>, <reply>, also capture so-called internal join points. The later are points in the interpretation of an activity, i.e., points at the interpretation level rather than at the composition specification level. To implement security logic we need to capture execution points during the orchestration engine's interpretation of a BPEL activity, where SOAP messages are sent/received. We need to modify a SOAP message before it is sent out in the course of interpreting certain BPEL activities invoke the security service.

V. Security Framework

In this section, we present the design and the implementation of our security framework.

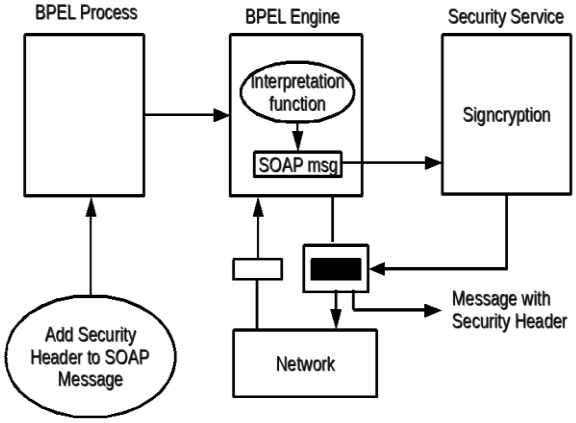

Fig. 5. The Security Framework

The architecture shown in Figure 5 depicts the overall structure of the proposed security framework. The core components of our security framework are the deployment descriptor, the security service and the process container.

The XML-based deployment descriptor relates process activities to its security requirements and the necessary parameters. For example, the deployment descriptor could declare that any <invoke> activity that calls the operation pay of the payment web service requires caller authentication. The security framework deployment descriptor invokes the security header of the soap message in the BPEL process. Then the deployment information is send to BPEL engine through the interpretation function. This function generates a SOAP [11] message and sends the results to the security service. The SOAP message with the security header is send through the network and performs the action and given back to the BPEL process.

VI. Conclusions And Future Work

In this paper, we presented a framework for securing web service compositions. The core components of this framework are the aspect-based process container, the security service, and the deployment descriptor. The process container intercepts some points in the process interpretation and plugs in to make BPEL interactions through messaging activities safer. The container is implemented as a set of aspects specified in the AO4BPEL. The deployment descriptor enables an easier usage of the framework and hides the implementation details from other users. The remainder of this project work involves, bringing security functionality as an aspect and encapsulated it as a separate entity.

References

[1] Y. Vasiliev, SOA and WS-BPEL, ser. From technologies to solutions. Packt, 2007. [Online]. Available: http://books.google.co.in/books?id=quRbYxaQ8wcC.

[2] Charfi and M. Mezini, "Ao4bpel: An aspectoriented extension to bpel," World Wide Web, vol. 10, no. 3, pp. 309–344, Sep. 2007. [Online]. Available: http://dx.doi.org/10.1007/s11280-006-016-3.

[3] Mira and C. Mezini, "Using aspects for security engineering of web service compositions," in Proceedings of the IEEE International Conference on Web Services, ser. ICWS '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 59–66. [Online]. Available: http://dx.doi.org/10.1109/ICWS.2005.126

[4] R. Schmelzer, XML and Web services unleashed, ser. Unleashed Series. Sams, 2002. [Online]. Available: http://books.google.co.in/books?id=XuxSAAAAMAAJ.

[5] Y. Zheng, "Signcryption and its applications in efficient public key solutions," in In Proceedings of ISW 97, volume 1396 of LNCS. Springer-Verlag, 1997, pp. 291– 312.

[6] "Apache software foundation. apache ode (orchestration director engine)." http://ode.apache.org.

[7] SamJaffray and S. SenthilVelan, "Measuring the dynamic of aspect oriented business process execution language in soa," in International Conference on Innovative Computing and Information Processing, ser. ICICIP '12, March 29-31 2012.

[8] Houspanossian and M. Cilia, "Extending an open source bpel engine with aspect oriented programming," in Proceedings of the Argentina Symposium on Software Engineering, ser. ASSE '05, 2005.

[9] Charfi, Aspect-oriented Workflow Languages: AO4BPEL and Applications, 2007. [Online]. Available: http://books.google.co.in/books?id=YV2mNwAACAAJ.

[10] R. Laddad, AspectJ in Action: Enterprise AOP With Spring, ser. Manning Pubs Co Series. Manning, 2009. [Online]. Available: http://books.google.co.in/books? id=ZmniOgAACAAJ.

[11] "W3c. simple object access protocol (soap) 1.1." http://www.w3.org/TR/soap/, May 2000.