An Access Control Metamodel for Web Service-Oriented Architecture

Christian Emig, Frank Brandt, Sebastian Abeck
Research Group Cooperation & Management
Universität Karlsruhe (TH)
76128 Karlsruhe, Germany
{ emig | brandt | abeck } @ cm-tm.uka.de

Jürgen Biermann, Heiko Klarl
iC Consult GmbH
Keltenring 14, 82041 Oberhaching
Germany
{ biermann | klarl } @ ic-consult.de

Abstract — **With the mutual consent to use WSDL (Web Service Description Language) to describe web service interfaces and SOAP as the basic communication protocol, the cornerstone for web service-oriented architecture (WSOA) has been established. Considering the momentum observable by the growing number of specifications in the web service domain for the indispensable cross-cutting concern of identity management (IdM) it is still an open issue how a WSOA-aware IdM architecture is built and how it is linked with WSOA's main elements, the web services providing functional core concerns. In this paper we present an access control model for WSOA and a blueprint of a WSOA-aware authorization verification service which is part of the IdM architecture. We show the integration of this service with WSOA consisting of both basic and composite web services. Our solution has been tested and evaluated in an implementation case study.**

Keywords — Access Control Model, Identity Management (IdM), Web Service-Oriented Architecture (WSOA), Metamodel, Policy Decision Point (PDP), Business Process Execution Language (BPEL)

I. INTRODUCTION

A. Background

Currently most enterprises try to align their business processes with the supporting IT by migrating towards web service-oriented architecture (WSOA). Web service technologies are commonly recognized as a promising way for the implementation of SOA. However, WSOA is not meant to be built from scratch but rather the functionality of existing systems and their components have to be leveraged to web services. Bottom-up approaches start with the existing software systems and ease traditional application integration: web services feature standardized interfaces described using WSDL (Web Service Definition Language) as well as a standardized communication protocol, namely SOAP, which both are commonly accepted. The integration process can then be applied by the composition of web services of heterogeneous underlying software systems using process execution languages like BPEL (Business Process Execution Language) [1]. Top-down approaches start with business processes and focus at their model-driven mapping down to basic and composite web services. They enable business analysts to perform so-called programming-in-the-large, the system-independent orchestration of business-related (web) services along business processes [2].

B. Motivation

Besides the development of WSOA's core concerns (cf. to aspect-oriented programming, [3]) there are several cross-cutting concerns that have to be addressed before being able to go productive with WSOA – a central one is to enable security, especially access control. Access control consists of authentication and authorization verification. Looking at the mass and complexity of the existing and upcoming specifications in the web service security area like WS-Security, WS-Trust, SAML, XACML or the Liberty Alliance's stack proposal, it is comprehensible that software developers often neglect the web service security part. Additionally, state-of-the-art IdM suites are not yet prepared for WSOA [4] as well as current application servers often do not as yet support a necessary combination of relevant IdM standards. This is why currently the existing web services in most cases have little or no security features. Complications even increase when composing several web services which provide functionality from different underlying applications – workarounds like using the applications' built-in IdM are no longer applicable; an overall IdM architecture for WSOA is needed.

In this paper we enhance and improve existing access control models in respect to WSOA. The development of an appropriate access control model is a highly relevant prerequisite because the migration towards WSOA implies a strongly increasing number of objects (i.e. web services) combined with a looser coupling between subjects and objects. State-of-the-art models like role-based access control (RBAC) [5] do not scale in WSOA and need to be enhanced [6]. Secondly, we present an architectural blueprint for a WSOA-aware IdM authorization verification service following this access control model and motivate its integration with WSOA's core concern part. Therefore we extend the view on WSOA from the functional perspective towards access control. Leveraging the functionality of existing applications to interoperable services interfaces using application servers (so-called "wrapping to web services") allows the composition of these web services, technically implemented using process execution environments like BPEL engines. Though slicing down existing applications for the encapsulation of their core functionality at service interfaces, access control has to be enforced – at least to the same extent as it has been previously within the application boundaries.

The contributions of this paper are:

1. An **access control metamodel for WSOA** revealing all relevant sets and relations necessary to calculate authorization verification requests. The goal is to respect WSOA-specifics like the loose coupling of basic and composite web services as well as the increasing amount of both subjects and objects – a fact that prevents scaling of existing models.
2. A **WSOA-aware authorization verification service** implemented as a policy decision point (PDP) applying our access control model. This service is a central element of the IdM architecture needed to enforce access control for WSOA's core concern services, both basic and composite. From the core concern perspective, the complexity of the IdM architecture is to be encapsulated at a minimum set of service interfaces which should not have domain-specific characteristics.

The paper is organized as follows: section 2 introduces the architecture of web service-oriented architecture (WSOA) and derives the requirements for an appropriate access control model and the corresponding IdM architecture. In section 3 we discuss related work on access control models and propose our enhancement towards WSOA. In section 4 we motivate the design of a WSOA-aware authorization verification service implementing our access control model; the integration with WSOA's core concern part is explained. In section 5, our approach is applied and evaluated in a case study. A conclusion and an outlook on future work in this area close the body of the paper.

II. Web Service-Oriented Architecture and the Requirements for Identity Management

To be able to derive requirements for an access control model for web service-oriented architecture (WSOA), an overview of the architecture of WSOA's core concern structure is given in this section.

A. *Basics of Web Service-Oriented Architecture*

The basic WSOA layering as depicted in figure 1 consists of existing applications at the bottom layer that are wrapped to (basic) web services. The wrapping is done using application servers applying the design patterns proxy or façade as described in [7]. Web services can be composed at an integration layer using BPEL and web portals are used to integrate the (human) users utilizing existing web technology like web browsers. A key driver for WSOA is the closer alignment of business processes with their supporting IT. This is why the focus in IT changes from the (internal) view on systems and application towards operated and quality-assured IT services [8]; as a result, the aforementioned further layers are introduced and put on top of the existing applications. These services are defined at standardized interfaces at the basic web services and the integration layer using the UML ball / socket notation combined with a WSDL/SOAP constraint (cf. figure 1). Using standardized interfaces eases the traditional integration process especially in heterogeneous environments, i.e. with existing applications of different brands with vendor-specific and incompatible interfaces, depicted at the legacy systems at the bottom of figure 1. Additionally it allows flexible service reuse in different business processes. The description of this common core of WSOA can be found in many publications [9, 10, 11, 12].

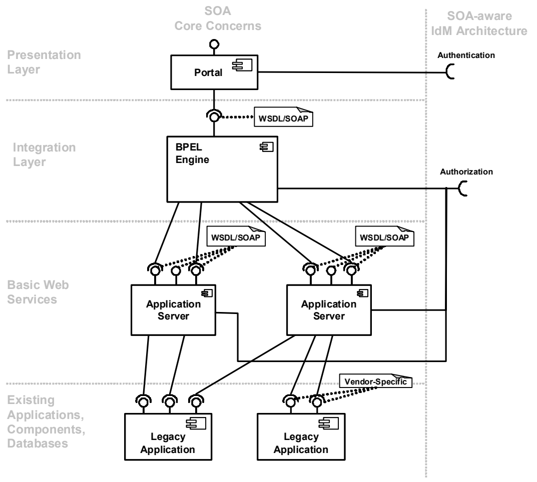


Figure 1. Architecture of Web Service-Oriented Architecture

It is important to notice that in the web service context, SOA does not imply strict layering. Web services can be accessed either directly or via one or many intermediaries like BPEL engines. From WSOA's viewpoint the service interface of a BPEL-composed web service is not distinguishable from a basic one as they are both described using WSDL.

B. *Requirements for Identity Management*

Besides further enhancements of WSOA's core concern part, fundamental questions arise: how is access control to be handled in this highly distributed and service-oriented environment? Slicing down existing applications to business related services, the internal IdM structures of the legacy systems are cut off. The alignment of the different system-specific IdM access control models and techniques with the goal of a local handling inside the applications complicates the integrated view on identity management. Therefore the development of a WSOA-wide, cross-cutting IdM architecture is favored. Being SOA-aware itself, this architecture is meant to expose its functionality at service interfaces decoupling core concerns from IdM, especially access control [13]. Following the paradigm of lose coupling [9] and separation of concerns [14], the IdM part of SOA's core concern services should be reduced to the bare minimum.

Access control is based on two prerequisites: first of all, an authentication process checking any possible credentials. This can be done once with validity for a series of subsequent accesses (relates to a single sign-on approach) or on every access. User authentication can be handled at WSOA's portal layer and will not be discussed further in this paper. Secondly, an authorization verification process is needed which checks if permission has been granted for the authenticated subject to invoke a WSOA service. In the following we will focus on this process which is to be encapsulated to a service of the IdM architecture.

Basic and composite services should not be put in charge of verifying caller's authorization themselves but they are to be enabled to clearly separate this task from their core concerns towards an authorization verification service. Therefore they should hand over all relevant data (according to our WSOA access control model) to the authorization verification service allowing each application server and BPEL engine to handle access control by calling the authorization service of the IdM architecture as depicted in figure 1.

III. An Access Control Metamodel for Web Service-oriented Architecture

One major goal of identity management is to establish effective access control that is the restriction of access to resources. To enable access control in WSOA, an access control model has to be developed adhering to WSOA's specifics. We describe this model as a conceptual metamodel. This metamodel defines the sets and relations on which a concrete access control decision can take place.

A. Related Work on Access Control Models

Formal access control models build the mathematical foundation to restrict access to resources. In 1969, the basics of access control were described very abstractly but formally for the first time [15]. Here the concepts of "subjects" and "objects" were introduced and it was suggested to link them using an "access matrix". This paradigm is now referred to as identity-based access control (IBAC) as the permissions are linked directly to the identity (i.e. the identifier) of the requesting subject without further levels of indirection.

Efforts to develop a complete mathematical formulation of access control have been undertaken in the early 1970s [16, 17]. Most of this work was sponsored by US defense sector. The common sense at this point of time was that there is a set of active entities, called subjects, and a set of passive (i.e. protected) entities, called objects. To control the access to objects, "security policies" were introduced basically consisting of two elements; first the type of access request: if "observation" (i.e. read) and / or if "alteration" (i.e. write / append) was requested. Secondly, an ordered set of security classification (e.g. unclassified, confidential, secret, top secret) was combined with a set of formal categories. A pair of classification and category was called "security level". Access control required a subject's security level to dominate the security level of the object. These models are called lattice-based access control (LBAC). In [18] their limitations are revealed: LBAC models are only effective for certain coarsely-grained security scenarios like in the military and lack both flexibility and scalability.

To overcome these limitations, the next step in evolution is role-based access control (RBAC) as introduced in [19] and refined in [5, 20, 21]. Although there are many different forms of RBAC, all RBAC models have in common that there is a level of indirection in the subject / object relation by focusing the business role which a subject is performing. Access permissions of an object are then linked to roles instead of individual subjects. Because access permissions do not have to be repeatedly assigned and maintained on a basis of individual subjects – a number that is constantly increasing – RBAC both significantly reduces administration overhead and scales much better than both IBAC and LBAC.

With software engineering heading for WSOA the traditional access control models need to be put to test again. In WSOA, the amount of objects to be protected increases significantly. Additionally the subject / object relation, instantiated in a consumer / provider link, is meant to be much more loosely coupled. In [6] a new access control model called attribute-based access control (ABAC) is introduced. Unlike IBAC and RBAC, in the ABAC model permissions are defined merely on any security relevant characteristics of subjects and objects, known as attributes. The goal is to almost completely decouple the subject / object relation by independently defining attributes of subjects, objects and environment state. It is a logical enhancement of IBAC and RBAC as the identity itself as well as the roles can be mapped to attributes. Policies on different abstraction levels that are defined on regular expressions consisting of attributes of subjects, objects and environment parameters enforce access control.

B. Heading for a WSOA-aware Access Control Metamodel

A problem shared by all aforementioned models is that the "action" is always reduced to basic system operations like *read*, *write*, *delete*, *execute* etc. This is problematic in WSOA as there the objects to be protected are at a different granularity. The most atomic object to restrict access to is a web service operation. It is important to notice that the web service operations are more similar to functions in programming languages that are executed with a defined set of parameters than like simple data objects that are manipulated. Therefore the set of actions as defined in RBAC has to be refined. In the web service context, it is not enough to define access to a web service operation based on a combination of permissions like *read*, *write* or *execute*. For example, a subject might be allowed to invoke a web service operation allowing the retrieval of personal data – but only if this is personal data about the caller himself. Hence, the parameters of the web service operation invocation must also be considered in the access control model as well. Basically this enhances the traditional RBAC actions which can be mapped to the fixed value of *execute* but additionally takes the invocation parameters into consideration for access control. A problem of ABAC as described in [6] is the indirection between objects and their access permissions. This unnecessary indirection increases complexity as the development of appropriate "loosely-coupled" policies is difficult

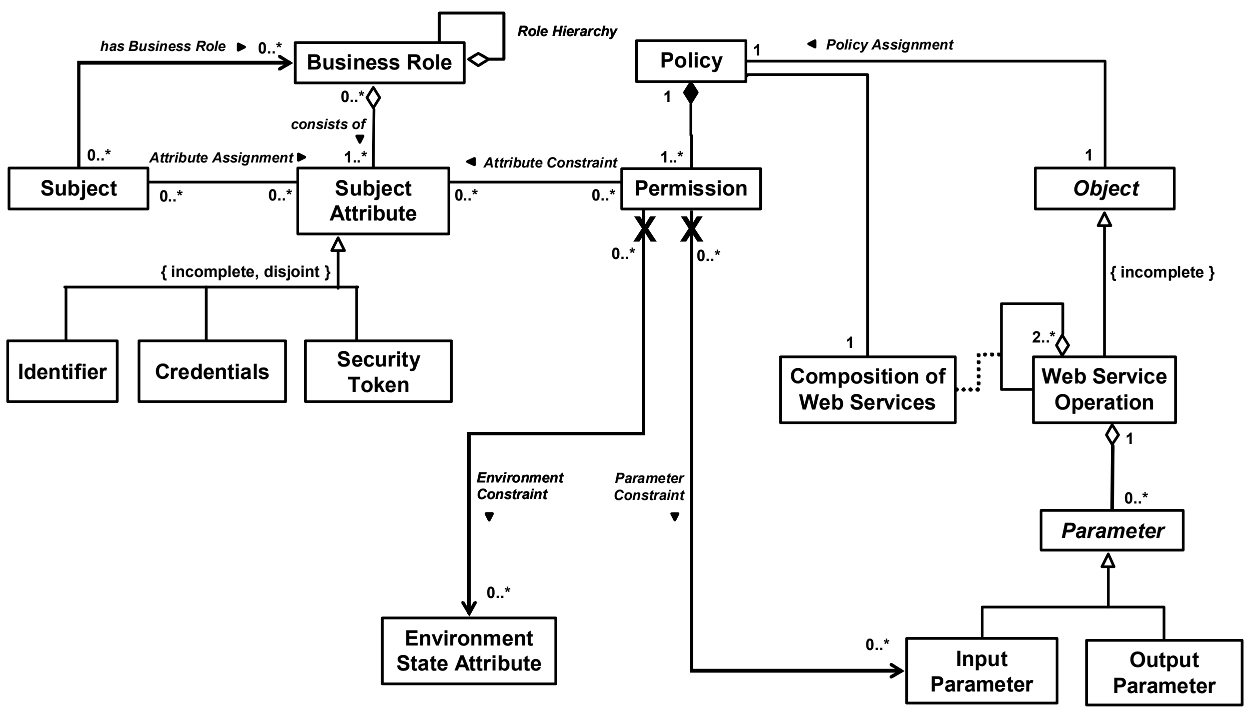


Figure 2. Metamodel for Access Control in Web Service-Oriented Architecture

A further specific of WSOA which must be dealt with is explicit service composition. Composition takes place if a web service calls different other web service operations and returns a combined result. The access restriction to the composed service has to be at least the sum of the restrictions of all underlying operations it is composed of and that are invoked mandatorily. This allows checking authorization at an earlier stage (i.e. the BPEL-composed web service) thereby limiting unnecessary calls ending in rollback operations if particular permissions for invoked basic web service operations are missing.

In figure 2 we present our metamodel for access control in web service-oriented architecture (WSOA) using a conceptual model in the UML 2.0 metamodeling approach to define the sets and relations used to enforce access control. This conceptual model is the first step towards development of a UML profile which can be used for a model-driven development of access control policies, which is not the focus of this paper. Our metamodel is an enhancement of the combination of hierarchical RBAC [5] and ABAC [6].

The central element of this model is *Policy* which is the composition of *Permissions*. *Permission* itself defines the traditional *Subject* / *Object* relation for a single service usage context. *Permissions* are always positive in our metamodel conferring to the ability of a subject (characterized by its attributes) to perform some action on the associated object. In access control literature negative permissions which deny rather than confer access to an object are sometimes discussed. In our metamodel denial of access is the default behavior and if a permission is granted, it has to be modeled explicitly. Nevertheless, it is possible to use negations in *Permission's* constraints. *Permission* combines one *Object* (related via the *Policy* towards which it is aggregated) and a set of *Subject Attributes* with the possibility to have constraints considering the *Object*'s associated *Input Parameters* and the *Environment State* (like date, time or any other attribute related to neither *Subject* nor *Object*). There are some special *Subject Attributes* that we explicitly modeled as the subject's *Identifier*, the *Credentials* and a *Security Token* (which is of temporary validity, i.e. refers to a session context).

In WSOA, *Subjects* can be either human users or active system components (i.e. self-acting services). The fact that there is a possible 1:n relation between a human user and a *Subject* (i.e. a user having more than one identity) is not explicitly modeled here as it is not relevant for the definition of access control. Users having more than one identity instantiate independent and different *Subjects*. *Subjects* are characterized by a defined amount of *Subject Attributes*. From the business perspective, *Subjects* act in the context of a *Business Role*. In our model, the concept of *Business Role* relates to a defined amount of (finer-grained) *Subject Attributes*. *Role Hierarchies* can be defined as well, all together finally mapping to a set of *Subject Attributes*. We do not focus directly on (business) roles for access control so there is no association between *Business Role* and *Permission*. This is a major difference to RBAC as defined by [5]. Furthermore it is possible to derive *Subject Attributes* from the role concept of RBAC: either the role itself can be defined as a single *Subject Attribute* carrying the role's name as the value or it can map to a set of *Subject Attributes; Permission* then links towards these *Subject Attributes*. This is both a major difference to RBAC as we do not link the permission to a (coarse-grained) *Business Role* but to a combination of *Subject*

Attributes as well as to the ABAC approach where the (business) role is explicitly not in focus. We call the composition of all *Permissions* for one specific *Object* (which is in our case a *Web Service Operation*) *Policy*; so the authorization of each *Object* is defined using exactly one *Policy*. One goal of SOA is the reuse of existing services in different contexts. This is why we use the concept of *Permission*; each *Permission* covers one service usage context. The *Policy* is the composition of all *Permissions* of an *Object* using Boolean "OR" concatenation.

A major advantage of our metamodel is that we remove the commonly used type of operation (e.g. read, write) [5] while placing the *Input Parameters* of the *Web Service Operation* into focus. So there are two relations from the *Permission* towards the *Object*: a direct one towards the *Input Parameter* (not backwards navigable) following the idea that a *Parameter* does not need to know if its value is evaluated for access control and an indirect one via the *Policy*.

Service composition is one goal of service-oriented architecture. This is why we explicitly address it in our metamodel. A *Composition of Web Services* consists of multiple invocations of other *Web Service Operations* in a specific order [22]. It has a web service interface like the basic web services consisting of operations. It can not be determined if the service interface is a composite or a basic web service. But service composition relates to access control in respect that there should be the possibility to pre-verify authorization at the layer of composite web services to be able to stop execution in case of missing authorization at an early stage. The composition aspect, modeled as an association class of the aggregation of *Web Service Operations* in figure 2, enables the authorization verification even at a composition level. For all *Web Service Operations* that are obligatorily invoked by the *Composition*, their *Policies* have to be added to the (overall) *Policy* of the *Composition of Web Service* using Boolean "AND" concatenation.

IV. DEVELOPMENT OF A WSOA-AWARE AUTHORIZATION VERIFICATION SERVICE

The access control metamodel formally specifies the sets and relations that are used to model permissions for access control in concrete scenarios – the permissions are used to calculate an authorization verification decision. In this section we propose how to map this access control model to a WSOA-aware authorization verification service.

The challenge can be divided into three parts: first of all, the definition of the service interface towards WSOA's core concern part; secondly, the development of the business logic which implements the functionality behind the service interfaces. Last but not least, it is to be specified where and how to store the data which is needed to verify authorization. Which data repositories already exist in WSOA and how to integrate them with the IdM architecture must be taken into consideration. To describe our setup, we use a middle-out approach beginning at the service interfaces following the IdM data layer and finally describing the business logic. Our focus is on authorization verification, so we neglect the authentication and administration part of the IdM architecture.

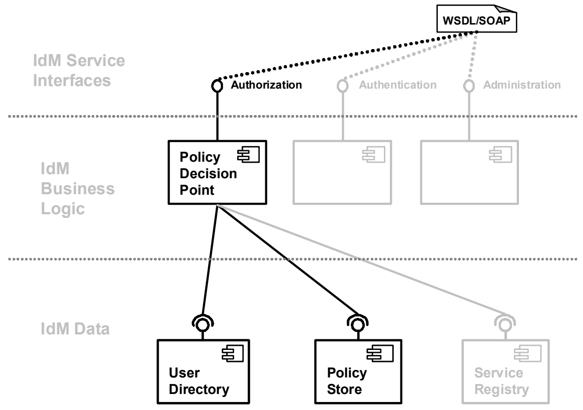


Figure 3. WSOA-aware IdM Architecture Focusing Authorization Verification

A. Service Interface Definition

In the following we outline the central operation of the authorization service using WSDL 1.1 [23].

```
<definitions name="Authorization" ...>
  <types>
    <!-- Definition of char, string, ArrayOfChar,
         ArrayOfString -->
  </types>

  <message name="AuthZ_Msg_Req">
    <part name="security_token" type="ArrayOfChar" />
    <part name="object_id" type="string" />
    <part name="input_parameters" type="ArrayOfString" />
  </message>
  <message name="AuthZ_Msg_Resp">
    <part name="result" element="boolean" />
  </message>

  <portType name="Authorization_SOAP">
    <operation name="Authorization_Verification">
      <input message="AuthZ_Msg_Req" />
      <output message="AuthZ_Msg_Resp" />
    <operation>
  </portType>

  <binding ... />
  <service name="Authorization" ... />
</definitions>
```

We concentrate on the definition of the messages that are exchanged. The incoming message carries the subject's security token, which speaking very generally is an array of 8-bit characters. The object identifier is required as well; we put it to the type of string as not in all scenarios are the identifiers only numeric. The input parameters of the web service operation are sent as an array of strings. The message response carries the decision of the authorization verification request as a Boolean value.

The authorization verification service can be called by any application server as depicted in figure 1. According to the access control metamodel, inside the application servers nothing has to be evaluated, only the input parameter of the invoked web service operation along with the caller's security token have to be combined with the web service operation's (unique) object identifier and sent to the authorization service. The return value is a Boolean value (true/false). This enables efficiency through separations of concerns for WSOA's core concern part. It can be called from both composite (i.e. BPEL) and basic web services.

B. Data Repositories

1) User Directory

The user directory stores information about subjects and their attributes according to our access control model. There are many ways of implementing a user directory besides a single-system approach, like meta directories, virtual directories or a directory replication network. This is not the focus of this publication; we assume that the business logic can access the data. Subjects are assigned a unique identifier to be able to distinguish them. During authentication, a subject's credentials are verified and a time-limited security token is returned that is piggybacked with every web service operation call.

2) Policy Store

Web service operations can be reused in different scenarios. As a consequence, an access policy of an object is usually a combination of permissions of the individual access contexts. To enable efficient authorization verification, we suggest aggregating the permissions for each object using a disjunctive normal form (DNF). DNF is a standardization of a logical formula which is a disjunction of conjunctive clauses. The conjunctive clauses are the context-sensitive permissions and the disjunction (represented by a Boolean OR) concatenates the different contexts. An access policy is stored as a pair of object identifier and the DNF-style expression.

3) Service Registry

When developing an IdM architecture, the link to WSOA's service registry should not be neglected. Here all information about WSOA's services, both composite and basic, is stored. The goal is to extend this existing data store by adding a unique object identifier to each of WSOA's web service operations at deployment. This is the object identifier which is used in the policy store to add the access policies to each web service operation.

C. Business Logic

The component handling authorization verification is usually called policy decision point (PDP). The PDP does the verification if a subject is allowed to invoke a web service operation. First of all, the access policy for the object is retrieved from the policy store. As the PDP does not receive the subject's identifier itself but the subject's (temporary) security token, there has to be a lookup, if the token is valid and which subject it can be mapped to. The mapping is done towards the subject's identifier. Then the subject attributes and the environment attributes defined in the policy are obtained and finally the policy is evaluated to a Boolean value.

D. Bridging the Gap Towards WSOA's Core Concerns

The IdM architecture needs to be linked to WSOA's core concern part consisting of basic and composite web services. The question where to put the policy enforcement point (PEP) which invokes the authorization verification service has already been discussed in [24]. Considering the architecture of web service-oriented architecture as depicted in figure 1, the PEPs are put as distinctive components into each application server, both the ones hosting basic web services as well as those hosting BPEL engines which provide support for composite web services. This enables each of the components implementing a single web service operation to do a call to the applications server's local PEP which handles the communication with the PDP.

V. IMPLEMENTATION EXPERIENCE

In this section we exemplarily present how we put our access control model and authorization verification service into practice within an integration project being pursued at our university.

A. Case Study

In our case study, we focus on two different applications which are used to generate certificates in the university context, so called transcripts of records (ToR). One of these two systems is SAP R/3 Campus Management, the other one is a HIS system. Both are commonly used at European universities. The starting point, depicted on the left side of figure 4, had already been implemented: ToR generation is implemented as a BPEL process using Oracle SOA Suite with Oracle BPEL process manager. The wrapping to web services of the SAP and HIS systems had been done using JBoss Application Server and BEA Weblogic. At each application server, a policy enforcement point (PEP) following the "secure service agent" design pattern is installed handling the communication towards the authorization service [24].

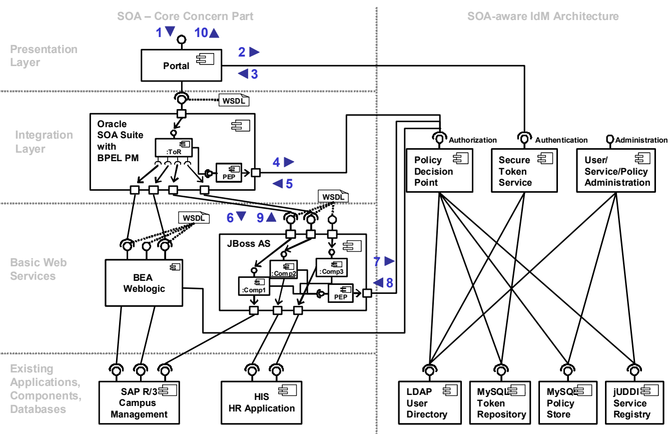
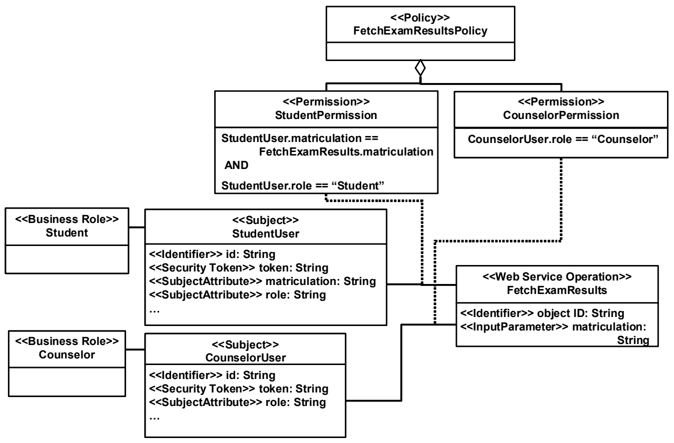


Figure 4. Case Study: Securing BPEL-composed Generation of Transcript of Records

Now the challenge is to derive the policies for the generation of a transcript of records (ToR). First, we start with an informal description. We assume that there are (at least) two different usage contexts:

- A student wants to get a ToR. He is only allowed to get his own ToR.
- A student counselor needs a ToR for consultation. He is allowed to get the ToR of any student.



<<Policy>> FetchExamResultsPolicy
<<Permission>> StudentPermission
StudentUser.matriculation == FetchExamResults.matriculation
AND
StudentUser.role == "Student"
<<Permission>> CounselorPermission
CounselorUser.role == "Counselor"
<<Business Role>> Student
<<Subject>> StudentUser
<<Identifier>> id: String
<<Security Token>> token: String
<<SubjectAttribute>> matriculation: String
<<SubjectAttribute>> role: String
...
<<Web Service Operation>> FetchExamResults
<<Identifier>> object ID: String
<<InputParameter>> matriculation: String
<<Business Role>> Counselor
<<Subject>> CounselorUser
<<Identifier>> id: String
<<Security Token>> token: String
<<SubjectAttribute>> role: String
...

Figure 5. Exemplary Access Control Model for ToR Generation

In figure 5 we exemplarily show how this informal model is depicted as a UML model at M1 level using our access control metamodel. On the right hand side, there is a basic web service operation, which retrieves exam results for the matriculation number that is handed over. Each web service operation has a unique object identifier. As it is used in two contexts, there are two permissions that are explicitly modeled and combined in the policy. The first context depicts a *Student* who tries to execute this operation; than the *StudentPermission* is evaluated. In case a *Counselor* tries to get access, the *CounselorPermission* is verified. The combination of both *Permissions* forms the *Policy* which is attached to the *Web Service Operation*.

Now we place this model into a real world system. The signature of the web service interface of the ToR generating service (which is BPEL-composed) is quite simple, just the matriculation number is needed for which the ToR should be generated. The name of the operation's input parameter is 'matriculation'. In our implementation, we use "s" as abbreviation for "subject" and "param" for "input parameter". Environment state attributes like date or time can be accessed using "esa". Following the "secure service agent" design pattern, the parameter carrying the subject's security token is automatically added to the operation's signature. When the web service is published in the service registry, an object identifier is assigned for each operation. This identifier is then used in the policy store to enable a quick lookup for the policy.

Object ID	*Access Policy*
…	…
14	((s. role == 'Student' AND s.matriculation == param.matriculation) OR (s.role == 'Counselor'))
…	…
19	TRUE
…	…
165	((s. role == 'Student' AND s.matriculation == param.matriculation) OR (s.role == 'Counselor')) AND TRUE
…	…

Table 1. Executable Policies

Table 1 shows an extract of our policy store. Three policies are needed for the generation of the ToR: object id 14 is the basic web service operation retrieving exam results for a defined matriculation number. Object id 19 corresponds to the web service operation obtaining lecture information – this is allowed for everyone which results in a policy directly relating to TRUE. Object 165 is the BPEL process which is assigned (at least) the same policies of the invoked operations. Additionally, further constraints can be added. We used a MySQL database to store the policies and a Java-based PDP (stateless session bean running on JBoss AS) for the calculation.

B. Performance Evaluation

For performance evaluation of our approach we compare the BPEL-based ToR generation with and without authorization verification. For automated testing we used Parasoft's SOAtest™ [25]. We put the BPEL-Process ToR on one machine running Windows 2003 Server with Oracle BPEL Process Manager 10.1.3, the two basic web services wrapping the legacy systems were put on two different machines both running SuSE Linux 10.1 and JBoss Application Server 4.0.5.

Measured Object	*Min*	*Avg*	*Max*
BPEL-Process ToR (w/o AuthZ)	391	860	2644
Basic Web Service 1 (w/o AuthZ)	125	152	921
Basic Web Service 2 (w/o AuthZ)	31	47	797
BPEL-Process ToR (with AuthZ)	553	1026	2838
Basic Web Service 1 (with AuthZ)	172	203	1207
Basic Web Service 2 (with AuthZ)	73	92	810

Table 2. Performance Measurement (in Milliseconds)

We measured two different scenarios, each of them having three targets. The first scenario was without authorization verification. The second was done with authorization verification thereby illustrating how time consuming this can be. The three targets are the complete BPEL process (including the basic web service calls) and two basic web services individually. We measured the runtime from the service invocation until the response message was returned.

We calculated that most of the values are close to the average, the standard deviation was minimal in all six cases. The few outliners result in higher network traffic. That is why we concentrate on the average values. Web service 1 is connected to an SAP R/3 system which resides on another system whilst web service 2 is directly connected to the database of the HIS system. This is the reason for the difference in average execution times of approximately 100 ms. The performance measurement reveals that the additional call to the authorization verification service costs an average of 50 ms per call. For the entire BPEL process this results in 150 ms as there are three calls altogether – which is a surplus in processing time of 20%.

VI. Conclusion and Future Work

In this paper we presented an approach how to handle authorization verification in web service-oriented architecture (WSOA). Therefore we designed an access control metamodel which enhances existing models to better suit the requirements of WSOA. We used this access control metamodel for the development of an authorization verification service which is part of the identity management architecture. We illustrated how it is linked with WSOA's core concern part and demonstrated the feasibility of our approach in a case study.

Our next steps concerning identity management for WSOA are to consider a conjoint and model-driven development of WSOA's core concern services with their associated access policies. Starting from computational independent models at the business process level, they can be derived to platform independent models and transformed to platform specific models (i.e. IdM architecture-specific) which are effective calculable policies. To enable interoperability, we will attempt the alignment of our approach with OASIS's eXtensible Access Control Markup Language (XACML) [26].

References

[1] Christian Emig, Jochen Weisser, Sebastian Abeck: Development of SOA-Based Software Systems – an Evolutionary Programming Approach, IEEE Conference on Internet and Web Applications and Services ICIW'06, Guadeloupe / French Caribbean, February 2006.

[2] Christian Emig, Christof Momm, Jochen Weisser, Sebastian Abeck: Programming in the Large based on the Business Process Modeling Notation, Lecture Notes in Informatics (LNI) 68 GI 2005, pp. 627-631, September 2005.

[3] T. Elrad, R. E. Filman, A. Bader, Guest Editors: Aspect-Oriented Programming. In: Communications of the ACM. Oktober 2001, Vol. 44, No. 10, 29–32.

[4] Mike Neuenschwander: Enterprise Identity Management Market 2006–2007, Burton Group Identity and Privacy Strategies, November 2006.

[5] D. F. Ferraiolo, Ravi Sandhu, Serban Gavrila, D. R. Kuhn, Ramaswamy Chandramouli: Proposed NIST standard for role-based access control, ACM Transactions on Information and System Security (TISSEC), Volume 4 , Issue 3, p. 224 – 274, August 2001.

[6] Eric Yuan, Jin Tong: Attribute Based Access Control (ABAC) for Web Services, IEEE International Conference on Web Services (ICWS 2005), Orlando Florida, July 2005.

[7] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides: Design Patterns, Addison Wesley, 1998.

[8] Frank Leymann: Web Services - Distributed Applications without Limits, Business, Technology and Web, Leipzig, 2003.

[9] Ali Arsanjani: Service-Oriented Modeling and Architecture, IBM developer works, 2004.

[10] Eric Newcomer, Greg Lomow: Understanding SOA with Web Services, Addison Wesley Professional, December, 2004

[11] Object Management Group (OMG): The OMG and Service Oriented Architecture, 2006.
http://www.omg.org/attachments/pdf/OMG-and-the-SOA.pdf

[12] Bernhard Humm, Markus Voss, Andreas Hess: Regeln für serviceorientierte Architekturen in hoher Qualität, Informatik Spektrum, Volume 29, Number 6 / December, 2006.

[13] Burton Group: Directory Landscape – Directory Products evolve towards Identity Services, Version 1.0, November 2004

[14] Edsger Dijkstra. On the role of scientific thought. EWD 447, 30th August 1974, Neuen, The Netherlands. Appears in: Edsger W. Dijkstra, Selected Writings on Computing: A Personal Perspective, Springer-Verlag, 1982. ISBN 0–387–90652–5, pp. 60–66.

[15] B.W. Lampson: Dynamic protection structures, AFIPS conference proceedings, FJCC 1969, p27-38.

[16] Roger R. Schell, Peter J. Downey, Gerald J. Popek: Preliminary Notes on the Design of Secure Military Computer Systems, MCI-73-1, Electronic Systems Division, Hanscom AFB, Bedford, Massachusetts, January 1973.

[17] D. Elliott Bell, Leonard J. La Padula: Secure Computer Systems – A Mathematical Model, ESD-TR-73-278, Vol. II, AD 771 543, Electronic Systems Division, Air Force Systems Command, Hanscom AFB, Bedford, Massachusetts, November 1973.

[18] Ravi S. Sandhu: Lattice-Based Access Control Models, IEEE Computer, November 1993, p. 9-19.

[19] D. F. Ferraiolo, D. R. Kuhn: Role Based Access Control, 15th National Computer Security Conference, 1992.

[20] D. F. Ferraiolo, J. Cugini, D. R. Kuhn: Role Based Access Control: Features and Motivations, Computer Security Applications Conference, 1995.

[21] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, C. E. Youman Role-Based: Access Control Models, IEEE Computer 29(2): 38-47, IEEE Press, 1996.

[22] Michael Hafner, Ruth Breu: Realizing Model Driven Security for Inter-organizational Workflows with WS-CDL and UML 2.0, MoDELS 2005, LNCS 3713, pp. 39-53, 2005.

[23] W3C: Web Services Description Language (WSDL), version 1.1, March 2001.
http://www.w3.org/TR/wsdl

[24] Christian Emig, Heiko Schandua, Sebastian Abeck: SOA-aware Authorization Control, International Conference Software Engineering Advances ICSEA'06, Tahiti / French Polynesia, November 2006.

[25] Parasoft SOAtest™, Product Homepage.
http://www.parasoft.com/jsp/products/home.jsp?product=SOAP

[26] OASIS eXtensible Access Control Markup Language (XACML) 2.0
http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml

All web references were verified on May 30th, 2007.