Decoupling Security Concerns in Web Services Using Aspects

G. Kouadri Mostéfaoui
Fribourg University, Switzerland
ghita.kouadrimostefaoui@unifr.ch

Z. Maamar
Zayed University, U.A.E
zakaria.maamar@zu.ac.ae

N. C. Narendra
IBM India Research Lab, India
narendra@in.ibm.com

S. Sattanathan
National Institute of Technology Karnataka, India
ss_nitk@yahoo.co.in

Abstract

This paper discusses the Aspect-oriented Framework for Web services (AoF4WS) that supports on-demand context-sensitive security in Web services. Flexible security schemes are needed in many Web services applications where authentication, authorization, etc., can no longer be used in their current form. Security mechanisms are to be customized to the continuously changing requirements of Web services. Examples of this customization concern cryptographic protocol for a specific situation and timeout for user credentials. The AoF4WS uses aspect-oriented programming and frames. Aspects provide flexibility to the framework, and frames adjust aspects to specific requirements.

Keywords. *Security, Aspect-oriented programming, Framed aspect, Web services.*

1. Introduction

Web services are an attractive approach for implementing business processes, which usually spread over companies' boundaries [1]. Over the last few years several efforts have been put in the development of standards related to Web services definition, discovery, triggering, composition, etc. Another element, which will without a doubt boost the acceptance rate of Web services by the IT community, is security. Like WS-Security and WS-Trust initiatives [2,7], other initiatives for Web services security are geared towards the low-level requirements of achieving on the one hand confidentiality, integrity, and non-repudiation of messages, and on the other hand authentication of users. This is by far not enough! Effective security strategies for Web services, should not only revolve around messages and users, but also focus on the capacity of adapting to continuous changes in the business environment [17]. The emergence of new standards and identification of new threats require a different way of engineering security.

Adaptability concerns any software system that operates within a changing environment. Security is an excellent trigger for reviewing and adapting strategies. Usually Web services are set according to a certain level of risk, which is bearable to owners (i.e., zero-risk situation does not exist), does not affect their integrity, and does not diminish the trustworthiness level of potential users towards them. However there are situations where this level of risk is no longer acceptable worrying providers and backing users off as well. This requires reviewing the security strategy of Web services, which is extensive, expensive, and error-prone. Inter-related mechanisms such as authentication and encryption need to be checked, and the review of one mechanism affects others due to code crosscutting. It is normal to face situations where modularizing some concerns (e.g., logging, error handling) using current software engineering techniques is hardly to achieve and sometimes impractical. As a result, code gets scattered all over the system and becomes difficult to localize and maintain.

To address the challenges of adapting the security strategy of a Web service and to promote a clear separation between "business" and "management" sides of a Web service (Figure 1), we adopt an Aspect-Oriented Programming (AOP) approach to develop this security strategy [3,11]. This research is part of our **A**spect-**o**riented **F**ramework for **W**eb **S**ervices (**AoF4WS**) project. It aims at looking into the role of aspects in decoupling various concerns in Web services like security. "Business" and "management" separation emphasizes the non-invasive requirement that needs to be considered during the whole development cycle of a security strategy. The mechanisms related for instance to security are confined into one module and do not scatter over the rest of modules of the Web service. Figure 1 illustrates the way concern separation occurs in a fictive Web service referred to as HotelBooking. The business side focuses on details directly related to hotel booking like checking room availability, rate verification, and confirming client reservation. The management side of a Web service gathers all modules like security, maintenance, and performance that back the operations of this Web service and permit boosting its acceptance rate. Constituents of the management side to be implemented as aspects need to be factored out of the core logic of the Web service. Ortiz et al. define aspects as units of

Proceedings of the Third International Conference on Information Technology: New Generations (ITNG'06)
0-7695-2497-4/06 $20.00 © 2006 IEEE

encapsulation that are built upon two elements [14]: join points and advices. Join points determine the places or pointcuts where the behavior alteration of an application will happen. Advices identify the new code to be injected in response to this alteration.

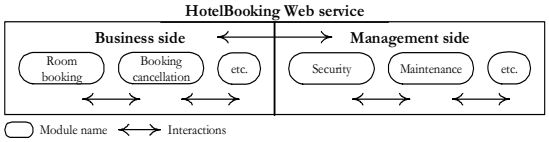

Figure 1 Concern separation in a Web service

The aspect-oriented approach we propose does not devise new security countermeasures against threats on Web services. It constitutes, however, a structured way to provide a generic architecture upon which security mechanisms are deployed in a plug and play manner. Complying with this approach, reconfiguring security mechanisms of a Web service is triggered upon assessing the risk level. The compliance with an aspect-oriented approach promotes modularity, reusability, and flexibility of a Web service. Modularity is the capacity of a Web service to separate between functional and non-functional modules of a Web service. Reusability is the capacity of a Web service to take advantage of the existing modules without altering its core business or considering code duplication. And, flexibility is the capacity of a Web service to select the modules that accommodate the progress of an ongoing situation.

Although the focus in this paper is on the value-added of aspects to Web services from an adaptive security perspective, extra perspectives like monitoring and maintenance could benefit from using aspects. For instance, monitoring the participation of a Web service in multiple compositions does not contribute to the core business logic of this Web service. Thus it would be appropriate to keep monitoring independent from the business logic of the Web service. The rest of this paper is organized as follows. Section 2 presents some initiatives on Web services security and some works on AOP and Web services. Section 3 discusses the architecture and operation of the AoF4WS. The paper concludes in Section 4.

2. Background

2.1. Initiatives on Web services security

The open and dynamic nature of the environment in which Web services operate poses various challenges to their security. New Web services appear while others disappear without prior notice. Furthermore, messages among component Web services of a composite service have to be checked for integrity, confidentiality, and authentication purposes. The need to secure Web services is strengthened in [7], as the use of Web services continues to increase. This increase is dependent on how much Web services are a serious development alternative to other rival middleware like CORBA and RMI. For instance, some still consider Web services as distributed objects that react upon request only [8]. Enhancing Web services with extra capabilities can happen along three perspectives as reported in [9]. The first perspective is about deploying Web services that assess the environment before they take part in a composition. The second perspective is about reducing the semantic heterogeneity gap between independent Web services that have all agreed to participate in a composition. Finally, the third perspective is about conciliating contextual information of Web services using ontologies.

WS-Security is an emerging standard dedicated to securing messages among interacting Web services [2]. To this purpose, WS-Security defines how security tokens are contained in SOAP messages. WS-Security is extensible to other security models like Secure Sockets Layer (SSL), Kerberos, and Public Key Infrastructure (PKI).

Transport Layer Security (TLS) secures interactions by using encryption and makes servers and clients collaborate in order to decide on the authentication process to adopt during data transfer. Unfortunately, TLS does not scale well to complex transactions like those involving Web services [4]. Traditional security techniques such as Virtual Private Network (VPN) and SSL cannot secure the large number of transactions that Web services expect receiving.

The W3C's Web services architecture adopts PKI to secure communications over public networks [5]. But, PKI is complex, which negatively affects its deployment cost, processing time, etc. Moreover, PKI has the reputation of being quite cumbersome. This could prove to overkill the Web services security to be engaged in intense interactions [6].

The eXtensible Access Control Markup Language (XACML) is an OASIS standard [23], which describes both a policy language and an access control decision service interface. A policy is extensible and describes general access control requirements. The request/response style for setting access controls allows forming a query to ask whether or not a given action is allowed: permit, deny, indeterminate, or not applicable.

Proceedings of the Third International Conference on Information Technology: New Generations (ITNG'06)
0-7695-2497-4/06 $20.00 © 2006 IEEE

2.2. AOP for Web services

Cibrán and Verheecke promoted modularizing Web services management with AOP [10]. That was motivated due to the hard-wiring technique that is used for integrating Web services into applications. This hard-wiring way has several deficiencies when it comes to working out how to adapt to changes, what if a service fails, and how to deal with issues related to peers like checking for availability, switching to other services, etc.

In [12], Charfi and Mezini apply AOP to Business Process Execution Language for Web Services (BPEL4WS) in order to achieve modular and dynamic adaptability of Web services composition. This is done using AO4BPEL that extends BPEL with features that permit for instance viewing business rules as aspects.

In [13], Ortiz et al. adopted an aspect-oriented approach to develop solutions for Web services composition (of type orchestration) and interaction patterns. They also raised multiple questions related for instance to the possibility of reusing interaction patterns previously implemented, and the efforts to put in for modularizing these patterns rather than scattering the code. During the experiments that were conducted, Ortiz et al. noted that modularity, reusability, and maintenance that should feature applications are not properly handled. Therefore they looked into ways of achieving these features using AOP.

In another work [14], Ortiz et al. decouple non-functional properties in Web services by adopting aspects. It was noted that adding such properties to Web services results in changes in the various modules of an application. Logging, timing, security, and authentication are examples of non-functional properties. Ortiz et al. used the timing non-functional property as an example to show how the code implementing this property is repeated and scattered over multiple modules and how much this code is difficult to maintain.

3. Presentation of the AoF4WS

3.1. Architecture

Figure 2 presents the way aspects are part of the AoF4WS. Three levels of abstraction are shown in this figure: composite, component, and resource. The constituents of each level are associated with a particular type of context denoted by C-context, W-context, and R-context. The rationale and role of each context type are given in [16]. The connection between composite, component, and resource levels is implemented with "participate in", "oversee", and "operate upon" relationships, respectively. Some features of the AoF4WS are as follows: multi-level concern separation using aspects, and contextual tracking of the security requirements of Web services.

The composite level is about specifications of context-aware composite services. A specification is split into two parts: business logic and aspects. The business-logic part reflects the overall objective that the composite Web service has to reach (e.g., hotel booking) using a set of component Web services. The aspect part reflects the cross-cutting concerns that are included in the operation of the composite Web service, and which are orthogonal to this overall objective. The business logic specifies the process by which user functional requirements are met, whereas aspects model user non-functional requirements like security, reliability, and performance with emphasis on security in this paper.

The component level is about context-aware Web services. Similar considerations apply to Web services, which are split into two parts: business logic and aspects. The business-logic part shows the actions a component Web service individually or collectively carries out in order to enable reaching the composite Web service's overall objective. The aspect part shows the different non-functional requirements that manifest themselves as cross-cutting concerns affecting the actions and interactions of the Web service.

The resource level is about context-aware resources. Resources represent the computing means on which Web services operate. The scheduling of execution requests of Web services is prioritized when enough resources are not available to satisfy them all at once. Moreover, resource allocation to Web services is subject to the context in which the Web services evolve. For instance, the computing requirements of a Web service need to be checked against the computing capabilities of the resources prior to performing resource allocation.

In Figure 2, the three-level representation of AoF4WS results in categorizing aspects into coarse grained and fine grained. A coarse-grained aspect is any non-functional property that concerns a Web service like security. A fine-grained aspect is any concern like Kerberos-based authorization that implements the security non-functional property of this Web service. In the rest of this section, we present our mechanisms for weaving security aspects into Web services. This is summarized using the following steps:

Proceedings of the Third International Conference on Information Technology: New Generations (ITNG'06)
0-7695-2497-4/06 $20.00 © 2006 IEEE

1. Deriving a security context of the component Web services using W/C/R-contexts.
2. Defining security policies based on security context and storing these policies in a repository.

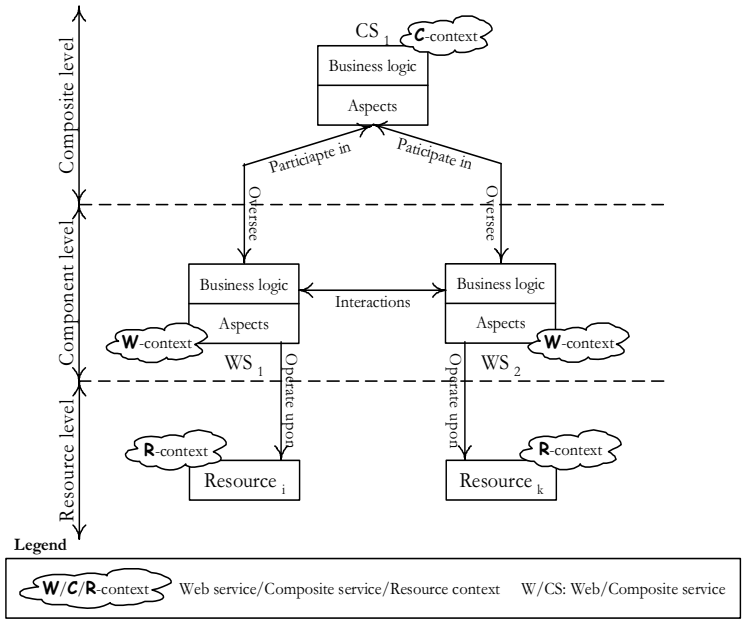

Figure 2 Architecture of the AoF4WS

3. Generating the aspects from the security policies and selecting the appropriate ones using security context.
4. Weaving the selected aspects into the component Web services.

3.2. Configuration of security aspects

The development of the AoF4WS happened along two dimensions. The first dimension is the need for an adaptive approach that triggers upon request security services to be implemented as aspects. For instance, in a specific situation, only authentication aspect needs to be activated, while an extra-logging aspect is activated in another situation. We refer to this dimension in the AoF4WS as composite configuration since it only targets composite Web services. The second dimension highlights the need for a fine tuning of each security aspect associated with composite configuration. For instance, the authentication aspect can be set to accept a timeout of 10 seconds when requesting clients' credentials. We refer to this dimension in the AoF4WS as component configuration since it only targets Web services.

The identification of a configuration that includes both composite and component levels calls for an additional technology to support aspect-oriented programming in modularizing crosscutting concerns at each level. This technology corresponds to frames. These latter are defined as wrappers around code snippets (e.g., source code, HTML code) [20]. A frame contains variation points that permit adding, deleting, or adapting a functionality in a specific application. This happens using various commands like overriding, extension, substitution, and iteration.

Composite configuration of security aspects. In Figure 3, the operation of the AoF4WS in a composite configuration is illustrated. This operation consists of selecting the security aspects that protect the whole Web services environment (these aspects are referred to as active in Figure 3). The selection process combines contextual information and policies. Contextual information offer details on the environment that surrounds each element (Web service, composite Web service, resource), and policies suggest the appropriate security aspects based on these details.

In addition to W/C/R-contexts of Web services, composite Web services, and resources in the AoF4WS, a new type of context that is just dedicated to security is required (Figure 3). S-context is fed with various details from W/C/R-contexts and gets involved in triggering policies for weaving active security aspects. For Kouadri Mostéfaoui, *a security context is a state of the working environment that requires taking one or more security actions. A security context is formed*

by a set of information collected from the user's environment and the application environment and that is relevant to the security infrastructure of both the user and the application [20]. In Figure 3 the feeding process is an event-trigger system that gathers contextual information from appropriate sources like contexts of Web services and contexts of resources. Additional sources could be used for feeding the security context like physical sensors in the environment or user inputs [21]. Based on the content validity of a context, policies are triggered. In the following and relying on a previous work in [22], we overview some arguments that populate each type of context and illustrate the specification of a policy in Ponder. For more details on Ponder and why Ponder is selected, readers are referred to [15].

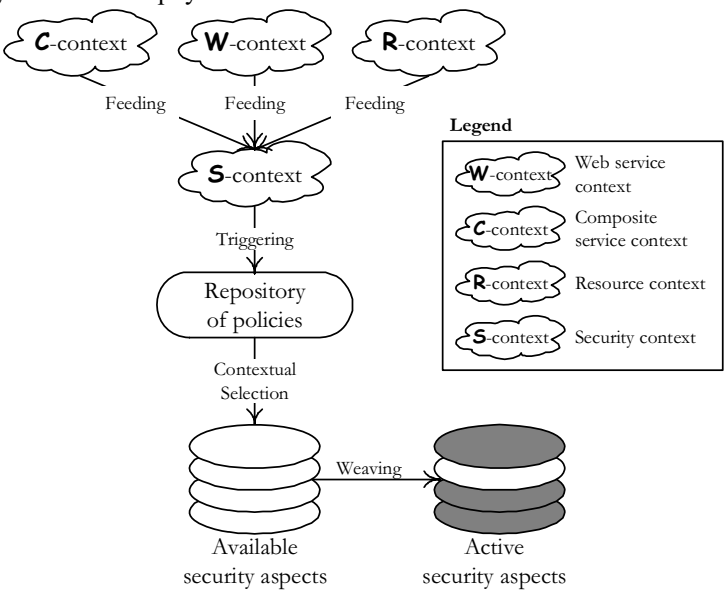

Figure 3 Operation of the AoF4WS

Some arguments of W-context are: signature (establishes the identity of the Web service so messages to peers are known), security mechanism (sets the encryption/decryption mechanism needed for authenticating messages received from peers), security status (indicates the status of authenticating the received message in terms of success or failure), and violation (indicates the type of security violation that a message was subject to). Arguments of C-context are similar to arguments of W-context but are interpreted at the composition level. Some arguments of R-context are: signature (establishes the identity of the Web service that operates on top of the resource), and violation (indicates the type of security violation that the Web service is involved in). Finally some arguments of S-context are: signature per Web service/Composite Web service/Resource, security mechanism per Web service/Composite Web service/Resource, security status per Web service/Composite Web service/Resource, and security violation per Web service/Composite Web service/Resource. The main role of S-context is to report on which authentication mechanisms (username/password pairs, binary certificate, etc.), certificate algorithms, etc. are supported by all components whether Web service, composite Web service, or resource, and when they are active.

Policies are *information which can be used to modify the behavior of a system* [15]. The use of policies in AoF4WS permits managing Web services at a higher level where guidelines for conducting composition of Web services are separated from guidelines for securing Web services. The following is a policy in Ponder that authorizes activating a certain security aspect following the invocation request that Taxi Web service receives from Travel Web service. This security aspect depends on the types of authentication and encryption mechanisms featuring Travel Web service. In this policy, details about these mechanisms are available in the S-context of Travel Web service.

```
inst oblig AuthorizeTaxiWebService{
    on ServiceRequest(s,t);
    when S-context.authentication_algorithm(s,"Kerberos",1) and
    S-context.encryption_algorithm(s,"DES",1)
    subject s = /travel-web-service;
    target t= /taxi-service;
    action t.activate(aspect_i);
```

Component configuration of security aspects. In Figure 4, the operation of the AoF4WS in a component configuration is illustrated. This configuration supports the customization of each active-security aspect that was identified in composite configuration and according to the requirements that a user sets. This is achieved using frames. Some examples of user requirements are authentication to happen within 10 seconds and AES-128 is the encryption algorithm. In Figure 4, we also present the way a customized aspect is defined (adapted from [20]):

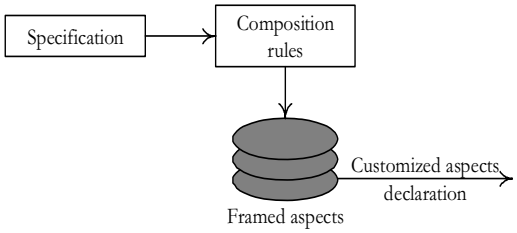

Figure 4 Framed security aspects generation

- Specification is about the developer's security specification to be customized. This consists of setting the values of the meta-variables and selecting the different options available in the framed aspect. For example, a specification sets the concrete value of a timeout variable that is contained in a framed authentication aspect code.
- Composition rules control the way aspects are bound together. For example, an aspect's parameter (e.g., timeout) already defined in the specification can be constrained to a specific interval. The weaving process will then occur upon these constraints.
- Framed aspect is a parameterized version of the original security aspect that was established in the configuration of type composite. In addition to the generalized aspect code that a famed aspect contains, more elements are added like conditional compilation and parameterization.

The reader may wonder the relationship between policies defined earlier and composition rules. Policies are responsible for selecting the security aspects to be activated according to the current context of the whole Web services environment. A composition rule defines how the selected security aspects will be woven in order to secure specific Web services. This rule is seen as a policy at the micro level of the aspect. The composition rules apply to the set of aspects once these aspects are selected and customized following their respective specifications.

3.3. Putting it all together

In previous contributions [18,19], weaving of aspects - for generic applications - is based on a simple schema, i.e., on the adaptation of the aspects at the composite level (Section 3.2 for more details on composite configuration of security aspects). The AoF4WS - more specific to Web services - adds an extra step that consists of running an adaptation at the component level by integrating a set of relevant contextual information. Compared to Figure 4, Figure 5 illustrates the operation of the AoF4WS after combining composite and component configuration. The new elements in the AoF4WS are as follows:

- Web services environment includes the different component Web services of a composite service.
- Security monitor provides the right set of configured security aspects to the Web services environment.
- Nanning runtime is based on the Nanning runtime tool (nanning.codehaus.org/) for runtime weaving of security aspects. AspectWerkz, JAC, JAsCo, AOPAlliance, and Prose are examples of other weaving tools.

Figure 5 shows the overall picture of the AoF4WS in operation. A transaction in the Web services environment (e.g., a request to use a Web service's functionality) requires from the security monitor to set the needed security aspects (i.e., a request is automatically forwarded to the security monitor before being fulfilled). Component and composite configurations of the AoF4WS engage in collaboration to fulfill this transaction. In the first step - composite configuration - a list of security aspects (e.g., authentication, logging) that need to be included in the security framework is produced. The selected security aspects are then framed in the second step - component configuration -. This step is how each single aspect will be customized in order to properly respond to the context of use (e.g., type of protocol used by the privacy service). The final set of framed aspects is then concretely woven using Nanning runtime and applied in order to secure the transactions in the Web services environment.

3.4. Illustrative Scenario

In this section we present an example that illustrates our ideas. Let us assume a **T**ravel-**A**gent **C**omposite **S**ervice (TA-CS_1) that puts together an

itinerary for a tourist. Two of the most significant component Web services of TA-CS_1 are **T**axi **B**ooking (TB-WS_1) and **H**otel **B**ooking (HB-WS_2). In the C-context of TA-CS_1, Blowfish algorithm is set as part of the security mechanism. In the W-contexts of TB-WS_1 and HB-WS_2, DES and AES algorithms are set, respectively. A resource that HB-WS_2 uses is an online database through which information on the famous places to stay in are provided. In the R-context of the database, authentication information is presented so HB-WS_2 gets to access this database.

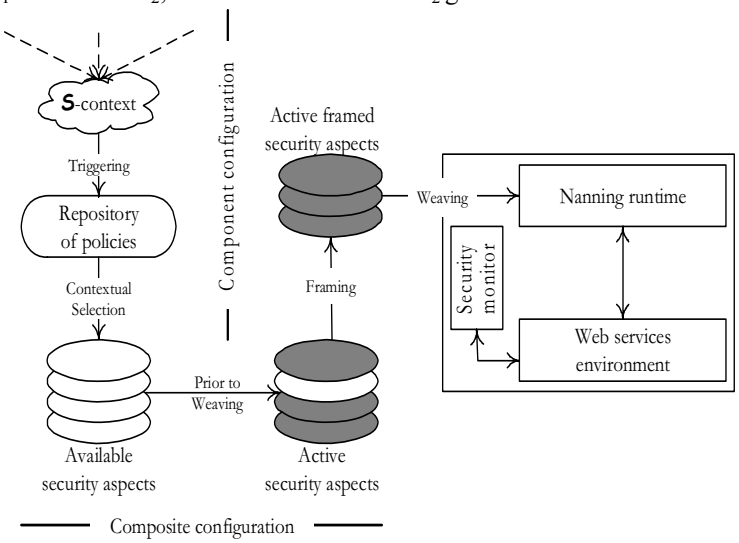

Figure 5 Composite and component configuration of security aspects

Based on the above W/C/R-contexts, the S-context arguments of TA-CS_1 are instantiated. Some of them are: DES algorithm for HB-WS_1, Blowfish algorithm for TA-CS_1, security status for TB-WS_2 accessing the database resource ("access granted"), and security violation (if any has occurred – in our case, so far, no). The S-context information is then used to populate the policy repository with the appropriate policies (Figure 5). A sample of policy is to authorize the invocation of TB-WS_1 upon request of TA-CS_1, assuming that the security conditions are met.

```
inst oblig AuthorizeTaxiWebService{
    on ServiceRequest(s,t);
    when S-context.authentication_algorithm(s,"Blowfish",1) and
    S-context.encryption_algorithm(t,"DES",1)
    subject s = /TA_CS1;
    target t= /TB_WS1;
    action s.invoke(t);
```

In other words, when Travel Web service authenticates itself to Taxi service, this latter is supposed to accept the invocation request from Travel-Agent service. A similar hotel room booking request can also happen, as shown below.

```
inst oblig AuthorizeHotelBookingWebService{
    on ServiceRequest(s,t);
    when S-context.authentication_algorithm(s,"Blowfish",1)
    and S-context.encryption_algorithm(t,"AES",1)
    subject s = /TA_CS1;
    target t= /HB_WS2;
    action s.invoke(t);
```

Based on the policies defined above, the list of appropriate security aspects is generated (i.e., DES aspect, Blowfish aspect, and AES aspect). Since the above policies are needed for our tourist, the appropriate authentication code is weaved into the respective Web services (TB-WS_1 and HB-WS_1), in order to ensure that the necessary security checks are carried out during the composition of these Web services. The actual weaving is itself carried out via frame technology, as follows. Prior to the weaving process, each framed security aspect -identified by the list generated earlier- is customized according to the values set in the specification as illustrated in Figure 4 (component configuration of security aspects). Afterwards the framed versions of these framed security aspects are woven using the Nanning runtime.

4. Conclusion

In this paper, we discussed the need for adaptive security in Web services environments. To this end, we proposed the Aspect-oriented Framework for Web Services for adaptive security using framed aspects, which combine frames and aspect-oriented programming. Frames enhance aspect-oriented programming by separating the specification of a security aspect from the aspect code itself. This approach allows a fine-grained variation of security aspects according to the context of use. The Aspect-oriented Framework

for Web Services relies on context-based policies expressed using Ponder. Our future work consists of the following: investigate ways to determine the execution order of security aspects; extend our framework for decoupling other concerns such as performance, reliability, etc.; and extend our framework for decoupling business regulations [24] into different aspects, so that changes to one set of regulations do not affect the others.

Acknowledgements
The authors thank Chris Giblin for his feedback. The fourth author thanks his Ph.D. advisor Prof. K. C. Shet, for supporting his work. Other company (i.e., non-IBM), product and service names may be trademarks or service marks of others.

References

[1] Kevin J. Ma. "Web Services: What's Real and What's Not?" IT Professional, 7(2), March/April 2005.

[2] Web Services Security. Version 1.0, April 2002. http://www.verisign.com/wss/wss.pdf. (Visited August 2005).

[3] Y. EL-Manzalawy. "Aspect Oriented Programming" http://www.developer.com/design/article.php/3308941 (Visited August 2005).

[4] Y. Nakamur, S. Hada, and R. Neyma. Towards the Integration of Web Services Security on Enterprise Environments. In *Proceedings of The Workshop on Web Services Engineering (WebSE'2002) held in conjunction with The IEEE/IPSJ Symposium on Applications and the Internet (SAINT'2002)*, Nara, Japan, 2002.

[5] W3C Working Group, http://www.w3.org/ (Visited August 2005)

[6] R. Sandhu. Good-Enough Security: Toward a Pragmatic Business-Driven Discipline. *IEEE Internet Computing*, 7(1), January/February 2003.

[7] K. R. Moorthy and A. Gandhirajan. The Foundations of Web Services Security. http://www.developer.com/services/article.php/3496326, (Visited August 2005).

[8] K. P. Birman. Like It or Not, Web Services Are Distributed Objects. *Communications of the ACM*, 47(12), December 2004.

[9] Z. Maamar, D. Benslimane, and N. C. Narendra. What Can Context do for Web Services? *Communications of the ACM*, 2006 (forthcoming).

[10] M. A. Cibrán and B. Verheecke. Modularizing Web Services Management with AOP. In *Proceedings of The 1st European Workshop on Object Orientation and Web Services (ECOOP'2003) held in conjunction with The 17th European Conference on Object-Oriented Programming (ECOOP'2003)*, Darmstadt, Germany, 2003.

[11] T. Cottenier and T. Elrad. Validation of Aspect-Oriented Adaptations to Components. In *Proceedings of The 9th International Workshop on Component-Oriented Programming (WCOP'2004) held in conjunction with The 18th European Conference on Object-Oriented Programming (ECOOP'2004)*, Oslo, Norway, 2004.

[12] A. Charfi and M. Mezini. Hybrid Web Service Composition: Business Processes meets Business Rules. In *Proceedings of The 2nd International Conference on Service Oriented Computing (ICSOC'2004)*, New-York, USA, 2004.

[13] G. Ortiz, J. Hernández, and P. J. Clemente. Web Services Orchestration and Interaction Patterns: An Aspect-Oriented Approach. In *Proceedings of The 2nd International Conference on Service Oriented Computing (ICSOC'2004)*, New-York, USA, 2004.

[14] G. Ortiz, J. Hernández, and P. J. Clemente. Decoupling Non-Functional Properties in Web Services: An Aspect-Oriented Approach. In *Proceedings of The 2nd European Workshop on Web Services and Object Orientation (EOOWS'2004) held in conjunction with the 18th European Conference on Object-Oriented Programming (ECOOP'2004)*, Norway, June 2004.

[15] N. Damianou, N. Dulay, E. Lupu, and M Sloman. The Ponder Specification Language. In *Proceedings of the Workshop on Policies for Distributed Systems and Networks (Policy'2001),* Bristol, UK, 2001.

[16] Z. Maamar, S. Kouadri Mostéfaoui, and Q. H. Mahmoud. On Personalizing Web Services Using Context. *International Journal of E-Business Research, Special Issue on E-Services*, The Idea Group Inc., 1(3), July-September 2005.

[17] J. Hillman and I. Warren. An Open Framework for Dynamic Adaptation. In *Proceedings of The International Conference on Software Engineering (ICSE'2004),* Edinburgh, Scotland, 2004.

[18] N. Loughran and A. Rashid. Supporting Evolution in Software using Frame Technology and Aspect Orientation. In *Proceedings of The Workshop on Software Variability Management,* Groningen, The Netherlands, 2003.

[19] P. Greenwood and L. Blair. Dynamic Framed Aspects for Policy-Driven Auto-Adaptive Systems. http://www.comp.lancs.ac.uk/computing/aose/papers/dynFr_daw04.pdf

[20] G. Kouadri Mostéfaoui. Towards a Conceptual and Software Framework for Integrating Context-Based Security in Pervasive Environments. *Ph.D. Thesis No. 1463*, University of Fribourg and Paris 6 University, 2004.

[21] A. Schmidt, M. Beigl, and H. W. Gellersen. There is More to Context than Location. *Computers & Graphics Journal*, 23(6), December 1999.

[22] S. Sattanathan, N. C. Narendra and Z. Maamar. Towards Context-based Tracking of Web Services Security. In *Proceedings of The 7th International Conference on Information Integration and Web Based Applications & Services (iiWAS'2005)*, Kuala Lumpur, Malaysia, 2005.

[23] OASIS, Extensible Access Control Markup Language (XACML), http://www.oasis-open.org (Visited June 2005).

[24] C. Giblin, A. Y. Liu, S. Mueller, B. Pfitzmann, and X. Zhou. Regulations Expressed as Logical Models (REALM). *IBM Research Report RZ 3616,* IBM Research Division, Zurich, July 2005.