

EFSOC: A Secure Framework for Event-Based Service-Oriented Computing

Kees Leune* Willem-Jan van den Heuvel

Tilburg University
Infolab, Department of Information Systems and Management
The Netherlands

E-mail: kees@uvt.nl , wjheuvel@uvt.nl

Abstract

Recently, a myriad of standards has been proposed to address several (related) issues for implementing the Service Oriented Computing (SOC) paradigm, including service invocation, description, composition, and security. However, what is missing is a consistent and solid framework that may serve as a canonical foundation on top of which existing standards may be unified, and new standards could be proposed. This framework should go far beyond formalizing service interfaces and should incorporate critical service primitives for securely blending software services with business processes, abstract services, service resources and the such.

To address this challenge, we introduce the Event-driven Framework for Service Oriented Computing (EFSOC) that is organized in three orthogonal tiers: the event tier, the business process tier, and the security tier. The event tier encompasses formalisms to rigourously define business-related events, and supports their propagation throughout the business process flow. The business process tier allows to specify dynamic interactions between business processes and services and the security tier defines authorization policies for service invocations.

*This research is sponsored by the Netherlands Organization for Scientific Research (NWO) as part of the PRONIR project.

1 Introduction

Web services are rapidly being adopted as the technology of choice for developing and deploying distributed applications. Web service technology is part of an emerging paradigm for modeling, analyzing, designing, developing, panning and monitoring platform agnostic services. In contrast to existing application development paradigms, this paradigm adopts a multi-dimensional perspective by using the concept of services as the main building blocks of both company processes and applications.

Web service standards collectively support a vision that is captured using the Service Oriented Architecture. SOA treats software resources as services available on a network [Papazoglou, 2003]. This new paradigm for building networked computing services, 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 does not affect any other. The combination of these two core principles implies that enterprises may implement web services without having virtually any knowledge of the providers of those services, and vice versa.

Portions of the Service Oriented Architecture have been materialized during the past years, adopting several standard specifications including SOAP, WSDL, UDDI, WSIF, BPEL, WSCI, DAML-S, WS-Security, WS-Transactions. However, what is missing is a consistent event-driven framework that is standards-independent, and may support the definition and evolution of (standardized) specification languages. In this way, the framework allows to combine existing standards, while guaranteeing their internal consistency. In addition, and perhaps more importantly, this framework allows to dynamically check context-dependent permissions of participants in business processes to invoke an external or internal port type.

In this paper, we present a secure framework for event-driven service oriented computing, called EF-SOC. Section 2 highlights the rationale for the framework and the assumptions that were made. Before outlining the basics of this framework, we then introduce a running example in section 3. The running example serves to illustrate, explore and partially validate the framework that is presented herein. Next, in section 4, a meta-model is introduced that contains three orthogonal tiers and serves as the foun-

dation of the EFSOC Framework, which serves to support run-time, event-driven and service-enacted processes. In section 5, a prototype implementation of the architecture of run-time support of the EFSOC framework is briefly discussed. This architecture is equipped with predefined queries for inferring facts about the EFSOC framework. The last section summarizes the results of this research and outlines future work.

2 Research rationale

Interoperability between organizations currently takes place via strongly coupled information conduits which impose limitations on an organization's ability to respond quickly to changing business environments. Joining new partnerships, or changing message exchange processes in existing relationships, often requires long and difficult negotiations with many stakeholders.

The SOC paradigm allows more dynamic business transactions between two or more trading partners. This is achieved by supporting asynchronous communication between the partner's business processes, treating them as self-contained, and loosely coupled units-of-work. Each party in a multi-process communication can have its own distinct conversation regarding the business transaction, e.g., communication about ordering or paying for parts.

To cater for these demands, EFSOC embraces an event-driven approach that is recursively defined and extended using events. In the framework, service end-points may be accessed by authorized participants only, depending on the context in which the service is invoked.

EFSOC distinguishes between security issues at three levels: business process level, invocation level and authentication level. The business process level supports the creation of a security context in which cross-enterprise processes are executed. The invocation level allows to define multi-lateral business protocols between trading partners regarding authorization policies that prescribe which actors in a particular business process are allowed to invoke particular services. These authorizations may in fact be delegated according to a predefined scheme. Policies at this level do not refer to low level data transport protocols such as SSL. Instead, they constitute of reusable authorization rules that may be associated with business processes, events or resources. Thirdly, the authentication level mechanisms allow identi-

fication of trading partner in various business process contexts.

Realizing trust plays an important role in cross-enterprise transactions, which requires not only implementation of security measures at the level of the communication (e.g., SOAP) channel, but measures to safeguard authentication and authorization of trading partners which are involved in a particular cross-enterprise transaction must also in place. Furthermore, in case of disputes, a clearly established audit trail needs to be available to ensure non-repudiation.

Trust, which we define as the extent to which parties exhibit expected behavior, may be increased by limiting the ability to exhibit unexpected behavior. Such limitations may be achieved by implementing adequate access control mechanisms.

Traditional access control mechanisms, such as discretionary access control (DAC) in which permissions are assigned directly to subjects, and mandatory access control (MAC) in which permissions are assigned based to hierarchical sensitivity labels, incur a heavy management overhead which makes them less usable for highly dynamic environments. Role-Based Access Control (RBAC) addresses these issues by introducing an indirection between users and permissions which is based on the concept of a role. In the EFSOC framework, we adopt an approach to management and enforcement of authorization policies which builds on top of the fundamental concepts of role-based access control.

In contrast to most conventional security strategies, the dynamic nature of service oriented computing not only implies that invocation of end-points is defined dynamically, but must be continuously adapted to accommodate changes in the business process, or its linked actors and resources.

These challenges can only be tackled by blending newly developed ideas with concepts that are emerging in current standardization initiatives such as WS-Security and SAML, which are often rigid and hard and costly to adapt. In addition, existing standardization initiatives often focus at an isolated part of the overall problem domain only.

3 Running example

Throughout this article, we will illustrate EFSOC with a realistic, but simplified running example, which is derived from the BPEL4WS specification [Andrews et al., 2003]. Figure 1 shows a business process

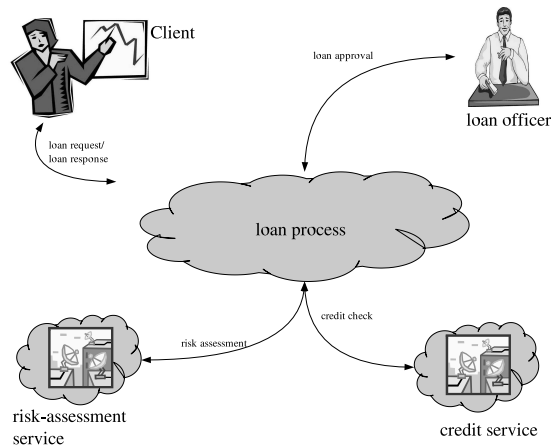


Figure 1: The running example

in which clients may request a loan. Business rules state that a loan of less than \$10,000 that is requested by low-risk individuals is automatically approved. In all other cases, loan approval is subject to credit checks and risk assessment. Based on the outcome of those checks, a decision is taken whether or not the loan will be granted.

In this example, we assume that the loan providing business process uses two services. The first service, which is the credit service, is used for credit checks and, by different business processes, for collecting interest and return payments.

The second service, is the risk assessment service, is exclusively used by the loan process to assess the amount of risk associated with a loan request.

4 Framework Essentials

EFSOC is divided in a number of orthogonal, functional tiers, which are supported by infrastructure services. Infrastructure services are a special kind of services which provide basic functionality to the separate tiers.

Using an semi-formal notation based on the Unified Modeling Language (UML, [OMG, 2003]), figure 2 depicts the core fabric of EFSOC. The meta-model distinguishes three related tiers: the event tier, the security tier, and the business process tier. The event tier provides facilities for defining business-

related events, and supports their propagation throughout the process flow. The security tier provides authentication and authorization facilities, while the business process service provides the interface to business processes and services.

4.1 The event tier

In the service oriented computing paradigm, interoperability is achieved by using loosely coupled services. The EFSOC framework assumes that such interactions are not limited to point-to-point interactions. Instead, we assume that processes are shared by (large) groups of business partners. An event-driven approach is a very natural approach to modeling possibly unstructured and unrelated interactions.

The event tier, supported by the event service, provides the infrastructure for generating, monitoring and relaying events. Each event has a predefined type and may be generated or monitored by subjects. Subjects, such as employees, business objects, agents, or services, are capable of generating and receiving events.

In accordance with findings of [Luckham, 2002], we identify two broad categories of subjects: event generators and event monitors. An event generator pertains to a subject capable of autonomously producing events, possibly triggered by an internal state change caused by an executing business rule, or by some internal business policy. An event monitor is an independent subject who registers incoming events and processes instances of the event types that it registered.

Events can simply be perceived as occurrences that happen over time, and independently from each other. Events represent a specific capability that is carried out by a subject, and are capable of carrying information. For example, a request product event can be parameterized with a product identifier, date and quantity.

Events consist of an event envelope and an event body. The envelope contains meta-information, such as an event type which represents the semantic significance of the event, and a timestamp which can be used to determine historic relativity of the event. In addition to the fields in the event envelope, each event has a body which contains information that the sender deems necessary.

Figure 3 shows the definition of an event represented in the EFSOC XML-Schema definitions. The

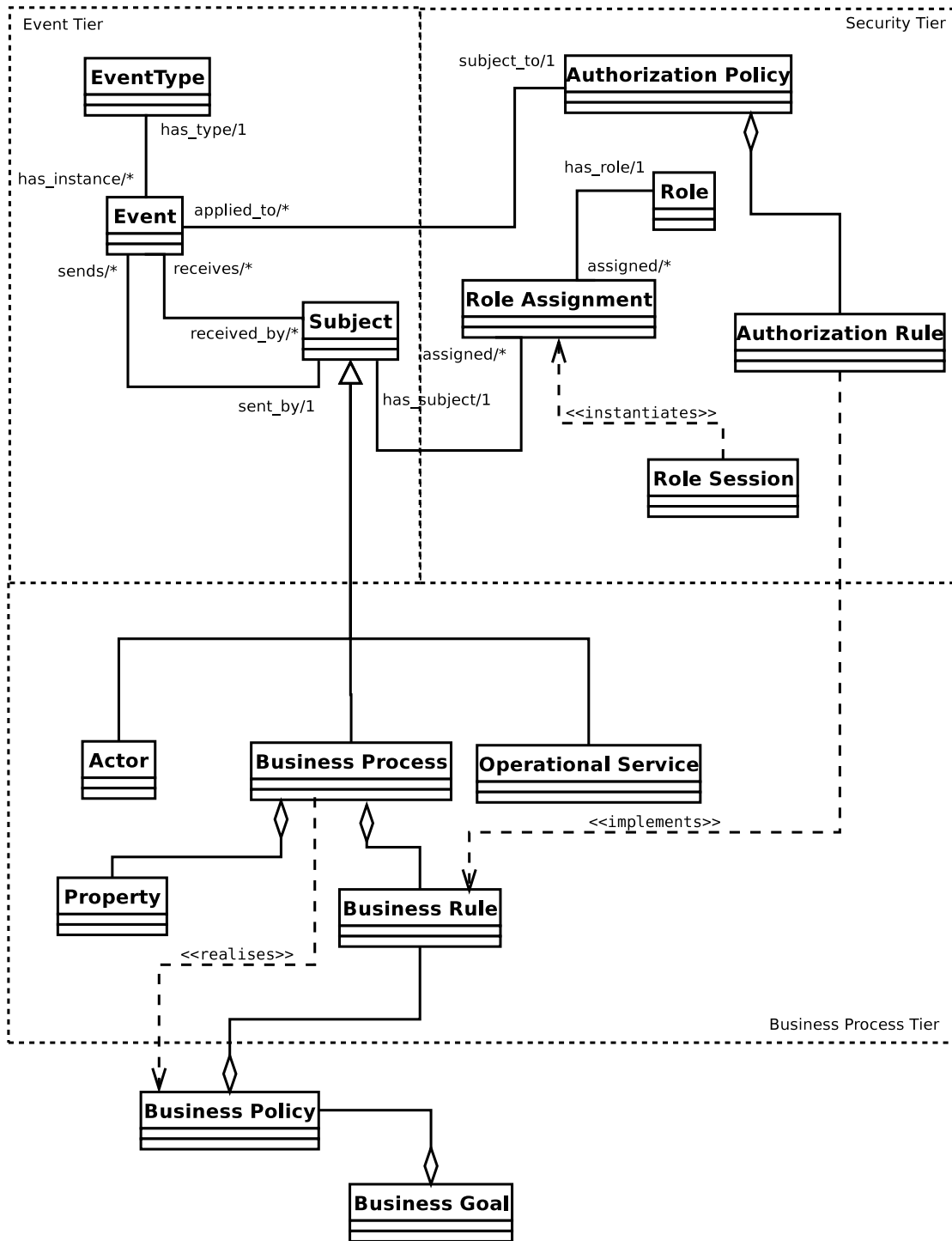


Figure 2: The EFSOC meta model

```
<event>
  <envelope>
    <eventtype>LoanRequestEvent</eventtype>
    <eventid>20040204193</eventid>
    <timestamp>1075890192</timestamp>
    <sender>john</sender>
  </envelope>
  <body type=loanRequestBody>
    <interestrate>0.05</interestrate>
    <paybackperiod>12</paybackperiod>
    <loanamount>10000</loanamount>
  </body>
</event>
```

Figure 3: An event representation

example adopts an event identifier that is generated using the date and a sequence number (February 4, 2004, sequence number 193) and a timestamp that is a standard UNIX timestamp. The EFSOC framework does not impose a requirement on the format of these fields, other than that the values must be meaningful to application that uses them.

As the event envelope and the event body can be regarded as related, but independent units of information, a separation of responsibilities may be introduced. We assume that the event envelope only contains system data, and as such may only be manipulated by system services. A subject, either sender or receiver of the event, may inspect the envelope, but cannot alter its contents. This approach enables the framework to take routing decisions and determine authorizations based on the envelope, without having to worry about the structure or contents of the event body.

Events are publicized (or generated) by calling the event service. Processes and services may register to monitor certain events. When a subject issues a request to subscribe to a particular event type, the security service will be queried to determine whether that subject is authorized to do so. If the security service allows the subscription to take place, subjects may receive event instances. Note that subsequent access control checks are performed when actual event instances are about to be relayed.

As large amounts of events of different types may be generated, it is useful to categorize event types in an event taxonomy as shown in figure 4.

In the event taxonomy, we distinguish between two elementary types of events: business events and

system events. Although the form of the event types in the business domain and the system domain the same, their semantics are fundamentally different. As the name implies, business events are directly relevant to the way an organization conducts business, whereas system events are merely used to keep the EFSOC system in a meaningful state.

Typical examples of system event types are `requestAuthorizationEventType` and `registerEventMonitorEventType`, while examples of business event types are `loanApprovalRequestEventType` and `riskAssessmentResponseEventType`.

System events can be further distinguished in definition events, execution events and query events. Definition events signify the introduction of new instances to the EFSOC model. For example, introducing a new employee to the organization will result in a definition event of the event type `NewActorEventType` to represent the creation of a new actor instance, and assigning the actor to a new role will lead to an execution instance of the event type `AssignRoleEventType`. Query events may be used to inspect the state of the model. Applications of query events may be to determine which actors participate in a certain business process, or to find services which participate indirectly in certain business processes.

Referring to the running example, we can define a number of business events:

- `loanRequestEventType`, which will initiate the loan process;
- `requestCreditCheckEventType`, which will initiate a credit check;
- `requestRiskAssessmentEventType`, which will initiate a risk assessment;
- `loanResponseEventType`, which signifies approval or rejection of the loan request that initiated the business process.

The dynamic nature of web services makes loose couplings between all participants a critical success factor [Alonso et al., 2003]. Service providers may be located anywhere in the service grid, only connected by the Internet. The limitations imposed by an unreliable medium include the ability to handle the introduction of new service providers, but also the ability to handle situations when a service provider is expected to be present, but cannot be reached for some kind of reason. An event-driven approach is able

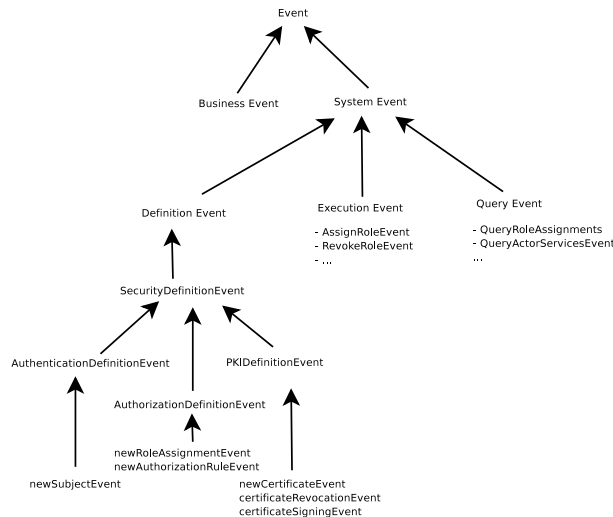


Figure 4: Event Type Taxonomy

to address these situations and enables organizations to rapidly adapt to the introduction of new business partners, services and business processes.

In our research, we defined an XML representation of events and event types. Using this representation, the loanRequestEventType may be represented as follows:

```

<EventBodyType name="loanRequestBody">
  <xs:complexContent>
    <xs:sequence>
      <element
        name="interestrate"
        type="xs:string"/>
      <element
        name="loanamount"
        type="xs:string"/>
      <element
        name="paybackperiod"
        type="xs:string"/>
    </xs:sequence>
  </xs:complexContent>
</EventBodyType>

```

A recent industry initiative adopts an approach similar to the event mechanism that we propose in EFSOC. WS-Eventing [Cabrera et al., 2004] is a specification which defines a protocol for one Web service (called an "event sink") to register interest (called a "subscription") with another Web service (called an "event source") in receiving messages about events (called "notifications"). WS-Eventing is

proposed by a consortium in which Bea Systems Inc., Microsoft Corporation and TIBCO Software Inc. participate. The event tier of EFSOC could be implemented on top of this standard using straightforward mapping rules. EFSOC goes further than WS-Eventing in that it not only supports a low-level event-mechanism, but includes the possibility to clearly define, and re-use, event-types.

4.2 The security tier

Service oriented computing assumes that business processes are formed by invoking a variety of services from distributed heterogeneous sources. Interaction between business processes and services implies that business processes invoke services, which may be provided by providers anywhere in the world. Such service invocations take place in a highly dynamical and rapidly changing environment.

To reduce the risks that are introduced by 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 communication between service provider and consumer is adequately safeguarded against unauthorized disclosure or manipulation. In other words, confidentiality and integrity of message contents and message protocols must be ensured. In addition to these basic requirements adequate authorization and authentication measures which enforce that each service provider can only interact with business processes in exactly the way it needs to do to deliver its service, must be in place.

As highlighted in the introduction, role-based access control associates permissions with roles and roles are assigned to users [Sandhu et al., 1996]. The indirection between (typed) roles and permissions decreases the management overhead that would otherwise be induced. A lower management overhead allows an organization to respond faster to changing environmental constraints. New actors, such as employees, business processes or services, may be introduced to the system ad hoc, without affecting the stability and integrity of the security and safety of the organization. Furthermore, by adapting authorization rules which evaluate to permissions, authorizations can be easily modified, granted or revoked without affecting the rest of the business processes. As such, role-based access control offers a powerful and flexible approach to dynamic enforcement of security constraints. However, role-based access control adopts a rather static and brittle concept of permission. In the EFSOC framework, permissions

are the result of the evaluation of business-driven authorization rules. As such, the permission model in EFSOC is more dynamic than an RBAC permission model.

The security tier thus provides primitives for describing authentication and authorization policies. We assume that roles are assigned to subjects, and permissions are the result of evaluating authorization rules, which are expressed in terms of roles. 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. The linkage between a subject and a role type is typified using a role assignment. We refine the definition of a subject to make explicit that any infrastructure service is considered as a subject.

Role types can be organized into role hierarchies, which indicate that one role is contained in another role. Organizing roles in hierarchies is a powerful mechanism for managing and querying the roles that have been defined. It has been proven that this containment relationship is irreflexive, a-symmetric and transitive [Jansen, 1998]. Role type hierarchies play 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.

The framework is currently policy-neutral, by which we mean that we do not presume any access control mechanisms. Thus, any approach that is desired from a business perspective can be used. Operational support for the security service is provided by a security service, in which it will be determined whether or not certain permissions will be granted. The security service communicates by monitoring and generating security event types specifically defined for this purpose. Services may generate authorization request events, which will be monitored by the security service. Using the data contained in the event body, as well as data from other sources, such as formalized business rules and access control policies, the security service will determine whether or not permission will be granted. Based on the outcome of the evaluation, the security service will generate a authorization response event that will be directed to the appropriate receiver by the event service.

Equivalent to the classification of event types, role types may also be organized in a taxonomy. We distinguish a number of system events, such as `businessProcessRoleType` and `operationalServiceRoleType` and allow application to introduce business events that build on these system events.

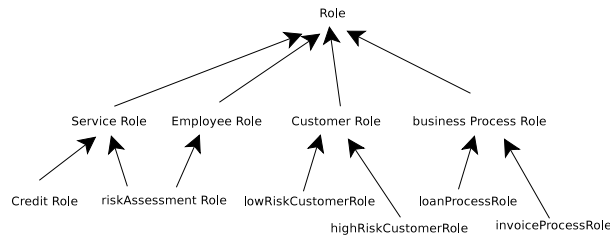


Figure 5: Role Type Hierarchy

Referring to the running example, we defined a number of business roles:

- customerRole; played by a customer,
- lowriskCustomerRole; a sub-role of customerRole, played by customers that match a low risk profile,
- highriskCustomerRole; a sub-role of customerRole, played by customers that do not match a low risk profile,
- loanProcessRole; a sub-role of businessProcessRole, played by the loan process,
- riskAssessmentRole; played by a risk assessment service or a risk assessment employee,
- invoiceProcessRole; a sub-role of businessProcessRole, played by the billing department for invoicing interest or return payments,
- creditRole; a sub-role of serviceRole, played by the credit service

Figure 5 represents the role hierarchy graphically. EFSOC not only supports static separation of duty, which may be used to prescribe that a subject may not be authorized for two or more conflicting roles, but also dynamic separation of duty to ensure that a subject may not act simultaneously in two or more conflicting roles.

The security service provides operational support for subject authentication and subject authorization, and for event integrity and event confidentiality. Subjects must be authenticated before they may interact

with any other subjects. The security service distinguishes two types of authentication: 1) partial authentication in which the identity of a subject is established and 2) full authentication where, in addition to establishing a subject's identity, the subject is also associated with one or more roles. After a subject has been authenticated, a session which determines to operational context of the subject is created. For example, an event which may be sent by a subject in which he requests the activation of a certain role may be represented as show below.

```
<event>
  <envelope>
    <eventtype>acquireRoleEvent</eventtype>
    <eventid>20040301029</eventid>
    <timestamp>1078148662</timestamp>
    <sender>kees</sender>
    <references/>
  </envelope>
  <body type="acquireRoleBody">
    <sessionid>123456</sessionid>
    <role>loanOfficerRole</role>
  </body>
</event>
```

This event represents the fact that a subject *kees* requests the acquisition of a role *loanOfficer* for a particular session. After processing the event, the security service will validate that user *kees* is authorized to send this particular event type. Subsequently, additional checks will be performed to validate that user *kees* is authorized for the loan officer role and that session 123456 is allowed to acquire additional session for this particular role session. These checks implement various separation of duty constraints, such as static or dynamic separations of duty.

When all checks return positive results, the current session will be closed and a new session, with the same properties of the previous session, plus the permission to play the loan officer role, will be granted to the session and a authentication response event will be generated.

The authentication response event, which is shown below, will be routed back to the original requester of the acquire role event.

```
<event>
  <envelope>
```

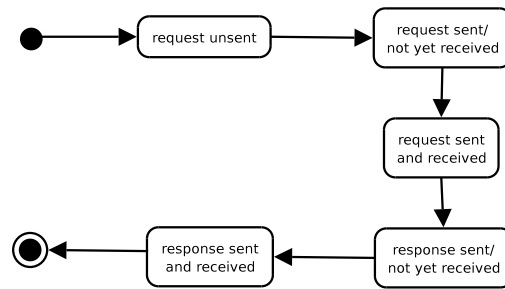


Figure 6: States in a request-response event cycle

```

<eventtype>authorizationResponseEvent
  </eventtype>
<eventid>20040301399</eventid>
<timestamp>1078149093</timestamp>
<sender>kees</sender>
<references>20040301029</references>
</envelope>
<body type="authenticationResponseEvent">
  <result>success</result>
  <sessionid>123473</sessionid>
</body>
</event>

```

Given the highly dynamic nature of the environment in which service oriented computing takes place, we assume that authorization policies may also change rapidly.

In addition to changing authentication requirements, business process may be terminated for a large variety of reasons. There may be (unexpected) runtime exceptions such as services being unavailable due to a variety of reasons, but also changing security consideration which mean that running processes may not be allowed to continue processing. We propose a system of active security, by which we mean that access control takes place throughout process execution and not only when the process starts.

The concept of active security requires that all active sessions are properly terminated as soon as the subject is no longer authorized. Our approach, allows to terminate event relay at any of the transitions that are depicted in figure 6.

For example, when permissions change causing a subject to become unauthorized, events of that particular type will no longer be relayed to that subject. Assume that a request has already been sent to the monitor and was properly received, but a response has not been sent yet.

When the response is generated, the event service will accept the event from the sender, but since the receiving subject is no longer authorized, the event will not be delivered to its end point and the sender will receive an exception notification.

Relevant work for the security tier takes is reported by several initiatives, such as XACML, SAML and more recently, WS-Security and WS-Policy. Moreover, W3C has recently published a working group note in which the outline a Web Services Architecture (WSA) [Booth et al., 2004]. However, a limitation of the WSA is that it is limited to describing requirements for a security layer in service-oriented computing, without describing how such requirements may be achieved. EFSOC complements WSA by not only describing a number of security requirements, but also by proposing an approach which addresses them.

XACML [Godik and (editors), 2003] provides an XML-based specification to represent security policies, including algorithms create new security policies that are based on existing ones. By describing all security policies in a similar fashion, manageability and re-use in an corporate environment is promoted. Additionally, when security policies are represented in a common, agreed-upon, format, it becomes possible to use the same policy for multiple applications.

SAML [Farell et al., 2003] provides a syntax and semantics of a specification for exchanging security assertions, such as authorization requests or attribute assertions. SAML assertions may be used as a mechanism to protect messages, such as event representations.

More recently, WS-Security [Atkinson et al., 2002] and WS-Policy [Box et al., 2003] have emerged. Both are specifications tailored to the requirements and limitations that are imposed by the use of web services. SOAP is an XML-based lightweight protocol for exchange of information in a decentralized, distributed environment, which is typically seen in such web service environments. The WS-security initiative describes SOAP enhancements to provide quality of protection, through message integrity, message confidentiality and single message authentication.

WS-security and SAML are closely related initiatives. SAML describes an independent mechanism to make security assertions and WS-security provides the mechanism that is required to integrate these tokens with SOAP technologies.

WS-Policy provides a mechanism that can be used by web services to specify policy information. Policies are not limited to security policies, as they can represent any characteristic, preference or requirement that might be necessary for a web service to function. WS-Policy does not define a relationship between policies and their relationship with services. This is achieved by yet another standard, called WS-Policy Attachment.

EFSOC adopts a holistic approach to policies, allow them to reflect business semantics in several contexts, such as authentication policies and business policies.

4.3 The business process tier

The business process tier addresses interaction properties of the different subjects, such as business processes and operational services.

Business rules are typically technology-independent descriptions of desired behavior. The subset of the business rules that applies to authorizations to send or receive events of certain types may be implemented in authorization rules. The remaining business rules are bound to heterogeneous implementations by explicit specifications.

A business process consists of a group of one or more related business rules. EFSOC is restricted to representing the operational features which are directly relevant to interoperability. This includes features such as service invocations, business process initiations or service terminations.

We assume that, within a context, business processes interact with other subjects exclusively by generating and monitoring events. As such, a business process must be uniquely identifiable within the context in which it operates. The events which are monitored by a business process may trigger business rules or authentication rules to activate and generate additional events in turn.

The business process service is an operationalization of the business process tier and offers runtime support for registering different kinds of subjects and describes the interaction between subjects by specifying which events are generated and monitored by each subject.

The following example, which is described in more detail in [Leune, 2004], shows a representation of the loan process as introduced in the running example.

```

<brule name="riskAssessmentAuth" id="R01">
  Risk assessment may be requested by
  a loan officer.
</brule>

<brule name="creditCheckAuth" id="R02">
  Credit checks may be requested by
  a loan officer.
</brule>

<brule name="riskAssessmentReq" id="R03">
  Risk assessment and credit check
  must be requested for loans
  over $10,000.
</brule>

<bpcontext name="loancontext">
  <bprocess name="loanprocess">
    <use rule="R01"/>
    <use rule="R02"/>
    <use rule="R03"/>
    <binding rule="R03"
      type="soapbinding"/>
    <monitors eventbodytype=
      "loanRequestEvent"/>
    <generates eventbodytype=
      "loanResponseEvent"/>
    <generates eventbodytype=
      "creditCheckRequest"/>
    <generates eventbodytype=
      "riskAssessmentRequest"/>
  </bprocess>
</bpcontext>

```

Previously, we stated that a subset of the business rules will be implemented in authorization rules. In the example shown above, rules R01 and R02 are examples of such rules. They deal exclusively with authorization information found in the event header, e.g. sender and event type. Rule R03 deals with information typically found in the event body, and as such must be bound to another implementation.

In service descriptions, like in business processes descriptions, the EFSOC framework limits itself to describing interoperability aspects of services. As web services in terms of WSDL models are invoked by sending messages to operations, interoperability between the EFSOC framework and WSDL-style

services is achieved by defining a WSDL-EFSOC mapping. The WSDL-EFSOC mapping is based on the assumption that for each operation one or more events are defined which will contain enough information for an operation invocation. In other words, by using the WSDL-EFSOC mappings, we will be able to map individual elements in the event body to corresponding elements in the operation invocation fields.

Note that the WSDL EFSOC mapping is inherently different from a binding in the terminology of the WSDL standard. WSDL bindings map service descriptions to implementations, whereas EFSOC mappings map elements in the event body to messages in the interface description.

A typical WSDL-EFSOC mapping could look like

```
<servicedescription
  name="creditservicedescr"/>
  <wsdl>http://pronir.nl/efsoc/credit.wsdl
  </wsdl>
  <mappings>
    <map eventbodyelement
      ="creditcheckrequest"
      servicemessage
      ="requestMessage"/>
    <map eventbodyelement
      ="creditcheckresponse"
      servicemessage
      ="responseMessage"/>
  </mappings>
  <invokers>
    <invoke eventbodytype
      ="creditcheckrequest"
      operation
      ="checkCredit"/>
  </invokers>
</servicedescription>
```

This service description assumes the availability of a credit service description in WSDL format at the provided URL. It then continues by mapping the body of a credit check request event type to the request message defined in the credit service's WSDL document which serves as input to the checkCredit operation. Finally, we specify that events of type creditcheckrequest will trigger the operation checkCredit.

Referring to the running example, we observe the following;

- a business process named `loanProcess` contains a number of business rules. The first business rule generates a `riskAssessmentRequest` event. The response to that event will be evaluated by second business rule, which will approve a client request if the client is considered to be a low-risk client and if the amount of the loan is smaller than \$10,000. If this is not the case, a third business rule will request a credit check by sending a `creditCheckRequest` event. The result of this request will be used to determine whether or not the loan request will be approved.
- The service description of the credit service is introduced to the system.
- The service description of the risk assessment service is introduced to the system.
- Each event will be associated with roles and it will be specified which events each role is allowed to monitor or generate.

Having defined one business process using two services, all corresponding role types and events, and having granted all roles appropriate event generating and event monitoring privileges, the model is now complete.

Two complementary initiatives to the work being performed in this tier of EFSOC are BPEL4WS [Andrews et al., 2003] and BPML [Arkin, 2002]. BPEL4WS, or BPEL for short, aims to achieve universal interoperability between applications by using web services. BPEL tries to reach this goal by defining an XML-based standard in which business protocols may be described. Two considerations with regard to the relationship between BPEL and EFSOC are that BPEL is driven from technology and although it has a conceptual foundation, it is mostly used to create static compositions. The EFSOC framework aims to capture more business semantics in its metamodel and it allows for semi-fixed or exploratory compositions of business rules, as well as re-use and re-allocation of policies and services.

Second, BPEL assumes that interaction between business processes and services is always based on point-to-point interaction. Both considerations have been addressed in the EFSOC project by defining an event-driven conceptual framework.

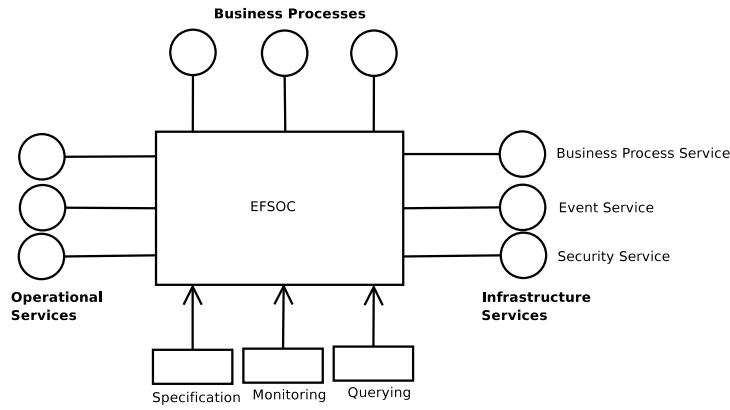


Figure 7: The EFSOC architecture

The BPML specification provides an abstract model and XML syntax for expressing executable business processes and supporting entities. BPML may have a more explicit conceptually rich foundation than BPEL, however it is less tailored to the service-oriented paradigm. Especially the loose couplings between organizations, business processes and services do not prominently manifest themselves. Regardless, BPML offers a very usable model to describe the inner workings of business processes and is therefore highly relevant to the EFSOC framework.

5 Prototype implementation

In this section, we will briefly discuss a prototype implementation of the framework that was presented in the previous section. The prototype has been designed to help focus the model on use in a realistic environment.

Figure 7 presents a simplified graphical representation of the EFSOC architecture. The architecture encompasses cooperating infrastructure services which support the definition and monitoring of the tiers defined in the framework. Infrastructure services are described in the same terms as operational services and as a consequence, they may be treated similarly.

The EFSOC management interface offers users support for a few primary tasks, such as creation of new role types, definition of business processes and assigning event permissions to role types.

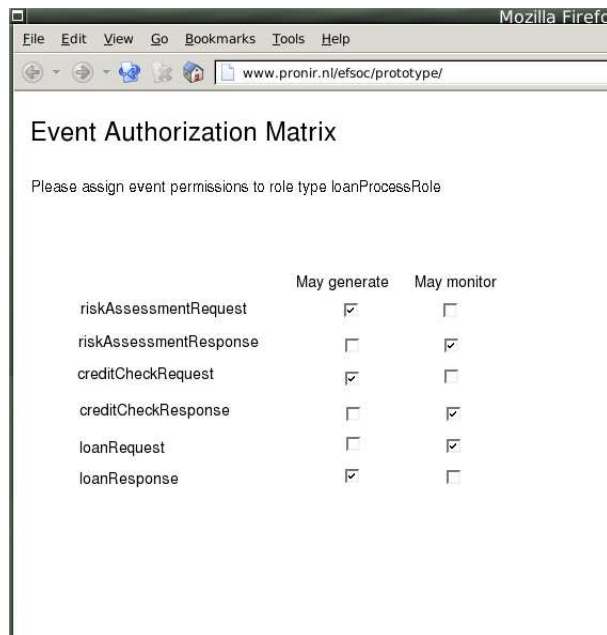


Figure 8: Event permissions management

After having defined business processes, and assigned appropriate roles to them, event permissions may be defined for role types that processes may play.

Figure 8 shows a partial screenshot of the management interface that is used to assign event permissions to a particular role type. It clearly shows that we are assigning event permissions to the loanProcessRole, which is a specialization of the BusinessProcessRole. The loan process role may generate a number of requests and while it is only allowed monitor one event.

Figure 9 contains a screenshot of the role hierarchy editor. The editor supports visualization of the role hierarchy and provides an intuitive interface to assigning subject to roles. In addition to the graphical interface, roles can also be managed via a command-line interface or by selecting the appropriate roles and/or actors from pulldown menus.

The prototype has been implemented using the PHP4 scripting language on a Postgresql database and will be made available at <http://maximus.uvt.nl/sigsoc>.

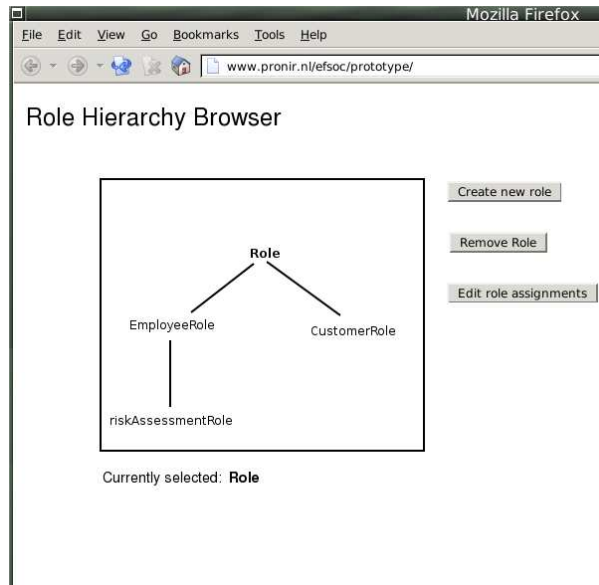


Figure 9: Role hierarchy editor

6 SUMMARY AND CONCLUSIONS

Service oriented computing designates an emerging paradigm for modeling, analyzing, designing, developing, planning and monitoring platform agnostic services.

Among the many exciting challenges to make this paradigm a reality is the relationship between business processes and services, which may be provided by external service providers. Assuming that services are provided by potentially unknown parties, a well-designed method for allowing interaction between the business processes of an organization and web services is required. Since web services typically manifest themselves in highly dynamic environments, organizations adopting the service oriented computing paradigm must be able to quickly and adequately adopt the way they do business, without sacrificing the integrity and protection of their resources.

Currently, many web services initiatives are specification or technology driven and do not focus on designing a conceptual architecture to which the work can be related. As a result of this, many standards and initiatives are currently ongoing or have completed, making it difficult for application developers to choose a development platform and really leverage the promise of service-oriented computing.

The Web Services Architecture (WSA) is an initiative in which an architecture is presented in terms

of functional components and relationships between components. However, WSA does not suggest a framework which will allow web service interactions to take place, or how service-oriented approaches interact with established business practices.

The EFSOC framework is framework for service oriented computing which takes an event-driven approach. The underlying assumption is that all parties involved in delivering business processes play a certain role to to achieve the desired results and that all coordination between business processes and services takes place in a service-oriented fashion.

To represent this, the EFSOC framework adopts a multi-tiered meta-model, consisting of a business process tier, an event tier and a security tier. In this paper, we discussed each tier in detail, illustrated by a running example, and related the framework to relevant standardization initiatives.

Our future work will concentrate on completing the prototype implementations of the model, including working version of all infrastructure services and making available detailed documentation of all event types and service descriptions and performing a case study to validate the concepts put forward in our research. In addition, our aim is to develop a query language which will allow us to proactively deal with changes such as the introduction of new services to a business process, or changing access control policies.

References

- [Alonso et al., 2003] Alonso, G., Casati, F., Kuno, H., and Machiraju, V. (2003). *Web Services: Concepts, Architectures and Applications*. Springer Verlag. ISBN 35404440089.
- [Andrews et al., 2003] Andrews, T., Curbera, F., Dholakia, H., Golland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I., and Weerawarana, S. (2003). Business process execution language. Specification Version 1.1, BEA Systems, International Business Machines Corporation, Microsoft Corporation, SAP AG, Siebel Systems.
- [Arkin, 2002] Arkin, A. (2002). Business process modeling language. Technical report, BPMI.org.

- [Atkinson et al., 2002] Atkinson, B., Della-Libera, G., Hada, S., Hodo, M., Hallam-Baker, P., Klein, J., LaMacchia, B., Leach, P., Manfredelli, J., Maruyama, H., Nadalin, A., Nagaratnam, N., Prafullchandra, H., Sewchuck, J., and Simon, D. (2002). Web Services Security (WS-Security). Specification Version 1.0, International Business Machines Corporation, Microsoft Corporation, VeriSign, Inc.
- [Booth et al., 2004] Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C., and Orchard, D. (2004). Web services architecture. W3c working group note, W3C.
- [Box et al., 2003] Box, D., Curbera, F., Hondo, M., Kaler, C., Langworthy, D., Nadalin, A., Nagaratnam, N., von Riegen, C., and Shewchuck, J. (2003). Web Services Policy Framework (WS-Policy). Technical Report Version 1.01, BEA Systems Inc., International Business Machines Corporation, Microsoft Corporation, SAP AG.
- [Cabrera et al., 2004] Cabrera, L. F., Critchley, C., Kakivaya, G., Lovering, B., Mihic, M., Orchard, D., Samdarshi, S., Schlimmer, J., Shewchuck, J., and Wortendyke, D. (2004). Web services eventing (ws-eventing). Technical report, BEA Systems Inc., Microsoft Corporation, TIBCO Software Inc.
- [Farell et al., 2003] Farell, 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., and Hallam-Baker, P. (2003). Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V1.1. Committee specification, OASIS. <http://www.oasis-open.org/committees/documents.php?wg-abbrev=security>.
- [Godik and (editors), 2003] Godik, S. and (editors), T. M. (2003). eXtensible Access Control Markup Language (XACML). OASIS Standard, OASIS.
- [Jansen, 1998] Jansen, W. (1998). Inheritance properties of role hierarchies. In *21st National Information Systems Security Conference*, Crystal City, Virginia.
- [Leune, 2004] Leune, K. (2004). Conceptual overview of the efsoc infrastructure services. Technical Report itrs018, Tilburg University, Infolab.

- [Luckham, 2002] Luckham, D. (2002). *The Power of Events. An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Addison-Wesley Press.
- [OMG, 2003] OMG (2003). *OMG Unified Modeling Language. Specification Version 1.5, formal/03-03-01*, OMG.
- [Papazoglou, 2003] Papazoglou, M. P. (2003). Service-oriented computing: Concepts, characteristics and directions. In *Proceedings of the 4th International Conference on Web Information Systems Engineering (WISE 2003)*.
- [Sandhu et al., 1996] Sandhu, R., Coyne, E., Feinstein, H., and Youman, C. (1996). Role-based access control models. *IEEE Computer*.