

Security Implications of Event-Driven Interactions in Service-Oriented Computing

Submitted to the 2005 International Conference on Service-Oriented Computing (ICSOC'2005)

Kees Leune and Jeroen Hoppenbrouwers

July 6, 2005

Security Implications of Event-Driven Interactions in Service-Oriented Computing

Kees Leune* and Jeroen Hoppenbrouwers**

Tilburg University, Infolab
P.O. Box 90153
5000 LE Tilburg
The Netherlands
{kees,hoppie}@uvt.nl

Abstract. In this paper, we address the question whether security of service-oriented computing in general, and web services technology in particular, is adequately equipped for any interaction patterns that are not request-response, such as event-driven interactions. Taking into account the distributed nature of access control, we argue that service-oriented computing needs message context level security, which captures the relationship of each message to other messages, and places it in the context of a services interaction sequence. We illustrate these requirements by discussing them from the point of view of the EFSOC framework.

1 Introduction

Service-oriented computing (SOC) is a computing paradigm in which distributed services cooperate via the exchange of messages which are sent and received by loosely coupled operations. SOC is manifested in the service-oriented architecture (SOA), of which the most successful implementation to date is realized by web services technology [1].

Web services technology commonly uses the familiar SOAP messaging framework to exchange messages. Although SOAP does not enforce any particular patterns in which messages must be exchanged, a frequently used implementation approach is the well-known request-response pattern. Here, the initial message that is sent represents a request to execute a certain operation. The invoked service will return a second message which will contain the results of the execution, or a fault code if the execution was unsuccessful. While request-response messaging undeniably has a place in service-oriented computing, we believe that an event-driven approach will increasingly become more important [2].

In Sect. 2, we explore what event-driven interactions are, and why they are suitable for service-oriented computing. Sect. 3 assesses the security implications

* This work has been partially funded by the Netherlands Organization for Scientific Research (NWO) as part of the PRONIR project.

** This work has been partially funded by the Netherlands Organization for Scientific Research (NWO) as a HEFBOOM grant.

of event-driven messaging. We describe the EFSOC framework in Sect. 4 and relate it to the previous two sections. Sect. 5 completes the paper with a summary and conclusion, plus our plans for subsequent work.

2 Event-driven service interactions

In an event-driven approach, services interact by exchanging events which flow dynamically between service providers and service consumers. Each event causes other services to respond and generate events in turn, resulting in a *global event cloud* which in theory makes service behavior observable and controllable [2].

Most event-driven approaches provide stateless and asynchronous message passing. This requires an application programmer to decouple the act of generating an event from the act of reacting to one, even if the event generation should cause a near-direct reply. Adopting an asynchronous approach facilitates loose couplings, and is often seen as a solution to achieve reliability and performance [3].

Event-driven interactions are characterized by a continuous flow of messages representing “things that happen.” Events are commonly represented by structured messages which are exchanged between the event generator and any number of event receivers, and carry contextual information in which the event is described, plus the circumstances in which it occurred. Such contextual information may be an explicit part of the event representation, or can be implicitly deduced from said circumstances.

There are distinct differences between event-oriented message exchanges versus request-response message exchange patterns when considering the routing and delivery of messages. A service which operates according to the request-response pattern will actively go out and attempt to discover other services to interact with. In an event-driven approach, selection of services to provide certain operations does not principally lay with the generator of the event. Instead, service providers express their interest in handling certain events. Based on those preferences, either an event broker or the requesting service will then decide which service will handle the event.

While the concept of a global event cloud is a useful metaphor to visualize a turbulent flow of events, it is important to realize that the “global” nature of that event cloud has only theoretical value in service-oriented computing. This is caused by the fact that each service operates as an autonomous and separate entity and is not subject to a form of centralized monitoring and/or control. For this reason, we partition the global event cloud into smaller *event spaces*. Each event space represents a group of collaborating services which interact by exchanging events, and changes over time as services join or leave the collaboration.

Event-driven interactions are only now gaining more interest of the WS-* community. Initiatives, such as WS-Eventing and WS-Notification are emerging and are starting to establish a firm foothold. Both initiatives will be briefly discussed in the following two sections.

2.1 WS-Eventing

WS-Eventing provides a specification which can be used by a web service to register interest with another web service in receiving messages about events [4]. Since WS-Eventing does not assume a specific interaction model, the specification allows clients to interact using mechanisms such as polling, call-back, wrapping events in SOAP messages, etc.

WS-Eventing provides a number of basic mechanisms that can be used to manage the subscriptions of services to specific messages, such as subscribe, renew, unsubscribe, etc.

The WS-Eventing specification recommends to use standards such as WS-security to achieve message-level security. While the issue of access control is touched upon, no recommendations are made, and no mechanism to achieve a system for access control is suggested.

All these mechanisms can be provided as value-added services to other services by so-called *subscription managers*. The subscription manager will handle all event-related operations, and relieve the service itself from this task.

2.2 WS-Notification

Event-based web services interaction has been explored by an industry consortium and has found its reflection in the WS-Notification family of related white papers and specifications, which define a standard approach to notification using a topic-based publish-subscribe pattern [5].

WS-Notification's pattern is based on the concept of a web service disseminating information to a set of other web services, without having knowledge of the nature or location of those services at service specification or coding time. This is achieved by the fact that the web services that wish to receive information register with the information producer.

Among the requirements that were used to define the WS-Notification standard are the ability to support both direct and brokered notification, but also the ability to transform and aggregate topics, and the requirement to provide runtime meta data. Subscribe request messages can optionally contain one or more filter expressions. This mechanism can be used to restrict the type of notifications that can be expected as a result of the subscription. They do not guarantee that no unwanted messages slip through.

At the time of this writing, the WS-Notification standard consists of a number of related documents. The WS-BaseNotification document specifies the web services interfaces for notification producers and notification consumers. It includes standard message exchanges to be implemented by service providers that wish to act in these roles, along with operational requirements expected of them [6]. WS-BrokeredNotification defines the web services interface for a broker which will act as an intermediary which allows publication of messages from entities that are not themselves service providers [7]. Finally, WS-Topics defines a mechanism to organize and categorize items of interest for subscription [8].

When using WS-Notification, web services must explicitly provide operations for subscribing and receiving events, unlike WS-Eventing where they outsource these operations to value-added service providers.

3 Security aspects of event-driving messaging

Security is a field which is commonly described in terms of authentication, authorization, privacy, and integrity. Some authors will also include availability as a security category. Authentication and authorization can be considered separately from privacy and integrity. The first two aspects are closely related to the business processes and need to be kept well-aligned as part of daily system support. Privacy and integrity are less organizational in nature, and are often achieved through the use of cryptographic solutions, such as digital message signatures and message content encryption.

Security in general, and access control in particular, often benefits from a central point of administration and/or enforcement. However, service-oriented computing adopts a perspective of individual and autonomous services which may interact directly. As a consequence, access control rules in service-oriented computing are administered and enforced in a distributed fashion as well. In Sect. 2, we partitioned the global event cloud into smaller event spaces. These event spaces also play a role in achieving an acceptable level of security. An event space spans all subjects which are currently engaged in a conversation that is relevant for the event message at hand. These subjects together have a momentary service state, and although the constellation of subjects does not follow preset rules, the service state must. This state is called a *context*, and when linked to a specific message, the *message context*. When services collaborate to get work done, their messages are all sent within a specific message context, and can influence access control rules and authorizations within that context. The role of message contexts is elaborated further in Sect. 4.3.

The security of a generic messaging framework can be considered as consisting of three layers, as shown in Fig. 1. The transport layer provides integrity and privacy between nodes of the infrastructure. The message layer adds protection for the message itself. The message context layer places the protected message in the proper relationship to other messages by adding service state.

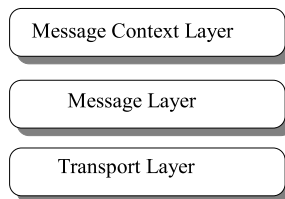


Fig. 1. Security layers

3.1 Transport-level and Message-level Security

Transport layer security attempts to protect the infrastructure over which the message is transported. Typically, such protection is achieved by introducing encryption tunnels, such as SSL. Unfortunately, SSL is vulnerable to attacks, such as the man-in-the-middle attack, which make it still possible that messages are intercepted and/or tampered with.

Likewise, message layer protection can be achieved by deploying proper cryptographic standards. Initiatives such as XML Signature [9] and XML Encryption [10] provide low-level support for signing XML messages. XML Signatures provides integrity, message authentication and/or signer authentication services for data of any type. XML Encryption is a standard for a process for encrypting/decrypting digital content (including XML documents and portions thereof) and the XML syntax used to represent it. Both standards operate directly on XML documents, and are schema-independent.

Web services security standards, such as WS-Security [11], WS-SecurityPolicy, WS-SecureConversation, etc., provide more elaborate approaches. WS-Security proposes a standard set of SOAP extensions that can be used when building security web services to implement integrity and privacy, supporting multiple tokens, trust domains, signature formats, etc. WS-SecurityPolicy [12] provides the mapping of the WS-Security standards onto the infrastructure proposed by the WS-Policy standard [13]. WS-SecureConversation is built on top of the WS-Security and WS-Policy models to provide secure communication between services. WS-Security focuses on the message authentication model but not on a security context, and thus is subject to several forms of security attacks [14].

All of the standards which are mentioned here have in common that they focus primarily on the (technical) privacy and integrity security aspects, to a lesser extent on (organizational) authentication, and not at all on (organizational) authorization. They provide security at the transport and message layers. However, service-oriented computing introduces additional requirements to a secure platform and needs a third layer.

3.2 Message Context Level Security

The message context layer is required to authorize operations on messages, provides that certain constraints are not violated and certain conditions are achieved. Such access control decisions must be made by a layer that has sufficient service knowledge, unlike the transport and message layers which are largely technical in nature. In the message context layer, service security rules are defined that span nodes, messages, and transactions. They are closely related to the services, but not at all to which service providers and consumers are actually engaged in the conversation.

Several existing initiatives touch on the message context layer, although none of them explicitly mention that they do. XACML [15] is an OASIS standard that provides an XML-based language to specify security policies, and describes a conceptual distributed architecture in which access control systems may be

deployed. XACML distinguishes several actors that play a role in an access control decision. These actors, such as policy enforcers or policy decision makers can be viewed as autonomous subjects who can use a common specification to arrive at an individual access control decision. An XACML profile tailored to the needs of web services is available, in which a number of aspects, such as reliable messaging, privacy, and authorization are addressed. However, the XACML profile for web services has not matured past the stage of a working draft and has also not been updated to the most recent XACML specification.

SAML [16] is a security assertion markup language which may be used to safeguard message exchange by allowing trading partners to share authentication and authorization information. SAML conveys assertions that may be validated by authorities. In addition to a set of predefined assertions, SAML offers a framework to define custom assertion types. Similar to WS-Security, SAML offers security at the level of message exchange. However, SAML messages can be deployed to implement message level security.

Summarizing, service-oriented computing needs message context level security. The message context captures the relationship of each message to other messages, and places it in the scope of a sequence of service interactions. Linked with security delegation to decentral nodes in the service network, this leads to our design of the EFSOC framework as discussed in the next section.

4 The EFSOC framework

Service-oriented technology is commonly deployed in a highly dynamic environment, in which services, service providers, and business rules alike may change often. As a consequence, security policies must be equally dynamic, closely following both the business rules and the environment. Traditional security policy definition and enforcement by a central facility through which all events are routed is not feasible, both for performance, availability, and flexibility reasons. Instead, service providers and service consumers must each specify and subsequently enforce their own access control policies, which quickly turns into a governance problem instead of a technical problem.

EFSOC is a multi-level event-driven framework for service-oriented computing, consisting of an event layer, a security layer, and a business process layer [17]. The event layer provides brokered event-driven message exchange functionality to services. The security layer adds service-aware security capabilities, with the emphasis on (organizational) authorization and access control. Finally, the business process layer provides the link between business processes requiring certain functionality, and the services that provide it. The primary focus of this paper is on web service security, and we will skip a discussion of the EFSOC business process layer at this time.

In this paper, we assume that a secure transport layer is available and will be used for all communications. EFSOC's event layer is mostly positioned at the message layer. Each message represents an event, and may include information about the message (such as sender or the time it was sent), and a payload in

the form of an event body, which may represent any information necessary for the proper interpretation of the event. Message-level security of event messages is achieved by deploying commonly available solutions, such as WS-Security, or XML Encryption and XML Signature.

The EFSOC security layer, and EFSOC business process layer are positioned at the level of message context, as they both relate messages to other messages. The distinction between these two layers is the fine line between what ‘must’ be enforced and what ‘should’ be enforced. The security layer blocks messages that are not allowed in their context and prohibits them from propagating. The business process layer routes messages to their intended target(s), depending on their context.

4.1 The event layer

Events play a pivotal role in the EFSOC framework as they serve as the elements providing the loose couplings between services. Events can be perceived as occurrences that happen over time, independently from each other [18].

Typical service-oriented solutions operate from the assumption that services interact directly—without using any kind of service broker—and in a point-to-point fashion. The request-response message exchange protocol illustrates this claim. Business interaction protocols for web services, such as BPEL, assume similar interactions. BPEL message exchange protocols revolve around the notion of partner link types which characterize the conversational relationship *between two services* by defining the “roles” played by each of the services in the conversation [19].

Traditional web service scenarios assume that service providers publish their contracts to one or more centrally located repositories. To represent that scenario in terms of event-driven computing, the service provider would begin by publishing certain event types, immediately followed by subscribing to them. Next, a service consumer may discover the event type, and generate the appropriate event to invoke the discovered service. This scenario is depicted in Fig. 2.

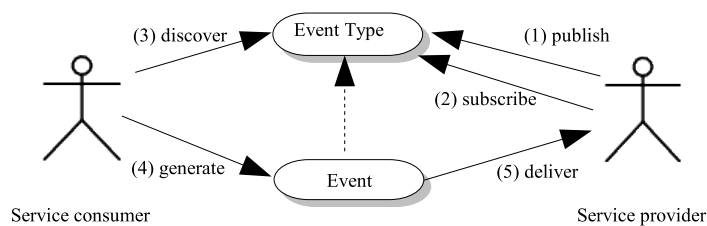


Fig. 2. Traditional web services event scenario

EFSOC acknowledges the fact that business interaction also can take place on a one-to-many level, where one service request or solicits responses of many

(possibly competing) service providers. Unlike the traditional event scenario, in which publishing an event type is closely coupled to the ability to process the associated events, EFSOC allows all services to publish event types without attaching any commitment to the publication. The distinction between event producer and event consumer is not made until at least one service decides to actually subscribe to the event type, in which case it will be considered an event consumer (and therefore a service provider).

Once an event type has been published, subjects can start generating events of that type. An event that has been generated, is delivered to all subjects that have subscribed to that event type (this subject set may be empty). This is further illustrated in Fig. 3. In this event scenario, the publication of an event type denotes that the service consumer is looking for a service provider who can produce the requested service. The service provider subsequently notices the request (i.e., discovers the event type publication) and subscribes to it, to become an event consumer for service requests. It is feasible that unfulfilled event type publications are used to create a service provider on demand, possibly leading to an active *service market* which is not only supply-driven, unlike the traditional web services model.

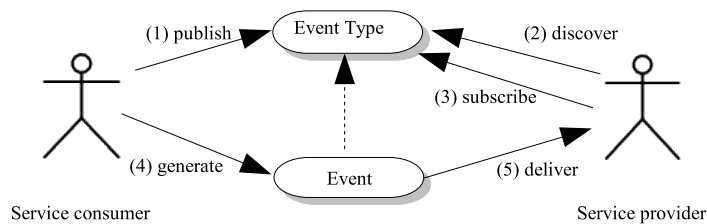


Fig. 3. Alternate web services event scenario

4.2 The security layer

The presence of a security layer on top of the event layer does not imply that the event layer comes without security. Traditional transport and message level security is assumed to be implemented at the event layer. The security layer is concerned with security beyond individual messages.

The EFSOC meta-model introduces subjects, roles, and messages, as shown in Fig. 4. It is loosely based on the concepts that are brought forward in role-based access control [20], augmented with discretionary properties. Authorizations can be granted to a *principal*. Principals are either specified by referring to a role, or to a named subject. When evaluating an access control rule, roles will be expanded to reflect the set of subjects which are currently playing that specific role, or to the set of subjects to who that specific role is assigned. The abilities

of subjects to manipulate roles are called role operations, and the abilities of subjects to manipulate messages are called message operations. Role operations and message operations can be parameterized as required.

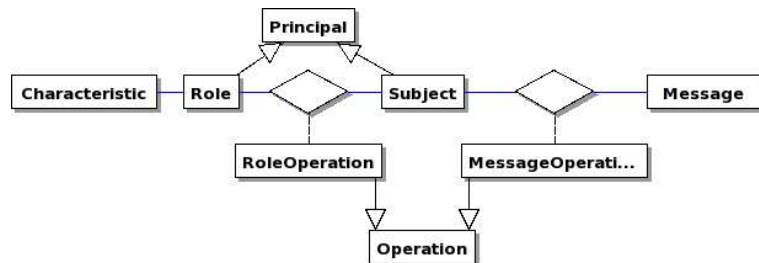


Fig. 4. EFSOC meta model

The principal role operations that are supported, are **publish**, **discover**, **assign**, and **activate**. All role operations are implemented as events; in other words, to publish a role, a subject would generate a **publishRoleEvent**. Allowing each subject to publish and discover roles leads to a naming problem. For example, two subjects might want to use the same role name, but do not realize that the other has already published it. The ramifications of this potential problem are discussed in more detail later in this paper, when namespaces are introduced. Once roles are published, they are ready to be used. Roles can be assigned to subjects by using the **assign** operation. Subjects are able to activate and deactivate the roles which are assigned to them at any time.

Authorization rules are rules which conditions must be fulfilled before a subject is granted a certain permission. For example, an authorization rule could specify that permission p_1 is only granted to a subject if that subject has role r_1 active. Each rule applies to one or more principals. If permissions are assigned to roles, a distinction is made between active roles and assigned roles, allowing for implementing separation of duty. Rule revocations will always take precedence over rule grants. In other words, if any access control rule grants a certain permission to a principal, but another rule revokes the same permission, the permission revocation will take precedence.

Access control is realized by restricting parameterized role operations or message operations. Restrictions are imposed by matching authorization rules with *access control rules*. Access control rules describe which conditions must be fulfilled for certain operations. For example, assume that a business scenario dictates that only subjects playing role r_1 may subscribe to events of type et_1 . An access control rule would specify that permission p_1 is required for operation **subscribeevent** of the event type is et_1 . Using the combination of authorization rules and access control rules, services can finely tune access to their operations. Access control rules are also specified in an XML specification.

Since event bodies must be explicitly published before they can be referenced, the structure of each event type is known. Using this knowledge, the condition which limits the scope of the access control rule can be specified using XPath expressions [21].

4.3 Contexts and Namespaces

To prevent naming conflicts between different subjects defining similar or even identical roles, event types, or permissions, EFSOC enforces *namespaces*. Specifically, when a subject decides to publish a certain event type, EFSOC requires the additional specification of a namespace. Using namespaces allows subjects to cooperate by sharing overlapping definitions, and provides an elegant and powerful way to avoid naming conflicts. This argument holds even when naming conventions do not carry implicit semantics, such as when using ‘113232’ as role name instead of ‘truckDriver.’ However, namespaces by themselves do not assist in *semantic* role matching or conflict resolution.

The context of an event follows from the formal service in which it is defined, and therefore is a formal concept as well. This suggests that the namespace of an event type should be determined by the context in which the event type will be used. Assigning namespaces to contexts, or to their associated business processes, provides for a straightforward and logical naming system. At the same time it suggests a way of structuring and sharing service definitions.

EFSOC namespaces can be implemented using WS-Topics, which provides a mechanism to organize and categorize items of interest [8]. WS-Topics’ categorization system is based on the well-known tree approach. Using a topic mechanism provides a powerful approach to segmenting the total topic space into smaller domain-specific segments, allowing for groups of cooperating subjects to share terminology with minimal efforts.

While WS-Topics provides the facilities for organizing topics in trees, there is no requirement to do so. It is perfectly acceptable to create a topic namespace which only consists of root nodes, as will often be the case in EFSOC. For example, while it may be useful to organize roles in a hierarchy, hierarchical event types offer little added value.

Several approaches are available to manage the semantic overlap between topic namespaces. The process has close parallels with thesaurus linking (in the library domain), ontology matching (in the semantic web domain), and some aspects of knowledge management. In all these research fields, relatively large collections of related concepts (thesaurus terms, subject headings, ontology nodes, knowledge facets) need to be related to other collections, while the individual collections remain independent and change continually. These relationships can range from semantically simple (‘search equality’, ‘comparable’) to very complex (explicit subsumption relationships which extend down the trees at both ends of the relationship). Although fully automatic ways to discover or negotiate topic space overlaps are not yet feasible (and whether they ever will be mostly is a philosophical question), manageable and scalable ways of manually maintaining the overlap have been developed [22]. These projects suggest that even an

informal or semi-formal linking scheme may already offer enough extra information that a system can make decisions which are ‘good enough.’ It is expected that knowledge formalization, as attempted by the semantic web movement, will eventually assist in topic namespace management. EFSOC itself does not focus on these issues.

5 Summary and conclusions

In this paper, we argued that event-driven interactions will find their way into service-oriented computing. Asynchronicity, which is closely related to event-driven messaging, provides a mechanism which allows loose couplings, reliability, and performance, which are all essential SOC characteristics.

Current initiatives to provide eventing and asynchronous notification to web services, such as WS-Eventing and WS-Notification, are a good first step. However, aspects like messaging security for service-oriented computing are not yet fully understood.

Security in service-oriented computing can be considered on three levels. The transport level provides a secure infrastructure, the message level provides reliable and confidential message exchange, and the message context level adds elements of organizational authorization and access control.

Work on web services access control, such as the AUTOFOCUS tool [23], often concentrates on design time aspects. The approach of EFSOC is to provide a multi-level event-driven framework to facilitate service-oriented computing. Its event model is flexible and can be used in a variety of messaging scenarios, and its security model explicitly decouples access control and authorizations. In addition, authorizations are computed dynamically, rather than assigned statically to roles and/or subjects.

We intend to focus on further development of the EFSOC security framework by developing the ability to temporarily delegate permissions, and by completing the formal underpinnings of the model. Additionally, we intend to investigate the possibilities of semi-formal context tree linking.

References

1. Papazoglou, M., Georgakopoulos, G.: Introduction to the Special Issue about Service-Oriented Computing. *Communications of the ACM* **46** (2003) 24–29
2. Luckham, D.: *The Power of Events. An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Addison-Wesley Press (2002)
3. Brambilla, M., Ceri, S., Passamani, M., Riccio, A.: *Managing Asynchronous Web Services Interactions*. In: *Proceedings of the IEEE International Conference on Web Services (ICWS’04)*, IEEE (2004)
4. Cabrera, L.F., et al, C.C.: *Web Services Eventing (WS-Eventing)*. Technical report, Microsoft, BEA Systems and TIBCO Software (2004)
5. Graham, S., et al, P.N.: *Publish-Subscribe Notification for Web services*. Technical report (2004)

6. Graham, S., Niblett, P.: Web Services Base Notification (WS-BaseNotification). Technical report (2004)
7. Graham, S., Niblett, P.: Web Services Brokered Notification (WS-BrokeredNotification). Technical report (2004)
8. Graham, S., Niblett, P.: Web Services Topics (WS-Topics). Technical report (2004)
9. Bartel, M., Boyer, J., Fox, B., LaMacchia, B., Simon, E., Eastlake, D., Reagle, J., Solo, D.: XML-Signature Syntax and Processing. W3C Recommendation, W3C (2002) <http://www.w3.org/TR/xmlsig-core/>.
10. Reagle, J.: XML Encryption Requirements. W3C Note, W3C (2002) <http://www.w3.org/TR/xml-encryption-req>.
11. Atkinson, B., et al, G.D.L.: Web Services Security (WS-Security). Technical report, Microsoft, IBM and Verisign (2002)
12. Della-Libera, G., et al, P.H.B.: Web Services Security Policy (WS-SecurityPolicy). Specification, IBM, Microsoft, RSA Security Inc., Verisign (2002) <http://www-106.ibm.com/developerworks/library/ws-secpol/>.
13. Bajaj, S., et al, D.B.: Web Services Policy Framework (WS-Policy). Technical report, BEA Systems, IBM, Microsoft, SAP AG, Sonic Software, Verisign, Inc. (2004)
14. Della-Libera, G., et al, B.D.: Web Services Secure Conversation Language (WS-SecureConversation). Technical report (2002)
15. Godik, S., Moses, T.: eXtensible Access Control Markup Language (XACML). OASIS Standard, OASIS (2003)
16. Farrell, S., Reid, I., Lockhart, H., Orchard, D., Sankar, K., Adams, C., Moses, T., Edwards, N., Pato, J., Blakley, B., Erdos, M., Cantor, S., Morgan, R.B., Chanliau, M., McLaren, C., Knouse, C., Godik, S., Platt, D., Moreh, J., Hodges, J., Hallam-Baker, P.: Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V1.1. Committee specification, OASIS (2003) http://www.oasis-open.org/committees/documents.php?wg_abbrev=security.
17. Leune, K., Papazoglou, M., van den Heuvel, W.J.: Specification and Querying Security Constraints in the EFSOC Framework. In: Proceedings of the International Conference on Service-Oriented Computing. (2004)
18. van den Heuvel, W.J., Leune, K., Papazoglou, M.P.: EFSOC: A Layered Framework for Developing Secure Interactions between Web-Services. (Distributed and Parallel Databases) (Publication pending).
19. Andrews, T., et al., F.C.: Business process execution language. Technical report, BEA Systems and International Business Machines Corporation and Microsoft Corporation and SAP AG and Siebel Systems (2003)
20. Sandhu, R., Coyne, E., Feinstein, H., Youman, C.: Role-Based Access Control Models. IEEE Computer (1996)
21. Berglund, A., Boag, S., Chamberlin, D., Fernández, M.F., Kay, M., Robie, J., Siméon, J.: XML Path Language (XPath) 2.0. W3C Working Draft, W3C (2005) <http://www.w3.org/TR/xpath20/>.
22. Hoppenbrouwers, J.: Multilingual access to subjects. Technical report, Tilburg University, Infolab (2001)
23. Deubler, M., Grünbauer, J., Jürjens, J., Wimmel, G.: Sound development of secure service-based systems. In: ICSOC '04: Proceedings of the 2nd international conference on Service oriented computing, ACM Press (2004) 115–124