Tilburg University

Exploring a multi-faceted framework for SOC

Leune, C.J.; van den Heuvel, W.J.A.M.; Papazoglou, M.

Published in:
Proceedings of the 14th International Workshop on Research Issues on Data Engineering

Publication date:
2004

Exploring a Multi-Faceted Framework for SOC: How to develop secure web-service interactions? (An extended abstract)

Kees Leune   Willem-Jan van den Heuvel   Mike Papazoglou
Tilburg University, Infolab,
The Netherlands

Abstract

Service Oriented Computing (SOC) demands an infrastructure that seamlessly integrates all connection points between business processes, services and associated support resources. Parts of the infrastructure may be supported by existing standards such as XACML and BPEL. However, an integral and sound framework that takes into account all these issues and serves as the formal underpinning of this infrastructure is currently lacking. A multi-facetted framework to enforce minimal levels of security not only at the level of the network (e.g., using encryption), but also of business processes, is of paramount importance.

To address this challenge, we explore an Event-driven Framework for Service Oriented Computing (EFSOC) that is organized in four tiers: the event tier, the business process tier, the resource tier, and the access control tier. The event tier encompasses definitions of business-related events, and supports their propagation throughout the business process flow. The business process tier specifies the dynamic interactions between business processes and services. The resource tier describes how service invocations interact with organizational resources, while the access control tier defines access roles that are allowed to invoke certain services.

1. Introduction

The manifestation of the Service-Oriented Computing (SOC) paradigm for Web applications is conducted on the basis of web services. Web service standards such as SOAP (Simple Object Access Protocol) [4], WSDL (Web Service Description Language) [5] and UDDI (Universal Description, Discovery and Integration protocol) [1] provide the foundations for building a service-oriented architecture (SOA). The SOA treats software resources as services available on a network. To achieve this, SOA relies on universally accepted standards to provide broad interoperability among different platforms, and loose coupling to separate the participants in distributed computing interactions, so that modifying the interface of one participant in the exchange does not affect the other. The combination of these two core principles means that enterprises can implement web services without having any knowledge of the consumers of those services, and vice versa.

Large scale web service deployments across multiple applications and services requires appropriate security, and particularly access control mechanisms are necessary to ensure that while a complex business process flows from one activity to the next, only authorized actors can invoke the supporting web-services. This requires deploying large-scale and reliable, Service Oriented Architectures based on the right blend of standards based technologies. This challenge is only partially tackled by current standards such as WS-Security, SAML, and XACML that are rather opaque and focus at an isolated part of the overall problem domain only, e.g., web service representation, or web service composition or web-service security. In particular, "traditional" security enforcement standards typically assume that roles can be structured hierarchically. However, this assumption is not any longer realistic in the SOA as it needs event-driven integration of connection points between business processes, services and associated support resources, i.e., back-end systems. On the other hand, emerging web-service standards tend to provide rather low-level, technical solutions while neglecting business semantics that are hardwired in business processes.

To address the challenge of developing secure interactions between web-services of various collaborating enterprises, a multi-facetted framework is required. This "unified" framework should be capable of associating external events, e.g., a customer (service requester) requesting to check the availability of a certain type of rental car, with the business process and services with which this event is associated as well as the resources that are invoked by these

services. In this way an organization is in a position to determine which services and business process are being handled by which actors at any point in time and what resources are affected, making it easier to safely adapt business processes in response to business changes.

In the remainder of this paper, we develop and explore such a unified framework that is entitled the Event driven Framework for Service Oriented Computing (EF-SOC) framework.

2. Related Work

As the framework that is presented herein is multi-facetted, this research draws upon results from various (virtually) independent domains.

Firstly, this research framework builds on top of the main manifestation of service flow languages: BPEL4WS (also known as BPEL) [2] is an standardization initiative that aims to define a notation for specifying business process behavior based on web services. The initiative has produced an XML specification for describing business protocols. Business protocols may be captured by specifying the service flow and sequence of service invocations. BPEL4WS assumes that all message exchange takes place in a point-to-point fashion. In other words, any business relation takes place between exactly two parties. The relationship between a service and a business process is described in terms of partner link types, which describe the roles that each partner may play in a conversation. This approach allows BPEL4WS to describe cross-business interaction as a large number of point-to-point message exchanges, without immediately specifying which role each business partner plays. A comprehensive overview of the main service composition languages, including a comparison, can be found in [13].

Although BPEL4WS includes a facility for exchanging messages between business partners, it assumes security to be an extension of the approach. In particular, security is enforced by supplementary standards, e.g., WS-Security and WS-SecureConversation. These standards help to secure message exchanges between parties just above the transport level (SSL), and are typically build on top of SOAP as encryption or signature headers.

SAML [8] is another initiative that stems from the SOC domain. In contrast to WS-Security however, this language is not defined in conjunction with WSDL or BPEL. In a nutshell, SAML 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, for example, an authorization decision assertion, SAML offers a framework to define custom assertion types. Similar to WS-Security however, SAML offers security at the level of message exchange, not at the level of business processes.

XACML [9] is a related initiative that allows security policies to be expressed in a common, XML-based, language. Using XACML, a common way to describe how authorization requests will be treated is available. XACML defines policies in terms of rules and combining algorithms. The XACML initiative is especially relevant to EFSOC as it is a candidate for implementing a part of the access control tier.

In contrast to the above security standards, the EFSOC framework security approach explicitly relates service invocations in a business process context to event types. This implies that security is not implemented as a set of SOAP extensions, but rather specified at the level of business events, processes and resources. However, we believe that the above standards could be used in conjunction with EFSOC to provide security at the "message-level". Moreover, the framework will offer a rich set of predefined queries to dynamically infer facts about security related issues. This feature is essential in the SOC-style eco-systems in which business processes operate in highly volatile environments and may change their resources from one moment to the next, e.g., due to load balancing, pricing or trust reasons. For example business process BP1 depends at moment X on two web-services from provider X, and the next, on services of provider A and B.

3. The EFSOC Framework

The EFSOC framework provides four tiers for supporting event-driven access control rules for invoking ports of business processes. In this way, the framework integrates points of access to services, which are enacted using roles, resources and events. The main constituents of each tier, and their cross-correlations are depicted in Figure-1.

3.1. Event Tier

Typically, enterprise business processes and workflows communicate with each other by continuously generating and responding to events. The Event Tier in the EFSOC meta-model aims at supporting such kind of complex events partly based on findings which are borrowed from the domain of Complex Event Processing (CEP) [12]. Events can be simply perceived as occurrences that happen over time, and independently from each other. Events represent a specific capability that is carried out by a service, and are capable of carrying information themselves, e.g., a request product event can be parameterized with a product identifier, date and quantity.

The Event Tier in EFSOC comprises two main components: subjects and events. A subject can be either a re-
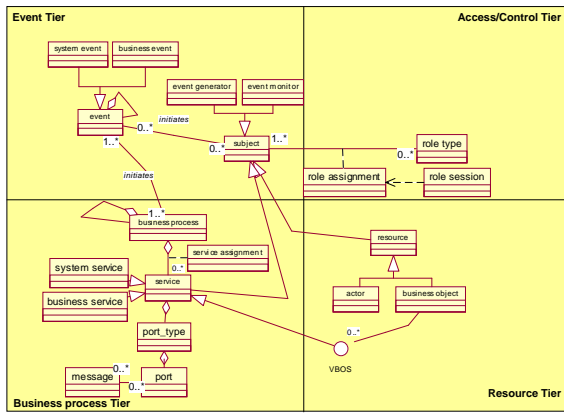
Figure 1. The EFSOC Framework

source or a service, and is capable of playing a certain role in the context of a business process. Subjects thus need not be necessarily human actors, but also can be implemented using an (autonomous) web-service. Events are generated and monitored by subjects and are typically part of an event chain. It should be noted that the internal structure of events (e.g., parameters, ECA rules) are for now left outside the scope of this article; only the basic framework is stipulated here.

3.2. The Business Process Tier

Our framework assumes that the coordination of web-service invocations are driven by business processes. A business process describes the flow of tasks that needs to be executed from a client request until the delivery of the required product or service, the information that is needed during each step, and the allocation of resources.

The business processes tier provides the (dynamic) allocation of services to specific tasks in the business process. This is achieved by defining a mapping operator between services and business processes. These mappings are named service assignments. Service assignments are persistent and may be queried run-time by a service flow engine to coordinate service invocation in the context of a particular business process.

To describe the features and behavior of a service we use WSDL, while the business process flow specifies the actual execution order of services described by WSDL interfaces. As observed before, several standards are proposed to specify and enact service process flows, e.g., the Business Process Execution Language (BPEL) [7], the Web Service Choreography Interface (WSCI) standard [6], and the Business Process Management Language (BPML)[3].

3.3. The Resource Tier

The Business Process Tier represents the abstract behavior of business processes in terms of service interactions, whereas the resource tier adopts a human resource- and implementation resource centric view of the services. The resource tier distinguishes between two kinds of resources; actors are active entities in an organization, such as employees, processes or agents and business objects (BOs) that are objects with well-defined business semantics, such as invoices, customers and purchase orders.

Actors are autonomous entities, e.g., human or e-agents, and reflect another alternative for enacting web-services. For example, the actor John Doe can implement a business service "Claim Management" by giving advice about insurances such as whether or not to buy damage liability coverage as a European customer in the US.

Access to a resource is provided subsidiary through a virtual business object (VBO). The purpose of VBOs is to hide the internal implementations of a resource in terms of enterprise back-end services. For instance, business objects are exposed to services via their interface, and thus VBOs act as technology neutral wrappers of resources such as existing information systems of legacy applications.

Regular services may not interact directly with a resource. For example, a VBO supports connectivity to back-end systems, provides aggregation and transformation of data, and allows for the dynamic presentation of information to authorized end-users.

As with any other service, the interface of a virtual business object service may be described using WSDL. A comprehensive methodology for mapping virtual object services to services has been scrutinized in [15].

3.4. The Access Control Tier

To reduce the risk of interacting with potentially unknown service providers, any approach that focusses on service integration in business processes must have strong security measures to ensure that all communications are adequately safeguarded against unauthorized disclosure or manipulation. Not only must the communication be safe, once messages have been safely relayed from the service provider to the service consumer (or vice versa), there must also be an adequate access control system in place that enforces that each service provider can only interact with business processes in exactly the way it needs to do to deliver its service.

The access control tier should be capable of taking into account several categories of interactions between services in an organization, including:

- subject to service: one-way interaction. Subjects are

active entities that are capable of producing events which may start processes of one or more services

- two way interaction: This category of interaction is also referred to as conversational interaction as it comprises pairs of message exchanges between services and or subjects

The Access Control Tier deals with allocating roles to subjects. A role factors cohesive behavior of subjects into some type, e.g., the subjects John Doe and Mary Doolittle both are specialized in advising clients about car insurances. From an implementation perspective, roles are used to prevent instances of web-service implementations (e.g, Enterprise Javabeans) to switch between classes. The linkage between a subject and a role type is typified using a role assignment. In role-based access control, permissions are associated with roles and roles are assigned to users [14].

Here, we refine the definition of a subject to make it explicit that any actor, service or business object is considered as a subject. Each subject may be assigned to one or more role types. A role type represents the function that a certain element fulfills in completing a business process.

Role types can be organized into role hierarchies, that are capable of indicate that one role is contained in another role. It has been proven that this containment relationship is irreflexive, a-symmetric and transitive [10]. Role type hierarchies play are an important means to substitute role instances and support the administration of access control privileges and constraints. Using role hierarchies, a role can be refined into another role that possesses stricter access control properties.

In the above, we stated that all events must originate from an authorized event generator and may only be sent to authorized event monitors. As role types can be assigned to subjects, they are a pivotal concept in our approach. A role assignment as such does not entitle the subject to any privileges. Instead, a role assignment must be explicitly activated, which results in a role session. Each role session is associated with exactly one role assignment.

A role session encompasses a set of access control policies that must be satisfied before a role, engaged in a particular role session, is allowed to invoke a capability of a service.

When the access control tier receives an event, it will evaluate an additional algorithm which will determine whether or not the event will be relayed to any possible event monitors. When the access control tier determines whether or not permission should be granted the event tier will receive the decision in the form of an authorization response event. Based on the contents of the authorization event, the access control tier will either relay the event, or prevent it from propagating further.
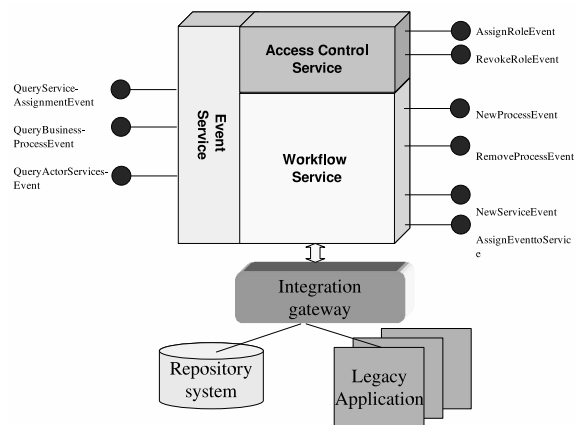

Figure 2. Infrastructure and Operational Services for EFSOC

4. Run-time Enactment of event-driven and service-based process with EFSOC

The above introduced framework provides the structural foundation for enabling even-driven business processes that are executed by authorized service resources.

In order to facilitate the run-time enactment of event-driven processes, the EFSOC framework basically utilizes two categories of services: infrastructure (system) services and operational (event) services. Thus, EFSOC is defined recursively as a conglomerate of composite infrastructure and operational services in itself. We have depicted the architecture of the supporting infrastructure and operational services in figure 2.

4.1. Infrastructure services

To implement the runtime enactment of the framework, we are developing a number of system services. Just like business services, system services are services, however, they are provided by the EFSOC framework instead of designed by the user. The EFSOC framework defines four system services: an event service that provides operations for sending, monitoring and relaying events, a workflow service which provides service invocation coordination and session management, an integration gateway service that servifies resources, and lastly, an access control service. The access control service is used primarily by the event service and the workflow service to establish whether or not events are allowed to be monitored or generated by subjects. The integration gateway provides a suite of wrapping capabilities, e.g., semantic wrappers and converters, to access resources such as legacy systems, and hereby gives support for building virtual business objects. For these pur-

```
<wsdl:interface name="accessControlInterface">
    <wsdl:operation
        name="requestAuthorization"
        pattern="http://www.w3.org/2003/06/wsdl/in-out">
    </wsdl:operation>
    <wsdl:operation name="newRoleType"
        pattern="http://www.w3.org/2003/06/wsdl/in-only">
    </wsdl:operation>
    <wsdl:operation name="newSubject"
        pattern="http://www.w3.org/2003/06/wsdl/in-only">
    </wsdl:operation>
    <wsdl:operation name="newRoleAssignment"
        pattern="http://www.w3.org/2003/06/wsdl/in-only">
    </wsdl:operation>
    <wsdl:operation name="addAuthorizedEventMonitor"
        pattern="http://www.w3.org/2003/06/wsdl/in-only">
    </wsdl:operation>
    <wsdl:operation name="addAuthorizedEventGenerator"
        pattern="http://www.w3.org/2003/06/wsdl/in-only">
    </wsdl:operation>
</wsdl:interface>
```

Figure 3. WSDL excerpt of the Access Control Infrastructure Service

poses, technology such as XSLT and semantic mappings are adopted by this component.

All system services are implemented as web services and WSDL descriptions are available [11] [1].

Figure 3 illustrates partially the interface description of the access control service, which provides a number of basic operations to the event service. For example, when a subject attempts to generate an event, the event service will invoke the requestAuthorization operation of the accessControlInterface. The operation will decide, based on a number of rules that must be evalatued, whether or not this subject is authorized to generate this event. Once the accessControlInterface has returned the decision, which is sent back to the event service in the form of a new event, the event will be relayed to the appropriate event monitors.

4.2. Operational Services

Operational services are constructed on top of the infrastructure services (see the event interfaces which are connected to the four components of the architecture), and serves to instantiate the types of the EFSOC metamodel, provide run-time support for event-driven processes, and, query the framework.

Event-driven processing may lead to large amounts of different events being generated and monitored. To address this issue, we have chosen to organize all event types in an event hierarchy as shown in figure 4. The total event space can be divided in three event categories: 1) definition event types, 2) execution event types and 3) query event types. Definition events are events that modify the existence
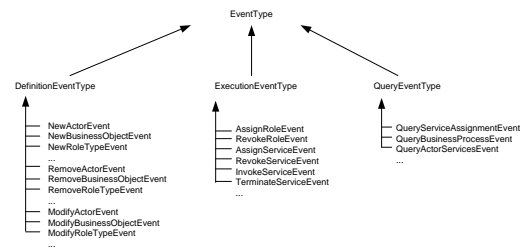
[1]WSDL definitions can be found at: http://infolab.uvt.nl/people/kees/efsoc/systemservices/wsdl

Figure 4. The (partial) Event Taxonomy

of model elements. For example, creating a new role type will trigger a `NewRoleTypeEvent`, or removing an actor will fire off a `RemoveActorEvent`. Execution events are events that represent changes in run-time properties of the model, such as service invocations, role assignments, etc. Query events can be used to inspect the state of the system. For example, to find out which actors are participating in a particular business process, a QueryBusinessProcessEvent will be generated.

The event taxonomy has been mapped to the infrastructure services. In particular, the definition and execution event types are supported by the Work Flow Service, excluding for those which are related to the access/control tier in EFSOC. The conformance of the result of execution event types is checked upon the existing definitions (specified using the definition events) using advanced query capabilities in the framework. These query facilities are implemented in the QueryEventTypes.

5. Summary and conclusions

SOC requires instantaneous process flows throughout the enterprise to ensure that business events are propagated along the chain of connections to the appropriate and authorized handling services and resources. Enterprises increasingly expose their internal business processes to external business events. Dealing with large numbers of incoming and outgoing events implies that organizations must keep control of both their internal and cross-enterprise business processes.

In this article, we have explored a four-tiered framework and some infrastructural services that could serve as the basis for securing interactions between trading partners. In contrast to existing web-service standards, this framework starts from the business processes and allows security of peer-to-peer interactions between trading partners. Actually, as argued in the article, this framework can be constructed on top of existing security standards such as SAML that merely provide (SOAP) message level security. In addition, and perhaps more importantly, the framework offers

a set of queries to dynamically infer new facts, e.g., web-service X can deduce whether an incoming request to invoke one of its port is authorized or not. This feature is of paramount importance as enterprises will have to operate in increasingly volatile business environments, with changing trading partners between one process execution and the next.

We realize that the current version of the EFSOC framework only provides the most basic functionality and constructs to specify and enact event-driven business processes for Service Oriented Computing. This framework encompasses four complementary tiers: the Event Tier, the Business Process Tier, the Resource Tier, and lastly, the Access Control Tier. This framework is capable of capturing external events and distribute them to associated internal business processes and services according to a predefined access/control scheme. There are a number of additional features, which would further enhance the usability and expressive power of the framework.

These include the following items:

- Extension of the EFSOC framework.

  The metamodel that is presented herein, typifies the basic constituents for complex event processing in SOC. We intend to elaborate both the metamodel and the underlying formalization so that they include a type system, which can more effectively deal with recursively defined events and business processes.

- Development of Unified EFSOC Language.

  We plan to work on a (formalized) unified language that combines and extends existing standards like BPEL, SAML and WSDL, and is based on a sound and concise formal metamodel.

- Query Language and Repository of Predefined Queries.

  We intend to design a query language and a repository system on top of the EFSOC language. In addition, we will predefine a rich set of queries to infer facts about the EFSOC framework.

An experimental system for enacting event-driven business processes within the EFSOC framework is under construction. This system is architected according to the organization of infrastructure and operational services that was presented in section 4.

References

[1] Universal Description, Discovery, and Integration (UDDI). Technical report. http://www.uddi.org.

[2] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S. Weerawarana. Business process execution language. Specification Version 1.1, BEA Systems, International Business Machines Corporation, Microsoft Corporation, SAP AG, Siebel Systems, May 2003.

[3] A. Arkin. Business process modeling language. Last call draft report, BPMI.Org, November 2002.

[4] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. F. Nielsen, S. Thatte, and D. Winer. Simple Object Access Protocol (SOAP) 1.1. W3c note, W3C, May 2000. http://www.w3.org/TR/SOAP/.

[5] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web Services Description Language (WSDL) 1.1. W3c note, W3C, March 2001. http://www.w3.org/TR/2001/NOTE-wsdl-20010315.

[6] A. A. et al. Web service choreography interface 1.0. Technical report, BEA Systems, Intalio, SAP, Sun Microsystems, 2002. http://dev2dev.bea.com/techtrack/wsci.jsp (visited 2-7-2003).

[7] F. C. et al. Business process execution language for web-services, version 1.1. Technical report, BEA Systems, IBM, Microsoft, May 2003. http://www.ibm.com/developerworks/library/ws-bpel (visited: 28-6-2003).

[8] S. Farell, I. Reid, H. Lockhart, D. Orchard, K. Sankar, C. Adams, T. Moses, N. Edwards, J. Pato, B. Blakley, M. Erdos, S. Cantor, R. B. Morgan, M. Chanliau, C. McLaren, C. Knouse, S. Godik, D. Platt, J. Moreh, J. Hodges, and P. Hallam-Baker. Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V1.1. Committee specification, OASIS, July 2003. http://www.oasis-open.org/committees/documents.php?wg_abbrev=security.

[9] S. Godik and T. M. (editors). eXtensible Access Control Markup Language (XACML). OASIS Standard, OASIS, February 2003.

[10] W. Jansen. Inheritance properties of role hierarchies. In *21st National Information Systems Security Conference*, Crystal City, Virginia, October 6-9 1998.

[11] K. Leune. Efsoc: Infrastructure services and prototype implementation. Infolab technical report, Tilburg University, October 2003.

[12] D. Luckham. *The Power of Events. An Introduction to Complex Event Processing in Distributed Enterprise Systems.* Addison-Wesley Press, April 2002.

[13] C. Pelz. web services orchestration: a review of emerging technologies, tools, and standards. Technical report, HP, January 2003.

[14] R. Sandhu, E. Coyne, H. Feinstein, and C. Youman. Role-based access control models. *IEEE Computer*, February 1996.

[15] W.-J. van den Heuvel. *Integrating Modern Business Applications with Objectified Legacy Systems.* PhD thesis, Tilburg University, Infolab, 2002.