APPLICATION ACCESS CONTROL USING ENTERPRISE MODELS

Dietz, Gunnar, Universität Hamburg, Schlüterstr. 70, 20146 Hamburg, Germany, gunnar.dietz@uni-hamburg.de
Weller, Jens, Technische Universität Dresden, Schumann Building, Münchner Platz, 01062 Dresden, Germany, jens.weller@tu-dresden.de
Juhrisch, Martin, Westfälische Wilhelms-Universität Münster, Röntgenstr. 9-13, 48149 Münster, Germany, martin.juhrisch@uni-muenster.de

Abstract

In this paper a framework for a model-driven control of identity management systems is presented. An important issue in today’s information systems security discussion addresses the effective authorisation of users. With established conceptual modelling languages the assignment of roles to the identity management software is an enormous organisational effort. To decrease administration costs we propose a direct connection between the identity management system and enterprise models which contain the organisational responsibilities. Therefore, we have created the modelling approach E³+WS available for the meta-CASE tool cubetto toolset and the Novell Identity Manager.

Keywords: Enterprise Models, Business IT-Alignment, Identity Management
1 INTRODUCTION

Many German universities are currently implementing or planning to implement identity management systems (IDMS). In Hamburg, Germany, six universities of different size have joint their efforts in a project called eCampus, which is currently in a second phase (eCampus II) which plans to implement the approaches planned in the first phase of the project.

Identity management systems are responsible to consolidate and synchronize many user repositories, some of them serving as a source of person data, some of them mainly as targets of either person data (e.g. white pages) or account data. The latter case is the so-called provisioning, which is responsible for the creation, modification or deletion of accounts on target systems and to equip the accounts with the correct rights the account owner needs.

To achieve this, mostly the RBAC approach (Role Based Access Control) is used (Ferraiolo and Kuhn, 1992). Roles describe the functions of a person within an organization and are used to determine the rights that person needs in IT-systems. Roles can either be derived from the person data coming from the source systems or can be assigned manually. For the latter, identity management systems normally support workflow systems that are used to entitle persons for specific rights (or to revoke them). Rules and policies are used to provision accounts in a role based way.

Thus, an identity management system will allow a better management of processes that manage accesses to several IT-systems, support the automation of inter-department workflows, and avoid concurrent administrative operations. Furthermore the transparency of the person data flow is greatly enhanced which leads to a better security and data protection.

The software used in this project is the Novell Identity Manager, formerly known as DirXML. The Novell software supports so-called Role Based Entitlements (RBE) which assign dynamically and automatically entitlements based on the roles (or more generally some conditions of the attributes) of a person. The entitlements moreover control access to systems, assignment to groups (and therefore the assignment of some rights) or other things. All this can be controlled very flexibly by rules and policies.

However, while the IDMS software brings the framework for all this, the question remains how to implement the IDMS. This involves firstly the question how to determine a useful set of roles and how to assign them correctly to the persons. Secondly, how to implement the rules and policies for the provisioning such that each person gets the correct rights on each target system? These questions are central in each identity management project and involve a great complexity, especially when the number of roles a person can have is very high, which is true in the case of universities. A deep understanding of the organizational structure is needed and a detailed process analysis.

Due to the raising usage of modelling tools, model data is often already electronically available and could be used in an automated way to configure business software like an identity management system. In this paper we suggest an automated binding of identity management functionality to enterprise models. In this scenario the IDMS directly accesses the enterprise models to adopt its behaviour according to the described organizational responsibilities. The SOA paradigm (Erl, 2005; Stojanovic and Dahanayake, 2005) offers a solid basis for the realization of such a connection, where we focus on the W3C’s Web service standards.

To be able to use model data automatically by Web services accessing the modelling tool, existing enterprise models have to be enriched by such data (e.g. role data for each process) and corresponding Web services have to be instantiated. To realize a solution for enterprise models in different modelling languages a grammar to model such Web services on a meta-model layer is presented here. This allows the instantiation of Web services accessing models using this meta-level description. While this approach is usage in general scenarios we focus on the configuration of identity management systems.

The paper is structured as follows. In the next section we give a brief theoretical introduction into the specifics of enterprise models and service-oriented architectures (SOA). Section 3 introduces a framework showing how enterprise and Web service models can control university’s systems functionalities. Section 4 introduces shortly the university research project and describes how a first
Implementation was realised within the project. The paper ends with a discussion, summarising the proposed ideas and exposing open questions regarding the realization of the application integration.

2 THEORETICAL BACKGROUND

2.1 Conceptual Modelling

A model is understood as the result of a construction process "... done by a modeller, who examines the elements of a system for a specific purpose ..." (Schütte and Rotthowe, 1998), which is defined by the user of the model. Depending on the kind of system that is examined during the modelling process (the problem domain), models in the information systems field can be classified as design models or conceptual models. While design models represent software systems or parts of it, conceptual models describe real world phenomena (Dieste, Juristo, Moreno, Pazos and Sierra, 2000; Evermann and Wand, 2005; Wand and Weber, 2002). For further scientific analysis, we will use enterprise models as synonyms to conceptual models. The language used to express the model is called the modelling language. Different modelling languages are currently used, which comply with different problem domains as well as with different modelling purposes (Brinkkemper, Saeki and Harmsen, 1998). Lastly, models are valid for a certain time interval leading to constant modifications of these models during their lifetime.

In the information systems discipline, models are used to redesign organizations or to develop software systems (Rosemann, Schwegmann and Delfmann, 2005). Therefore, graphical models have been established as "...a medium to foster communication with prospective users..." (Frank, 1999). For reasons of simplicity, complex models often consist of several views showing just a section of the whole model (Schéer, 2000). The different views, however, are not separated but integrated using the same model elements within different views (Schütte and Rotthowe, 1998). Thus, we can argue that a model always consists of its content and one or more graphical representation, each of which illustrates a set of the model’s elements and their relationships. Considering this aspect, modelling languages comprise the definition of a graphical representation (concrete syntax), the context (abstract syntax) and the meaning of the constructs (semantics) defined within the language (Clark, Evans and Kent, 2002).

To assist the modelling process, methods have been established within the IS discipline. IS methods consist of a specification of activities describing the steps of how to build up a model, and a description of the modelling language used for modelling (Harmsen, Brinkkemper and Oei, 1994; Kelly, Rossi and Tolvanen, 2005; Wand and Weber, 2002). The description of the modelling language is also called product description; the activity part is called process description (Brinkkemper, 1996).

Using a given method is sometimes difficult when creating models of a specific domain the language was not explicitly made for (Brinkkemper et al., 1998). In this case, modellers might want to add new concepts or rules that are not covered by the original modelling language. The situational method engineering approach addresses that problem (Harmsen et al., 1994; Kelly et al., 2005). Following that approach, methods used within a modelling project are adapted permanently according to the needs of the modellers. An overview about the different types of (situational) method engineering is given in (Ralyté, Rolland and Deneckère, 2004). To support the modelling process with different, individual methods, meta-CASE tools, like MetaEdit+ (Kelly et al., 2005) or cubetto toolset (Cubetto, 2008), have been established. Those tools enable the user to define and modify methods that will be used within the modelling project. Hence, generic meta-CASE tools offer modelling support in various problem domains. As generic method engineering tools, they provide a meta-modelling language to extend or to simplify modelling languages as well as to create new ones.

2.2 Service-oriented Architecture

A Service-oriented architecture (SOA) is a method to design system landscapes and it can be understood as a part of a company's Enterprise Architecture (EA) (Barry, 2003). Grounded on the concepts of contract, service, and interface, the SOA paradigm aims the service relation to a semi- or fully automated activity in business processes. This happens following the contract terms in which the characteristics of the activity’s implementation are defined (Dietzsch and Goetz, 2005). The service
functions – meaning differentiated and autonomously working functions of a service also usable by other services – are utilised by the interface of an application.

A difference between the SOA paradigm and other software paradigm is its service function concept that represents an entire business activity and is reused on system landscape level. This occurs when an SOA reaches a development phase where the prerequisites for a reuse of software solutions in a company are fulfilled. In this case there is an opportunity for multi-reuse of business solutions including their implementation in software as a service (Dietzsch and Goetz, 2005).

This paper focuses on a Web service based implementation of the SOA paradigm following the ideas of the W3C consortium (W3C, 2003). W3C’s layered architecture for today’s Web service technology focuses on three core elements. The Simple Object Access Protocol (SOAP), the Web Service Description Language (WSDL) as well as the Universal Description, Discovery and Integration Language (UDDI) (Walsh, 2002) have become de facto standards for XML messaging, Web service description and registration. Additionally composition of Web services requires higher describing levels. Orchestrations or choreography languages base on several advanced standards like WS-BPEL or WSFL (Web-service Flow Language) (IBM, 2001). By comparison to the core protocols these high level languages take a step forward towards integrating Web services in business process models. The main idea is to combine business processes across enterprise boundaries by modelling Web services in directed graphs after their chronological sequence.

Hence, business process alignment is currently reached using attached workflow management systems (WMMS) operating on BPEL models. These BPEL models integrate services via complex data interchanges (WS-BPEL; Andrews et al., 2003). The prerequisite for BPEL-support is the ex ante modelling of all executable business processes by a central modeller (Krcmar, Schwarzer and Zerbe, 1997). For a raised flexibility of BPEL a huge number of variations on business processes must be modelled. This means a dropping number of executions per process definition that is unprofitable when using this extensive modelling technique. Functional adaptation is usually carried out by a complete new modelling and implementation. While Enterprise Architecture (EA) frameworks can help understanding and managing the different layers, they generally do not solve problems arising if one view (or architectural layer) alters, too. In this case, the other views have to be adapted according to the dependencies between them.

The recommended approach introduces a way for tied integration of enterprise models using the common EA views: data, function, executable processes and protocols. This means tailoring the views which support the enterprise application if an enterprise model (e.g. process model) changes. Our approach is based on the concept of SOA but enlarges the domain of integration using modelling CASE tools. With the inclusion of enterprise model information the integration paradigm gains an additional aspect.

3 MODEL-DRIVEN ACCESS CONTROL

The idea to integrate modelling tools as Web services in a service-oriented architecture comes along with a growing demand for management methods helping organisations to map business requirements documented in enterprise models onto service compositions most likely to BPEL (Bloomberg, 2002).

Referring to the model-driven architecture’s (MDA) levels of abstraction (MDA, 2003) this mapping is related to the transformation between computation independent models (CIM) representing domain specific information, independent from implementation technologies and design languages (platform independent models, PIM) describing business logic combined with problem solutions in terms of software systems. A well-known approach for model driven service composition is the transformation of Business Process Modeling Notation (BPMLN) models into BPEL XML graphs (Ouyang, van der Aalst, Dumas and ter Hofstede, 2006) (see Figure 1). WS-BPEL process definitions are used to map business logic to an underlying Web service layer (invoked applications).
In case the models created with BPMN can be classified as CIM, we have doubts about an automatic transformation process. Recent research shows that CIM in their role as problem description are the input for a creative, manual decision process that produces a solution in terms of a software design (Génova, Valiente and Nubiola, 2005; Karow and Gehlert, 2006). Thus, this way of integration requires manual adaptation. This results in a static separation between business logic from the underlying application architecture. Potentially this leads to a problem, as the business architecture and its application architecture evolve independently of each other.

Thus, the authors of this paper accept this task as a creative task done by humans and focus only on the control of workflows in addressable application components that are dependent on a certain business context. The context of a process flow is understood as the information passed between business process activities in an enterprise model together with the information provided by the modeller of the business process (Leymann, Roller and Schmidt, 2002). Although conventional conceptual modelling languages are able to express requirements and behavioural models of target systems, no existing language supports the integration and sharing of model information with IT applications through Web services of enterprise models. We achieve this transparency by the extension of conceptual modelling languages, e.g. ARIS (Scheer, 2000) with language constructs that aim at the description of model information as Web services.

The property of Web services to be easily specifiable with design languages leads to an evaluation of mapping possibilities from formal concepts of Web service design to the meta-level of conceptual modelling. In consideration of the outlined deficits we propose a semantically enriched meta-model for conceptual modelling. The genericity of these Web services enables a flexible integration between infrastructure and enterprise models. This requires additionally to the description of system requirements during the modelling of business applications a model of Web services describing the services which are used by the applications.

To summarise, the automatic adaptation of application system functionality using our integration approach is the main difference to previous workflow modelling languages within the software engineering discipline (Busi, Gorrieri, Guidi, Lucchi and Zavattaro, 2005). However, in contrast to the composition of Web services to executable business processes, our approach is limited to a certain amount of addressable application systems.

3.1 Service Modelling Language

In order to achieve a Web service modelling independent of a language an existing method of the method-engineering discipline was used and improved. During this process, elements related to the
Web services are integrated into a meta-modelling language. In this section, the $E^3$+WS method is introduced that enables a description of a Web service basing on meta-model information as well as a model-driven realisation of Web service functionality. This method engineering predicated on the existing $E^3$-Model ($E^3$; Greiffenberg, 2004), classified as meta-meta-model on M3-Level of the Meta-Object-Facility Architecture (OMG, 2002).

The Web service remains as the essential component of our modelling approach. A Web service is developed in two ways: the functionality and the (unique) name are defined in the abstract definition. Functionality is then connected to Web services via aggregation of service functions whereas the tasks of the designator are to identify and declare the semantics of the Web service. Elements of Web services can be composed out of service functions with the help of edge bundling. Messages are connected to service functions via so-called messageEdge_in or messageEdge_out components depending on the message role regarding to the related service functions. Thereby a service function can be connected to maximum one messageEdge_in and always exactly to one messageEdge_out (see Figure 2). Hence, it is assumed that the possible service functions communicate either by a request response or with a notification pattern (Walsh, 2002). Thus, Web services correspond to a number of similar service functions. Furthermore, a service function includes a suitable parameter assignment. Therefore, we define special message objects that have a task of input containers for corresponding output parameter objects.

In order to define a single message parameter a construct parameter object is introduced that is dividable into simple or complex objects. Simple parameter objects are allocated to simple data types that on the one hand encompass predefined XML schema data types and on the other hand own data types defined in the WSDL tag types. For instance, with self-declared data types we can define suitable report formats as potential result layouts for a model migration operation. A complex parameter object corresponds to design activities of a method engineer. Therefore, an $E^3$-data type is introduced to establish ties between complex parameter objects and meta-model patterns. At first we only tie $E^3$ object types down to complex parameter objects. Message element and parameter object are analogical to service function and to Web service element. For more detailed information of the $E^3$+WS modelling notation we reference the work of Weller et al. (2006).

Figure 2. Service modelling language

Abstracting from the established notation, the first step is a development of a language based meta-model, in our case including all actions to model a meta-model, a Web service interface and its implementation. Sharing the $E^3$+WS meta-model in different views we can simplify the development
process. First, we assume that all the elements of the meta-model we want to extend and relationships between these elements are described in the $E^3$-Model language standard defined in an $eE^3$-View (extended $E^3$-View). Next we need to generate Web service descriptions in a separate Web service view. The engendered modelling language is able to express WSDL according to the W3C standard and can be recognized and adopted by target users of the $E^3$+WS method (see again Figure 2). A key requirement of this framework is to avoid the restrictions of the regular modelling task on object level. Both domain (clients) and modelling experts (model creator) should not notice that their enterprise model input is visible. Usually the development of enterprise models requires “a specific competence that is neither covered by software engineering nor by organization science” (Frank, 1999), so implementation related information have to be kept away.

4 APPLYING THE MODELLING APPROACH

4.1 Project Aim Definition

Universities are currently under the pressure of multilayered changes and the increasing complexity of IT-based services and infrastructure. The Bologna process (European project for creating academic degree standards) and of modern e-government must be taken into consideration. In order to cope with the resulting requirements the universities the six different universities of Hamburg, i.e. Universität Hamburg (UHH); Technische Universität Hamburg (TU); Hochschule für angewandte Wissenschaften (HAW); Hafen City Universität (HCU); Hochschule für Musik und Theater (HfMT), and Hochschule für Bildende Künste (HfBK) have launched together with the Multimedia Kontor Hamburg (MMKH) the project eCampus. In the first phase eCampus I the topics “base services” (mainly authentication, authorization and accounting), “student selection and management”, “course and exam management” and “inter-university web presence as a basis for a science portal” were discussed. Two central parts of the current phase eCampus II, which runs from 2006 to the end of 2008, is the joint implementation of campus management systems and identity management systems as a continuation of the approaches discussed in eCampus I.

The joint approach to identity management plans a cascading set of metadirectory implementations: one central metadirectory responsible for the consolidation of user data in an inter-university way and for the support of university-overlapping services; furthermore local metadirectories at least at each of the three bigger universities (the local services) that are responsible for the provisioning of internal university systems and to support internal services. The local services will be provisioned by the eCampus-IDMS (see Figure 4).

4.2 Implementation

To come to a (prototypical) implementation of the presented theoretical approach, three things had to been taken into consideration: The extension of the modelling language to hold role data, the instantiation of a Web service to access the modelling data stored in cubetto and the implementation of a DirXML driver on the Novell side to push this information into the IDMS. Using the meta-CASE tool cubetto toolset (Weller et al., 2006) we extended the modelling language ARIS using as described $E^3$+WS to be able to hold role information in EPC (Event-driven Process Chain) diagrams. For each process within such an EPC diagram one or multiple roles that are needed to fulfill this process task (i.e. to get the necessary access rights to the corresponding (Web) service) will be attached. A Web service has been instantiated to access the modelling data stored in cubetto. This Web service can be used to monitor changes of the modelling data. For example, in Figure 3 a change in role assignment at the exemplarily process “examination as to formal requirements” is shown. Such model change could be motivated by the fact that some tasks would have been delegated e.g. by the head of the department to his secretary. In agreement with a domain expert the model change would have been done by a model creator.
To provide the IDMS with the needed information stored in the models, a so called (DirXML-)driver has been implemented that is able to access the cubetto Web service. Figure 4 shows the design of the identity management implementation within the project eCampus II and the integration of the cubetto driver as part of this implementation. A change of the model data (in the sense of Figure 3) within cubetto is forwarded by the driver to the identity vault (directory service). As shown in detail in Figure 5 this results in a directory of all processes and roles and their relations within the identity vault. Moreover the cubetto driver is able to update the corresponding Role Based Entitlements (RBE). All these propagated changes within the identity vault generate events within the Novell Identity Manager engine, which can be used by rules and policies within the cubetto driver or other drivers to provision the target systems and to grant (or revoke) rights for the Web services in question.
Figure 4. Design of the eCampus II IDMS and integration of cubetto
5 CONCLUSIONS AND FURTHER RESEARCH

With the prototypical implementation of the presented approach of this paper to use enterprise models to configure in an automated way identity management systems part of a more general vision has been brought to life. More generally, the reuse of enterprise models by using them to manage arbitrary application architectures will come up to a major challenge to increasing the use of enterprise analysis methods in businesses and organizations (Tissot and Crump, 1998). The presented E³+WS method could be used in a more general context. The possibility of the automated creation of Web service descriptions for accessing model information stored within the modelling tool cubetto toolset to configure adaptive software applications within an EAI environment will help to extend enterprise model utilization over its original domain of documentation to integration purposes with other software systems (Weller et al., 2006).

The usage of an extended ARIS meta model allows standard enterprise modelling using the ARIS notation. The E³+WS approach allows furthermore to describe Web services to access the (enriched) modelling data and to instantiate these Web services in an automated way.

We were able to test the presented approach within the described identity management scenario in a prototypical way. The cubetto toolset and the Novell identity manager software have built a solid base for such an implementation. Further detailed case studies will follow to empirically prove its usefulness within the eCampus II project. Some parts of the software implementation are still in progress and have to be finalized to be used in a real scenario. This applies to the automated Web service instantiation of the model server (until now the WSDL files can be generated automatically but
the Web service implementation is still manual) and also the DirXML driver for accessing these Web services.

Moreover, further discussion on what information should be added to the process models and what information should remain in the IDMS is needed. Leaving the identity management scenario this question has to be discussed in general: Which information can be usefully stored within enterprise models for the automated configuration of a SOA?

References

Bloomberg, J. (2002) Web Services Management: The Key to Service-Oriented Architectures,
ZapThink report.
Standardsoftware und zur Unterstützung flexibler Workflows., Chair for Business Informatics, Technical University of Munich, Munich, Germany.


