Web Service Engineering – Advancing a New Software Engineering Discipline

Ruth Breu[1], Michael Breu[1], Michael Hafner[1], and Andrea Nowak[2]

[1] Universität Innsbruck, Institut für Informatik, Techniker Straße 21a, A – 6020 Innsbruck
{ruth.breu,michael.breu,m.hafner}@uibk.ac.at
[2] ARC Seibersdorf Research, Kramergasse 1, A–1010 Wien
andrea.nowak@arcs.ac.at

Abstract. In this paper we present SECTET, a tool-based framework for the design, implementation and quality assurance of web service based applications. Main focus in SECTET is put on the design of inter-organizational workflows, the model driven realization of security aspects and testing of workflows. We present an overview of the model views, the design activities and the underlying architecture.

1 Introduction

Component-based software development has been one of the hot topics in software engineering for at least the last decade. The idea of constructing IT systems in the same modular way as cars or washing machines is appealing and has led many people to think about new markets and business models for software components.

While platform dependence was a great obstacle for bringing such scenarios into practice some years ago, web service technology now opens a plethora of new possibilities ranging from the realization of inter-organizational workflows, new flexible ways of cooperation between business partners and virtual web service market places. Not every today´s vision will find its way into practice, but in any case composing web services to new applications is an upcoming important paradigm of software development.

What we will address in this paper is the question how techniques and methods of software engineering apply to this new style of programming. More precisely, we will focus on *modelling* and *testing* web service based systems.

In a web service based application there are always at least two types of stakeholders – the supplier of some service and the client using the service. In this paper we will only take the client view and assume that the web services to be used are already given. We will not deal with the steps to deploy some web service and the steps to find web services of interest.

In the subsequent sections we will present the tool-based method SECTET for web service engineering. SECTET is developed by our research group Quality Engineering in a cluster of cooperation projects with our project partners ARC Seibersdorf research (project SECTINO) and world-direct/Telekom Austria (project FLOWTEST).

SECTET is targeted towards the high-level development of inter-organizational workflows based on web service technology. From the technological side the basic constituents of our framework are the atomic web services and a web service orchestration language like BPEL4WS [8] together with the related tools [9]. From the

D. Lowe and M. Gaedke (Eds.): ICWE 2005, LNCS 3579, pp. 8–18, 2005.
© Springer-Verlag Berlin Heidelberg 2005

methodological side we deal with aspects how to specify the interface of a web service, how to design an inter-organizational workflow step by step and how to test such an application.

We put a special focus on the aspect of security playing a crucial role in most inter-organizational workflows. Our goal is to assess security requirements at a high level of abstraction and to provide pattern-based solutions. Due to the background of our project partners we concentrate on applications in e-government and e-business, though the general approach is application-independent.

The backbone of our method are UML models. We use class diagrams and the predicative language OCL [19] for describing (XML-)data and interfaces, and activity diagrams for describing workflows. We use these models from two perspectives. The first is the requirements specification perspective supporting a step by step development of inter-organizational systems. In the second perspective we pursue a model-driven approach in which code is generated based on specific models.

Since all standards and languages in web services technology are based on low-level XML structures in our conviction such a model-driven approach is of primary choice to achieve an adequate level of abstraction for the development of web services based applications.

Our approach is novel in many respects. We contribute substantially to requirements specification, model-based specification of security requirements and testing of inter-organizational workflows in the context of web services. Related approaches which however focus on different technologies can be found in the areas of workflow management (e.g. [3, 10, 11, 13]), authorization models (e.g. [5, 15, 22]) and testing [21]. To our knowledge the term *Web Service Engineering* has been first used by Starke [20], however in a very unspecific way.

In the sequel we present an overview of the core concepts of our framework. In the SECTINO project we have developed a set of basic models and views of an inter-organizational application. We distinguish two basic classifications, the component vs. workflow view and the global vs. local view which are presented in section 2. Section 3 deals with the aspects of model-driven software development in SECTINO, while Section 4 is devoted to the step by step development of inter-organizational systems. Section 5 sketches the requirements to a testing environment for inter-organizational workflows and, finally, we draw some conclusions in Section 6.

For a more detailed presentation of single aspects of our approach we refer to a series of accompanying papers, in particular [2, 6, 12].

2 Views and Models

We conceive an inter-organizational application as a network of partners communicating by calling (web) services and exchanging (XML) data. Within the design of such an application we distinguish two orthogonal classifications and views of the system: the *global* or *local view* on the one hand side and the *component* or *workflow view* on the other side.

The *global view* conceives the inter-organizational system as a whole, the *local view* focuses on the behavior and structure of one partner within the network.

In the *component view* each partner is conceived as a node offering a set of services with given properties. The component view is independent of the context in which the

services are used. In the *workflow view* the orchestration of the partners´ services is defined.

Within this classification schema we distinguish the *interface model* (local component view), the *global workflow model* describing the business protocol of the cooperating partners, and the *local workflow model* describing the behaviour of each partner node (cf. Fig. 1).

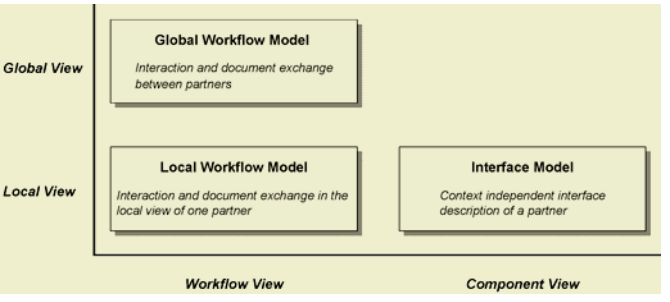


Fig. 1. Models and Views

This orthogonal perspective allows us to combine the design of components offering services that different types of partners may call in different contexts with the design of workflows that focus on particular usage scenarios. In many applications the component view of (some or all) partners is already given when the inter-organizational workflow is developed.

As running example we will use the interaction between a business agent (the Tax Advisor) and a public service provider (the Municipality) for submitting and processing annual statements concerning the municipal tax of companies. Table 1 shows a portion of the informal textual description of the workflow.

2.1 Interface Model

The interface model describes the set of services a partner node offers to the outside. The interface model consists of the following submodels.

- The *document model* is a UML class diagram describing the data type view of the partner. We talk of documents in order to stress that we do not interpret this class diagram in the usual object oriented setting but in the context of XML schema [23].
- The *interface* contains a set of abstract (UML-)operations representing services the component offers to its clients. The types of the parameters are either basic types or classes in the document model. Additionally, pre- and postconditions (in OCL style) specify the behavior of the abstract services.
- The *role model* describes the roles having access to the services. An example of a role within the municipality component is the tax advisor (e.g. calling the web service sendProcessedAnnualStatement of the municipality).
- The *access model* describes the conditions under which a certain role has the permission to call a given service. We use the predicative approach of [7] to specify the access model. This approach uses an OCL dialect to specify conditions under

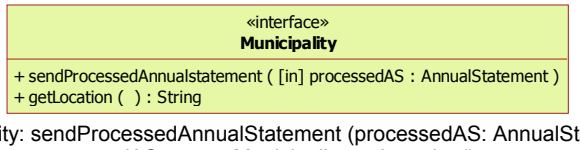
which a given role has the permission to call an operation. This permission may depend on the parameters, the state of the calling subject (e.g. its geographical location) and on the internal representation of the subject.

Table 1. Informal Description of the *Workflow Process Annual Statement*

Workflow Process Annual Statement
Partners Tax Advisor, Municipality
Main Steps 1. The Tax Advisor prepares the annual statement of his client 2. The Tax Advisor sends the annual statement to the Municipality. 3. The Municipality checks the incoming statement for validity. 4. The Municipality stipulates the communal taxes based on the incoming annual statement and the received payments during the year and prepares the notification. 5. The Municipality sends the notification back to the Tax Advisor. 6. The Tax Advisor receives the notification and checks it according to expected results.
Variants ...
Security Requirements 2. The annual statement is confidential and has to be signed by the tax advisor. 5. The notification is confidential and is signed by the Municipality. 2./5. The reception of the annual statement and the notification have to be non-repudiable.

Example

The tax advisor has the right to call the web service sendProcessedAnnualStatement of the municipality component if the town of the applicant´s annual statement is the same as the town of the municipality. We assume that **AnnualStatement** is a class in the document model describing the structure of annual statements; one of the attributes of **AnnualStatement** is town. Moreover, getLocation() is a service that returns the town of the municipality.



context Municipality: sendProcessedAnnualStatement (processedAS: AnnualStatement)
perm[tax advisor]: processedAS.town = Municipality.getLocation()

2.2 Global Workflow Model

The *global workflow model* describes an abstract view of the business protocol between partners in autonomous organizations. The global workflow is abstract in the sense that it describes the interaction of partners at a level that contains neither internal steps nor the connection to the business logic. The global workflow model consists of the following submodels.

- The *global workflow* is described by a UML activity diagram enhanced by security requirements concerning the communication between the partners The actions in this workflow diagram refer to the services offered by the respective partner.
- The *document model* and the *role model* describe the data exchanged by the partners in the workflow and the partner roles, respectively. Both models are class diagrams.

As an example, Fig. 2 depicts a portion of the global workflow between the Tax Advisor and the Municipality. The notes attached with the objects processedAS and notification, are security requirements explained in more detail in the subsequent section. In the complete model the workflow comprises additional partners (e.g. the health insurance for checking employees registered).

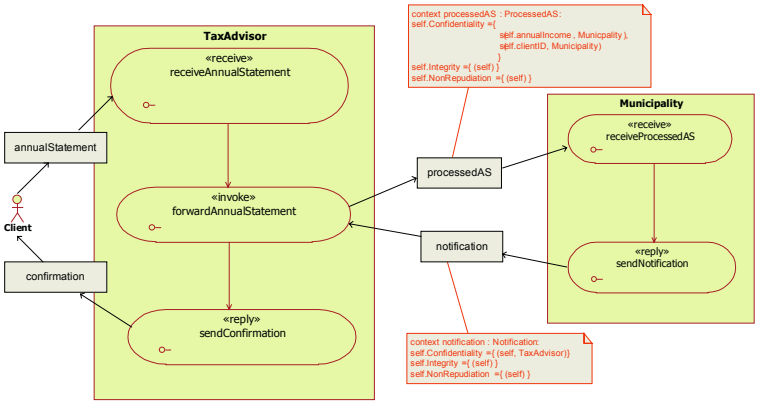


Fig. 2. Portion of the global workflow *Process Annual Statement*

The global workflow model is typically designed by the consortium of partners involved in the workflow. Many applications include strong legal requirements for the workflow or document model.

2.3 Local Workflow Model

A *local workflow model* is developed for each partner type. The local workflow defines the portion of the global workflow which each partner is responsible for and corresponds to the "Executable Process" in BPWL4WS1.1. The local workflow is a concrete process description. It does not only consider service calls from the outside but also contains internal actions and connections to the business logic. A complete local workflow model is direct input for a local workflow management system. The local models are typically developed by representatives of the partners involved.

Similarly to the global workflow model, the local workflow model consists of an activity diagram modeling the local workflow, document models and role models.

3 Model-Driven Development of Inter-organizational Applications

In our method we use models both for requirements elicitation and for code generation. The backbone of the inter-organizational application are the web services provided by the partners together with the local workflow engines controlling the workflow instances at each partner´s side[1].

[1] We conceive a centrally managed workflow as special case in which only one central partner is provided with such a workflow engine

The local workflow model of each partner type is the input to the local workflow engine. The target architecture does not provide an own workflow engine but uses a BPEL4WS-based workflow engine and a related UML front-end [16], other workflow engines and modelling front-ends are equally possible.

What SECTINO focuses on is the model-based generation of security components. While there are plenty of standards for web service security allowing security requirements like confidentiality and integrity to be implemented at XML and SOAP level (e.g. [18]), our claim is that a broad application of these standards requires a high-level development environment. With our approach we pursue model-based development of security components. The related core security architecture for each partner node is depicted in Fig. 3. This architecture wraps the basic web service components and the local workflow engine by a security gateway supporting the following services in the current version.

- Authentication of the requestor
- Decryption/encryption of messages
- Signing messages (with a system generated signature)
- Checking authorization of web service calls

More details about this reference architecture can be found in [12]. Fig. 4 illustrates the core inputs and outputs of the code generation.

- We specify security requirements concerning message exchange between partners in the global workflow model (cf. the notes in Fig. 2 requiring the confidentiality, integrity and non-repudiation of the documents exchanged) and generate the configuration files of the security gateway.
- The role and access model of the interface model are transformed into policies in the Policy Repository. The policy enforcement is based on the OASIS standard XACML [17].

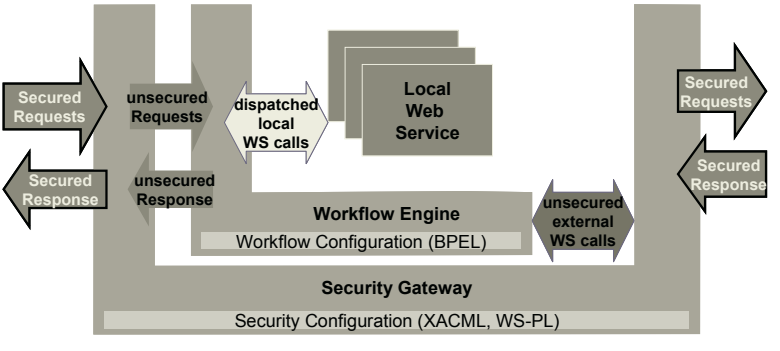


Fig. 3. SECTINO Schematic Reference Architecture

4 Requirements Elicitation

The models allowing code generation which have been discussed in the previous section are at a high though programmatic level of abstraction. Thus, our goal in SECTET is to integrate these models in a method that guides developers step by step to realize security-critical inter-organizational applications.

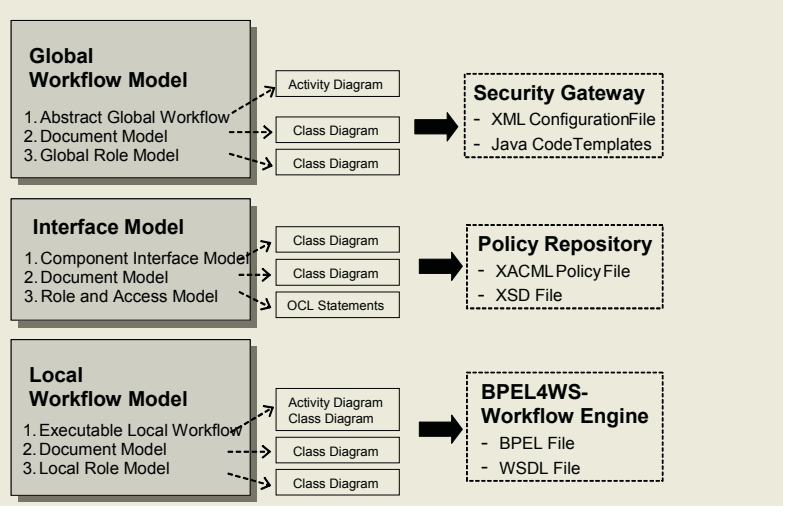


Fig. 4. Model-Driven Security in SECTET

For developing the local workflow models and the security-related annotations we proceed in the three basic steps of Fig. 5.

S1	Describe the business protocol informally in a global view
S2	Develop the global workflow model
S3	Develop the local workflow models for the partner nodes

Fig. 5. The three basic design steps of inter-organizational workflows

S1 – Describe the Workflow Informally in the Global View
Table 1 shows a small portion of such an informal description. We suggest a format close to an informal use case description [14] including the following parts.

- Name of the workflow
- Partner types
- Roles within the partner types (e.g. there may be roles *Secretary* and *Tax Advisor* within the partner type **Tax Advisor**)
- Main steps of the workflow together with variants and exceptions
- Security requirements on the workflow

The informal description is typically developed in cooperation with domain experts. This is particularly important for the security aspects which in many cases refer to legal requirements such as data protection or signature act. In non-standard cases a detailed threat and risk analysis of the security requirements is advisable (cf. [4]).

S2 – Develop the Global Workflow Model
The next step is to transform the textual description of the workflow into the formal representation of the global workflow model. Main activities within this step are the following.

- Identify the actions within the workflow taking into account the interface view of the partners and the primitives of the web service orchestration language (e.g. invoking web services, sending replies)
- Define the whole process including variants, loops and concurrently executed actions
- Model the security requirements in a formal way based on the annotation language
- Define the Document and the Role Model

It is important to note that not all informal security requirements can be transformed into formal ones in the model. Indeed, our language offers a set of patterns expressing standard security requirements. Security requirements that cannot be mapped onto the patterns have to be implemented programmatically in the security components.

S3 – Develop the Local Workflows for the Partner Nodes
In the last step the executable local workflows are developed. This comprises the following tasks.

- Divide the global workflow into local portions for each partner type
- Refine this local workflow based on the primitives of the chosen web service orchestration language by adding local actions and calls to the business logic
- Develop the document and the role model based on the respective models of the global workflow model

In general, the first task is non-straightforward and a problem that is subject of intensive research in a more formal setting [1, 3].

We follow a pragmatic approach and map the global workflow to a number of local workflow stubs. In this setting it is in the responsibility of the local partners to implement behavior that conforms to the global workflow. Rather than proving the correctness of the local executable workflows with respect to the abstract global workflow we provide a testing environment (cf. Section 5).

5 Testing Inter-organizational Workflows

Though developed at a high level of abstraction the resulting inter-organizational application in general will contain bugs. This both concerns the web service calls and the flow itself. In the cooperation project FLOWTEST we currently work on a model-based test environment for inter-organizational workflows.

Three major building blocks for such a test environment can be identified as follows.

T1 Local and global testing
T2 Specification of test conditions
T3 Test management and test data generation

Fig. 6. Basic building blocks of a test environment for inter-organizational workflows

T1 – Local and Global Testing
According to the model views and the executable artefacts we can distinguish three levels for testing inter-organizational applications. Each of these levels focuses on specific aspects of the execution.

- Testing the atomic web services – these tests are performed by the provider of the web service and the user of the web service. Client tests refer to the externally visible behaviour of the underlying business logic, while provider tests refer to the internal behaviour.
- Testing the local workflows – these tests are performed by each local partner. Local workflow tests focus on the orchestration of web services from the view of the partner. In order to support local testability the web services of the partner nodes (that may be called from the local flow) should be replaceable by local stubs.
- Testing the global workflow – these tests are performed by the responsible for the global workflow. Global workflow tests focus on the interplay of the local workflows. Similar to local workflow testing the availability of local stubs is a crucial requirement for global workflow testing.

T2 – Specification of Test Conditions

Our goal is to provide a test environment which works at the same level of abstraction as the design and implementation environment based on models. To this purpose we need a language for expressing test conditions.

Similarly to the specification of permissions we provide an OCL dialect for expressing properties over the document model. The test properties may be defined as conditions that have to be fulfilled independently of the input or as conditions that have to be fulfilled by specific test data.

As an example, Fig. 7 contains a simple postcondition requiring returned person data to fulfil an obvious constraint and a specific test condition for the test input string "`012345`".

```
getEnrollmentData (passportId: String): Person
post result.dateOfBirth.year <= 2005
```

```
sampleTestcond::
getEnrollmentData ("012345")
post result.name = "Erika Mustermann"
```

Fig. 7. Sample Test Conditions

Taking into account the test levels described in T1 the test conditions may be associated with the following model elements.

- Conditions on the execution of atomic web services
- Conditions on the execution of local workflows or on single steps within these workflows
- Conditions on the execution of global workflows or on single steps within these workflows

T3 – Test Management and Test Data Generation

The last aspect is concerned with the generation of (XML-)test data and the management of tests. Since the executable artefacts are programs including branches, loops and sequential and parallel composition well-known techniques of test data generation and test management can be applied at this place. This includes equivalence classes or random value tests.

6 Conclusion

In the preceding sections we have presented SECTET, a tool-based method for the development of security-critical inter-organizational applications. SECTET provides many novel aspects ranging from model-based development of security requirements to requirements elicitation and testing of inter-organizational workflows.

Future work has to be done in several directions. First, we will extend the set of supported security requirements. Primary candidates for such an extension are the support of qualified signatures and rights delegation. Second, we currently define the development method in more detail in the application area of e-government workflows. Moreover, the proposed requirements for the testing environment have to be elaborated and realized. A further aspect we will work out in detail is change management of the workflows.

First positive results in pilot applications with our industrial cooperation partners encourage us to further steps.

References

1. Van der Aalst, W.M.P. 2000. Loosely Coupled Interorganizational Workflows: Modeling and Analyzing Workflows Crossing Organizational Boundaries. In: Information and Management 37 (2000) 2, pp. 67-75.
2. M. Alam, R. Breu, M. Breu. Model-Driven Security for Web Services (MDS4WS), Proc. INMIC 2004, 2004.
3. W.M.P. van der Aalst and M. Weske. The P2P appraoch to Interorganizational Workflows. In: Proc. CAiSE 01, Springer Lecture Notes in Computer Science vol. 2068, pp. 140-156. Springer-Verlag, Berlin, 2001.
4. R. Breu, K. Burger, M. Hafner, G. Popp. Towards a Systematic Development of Secure Systems. Special Issue of the Information Systems Security Journal, Auerbach, 2004.
5. E. Bertino, S. Castano, E. Ferrari: Securing XML Documents with Author X. In: IEEE Internet Computing, vol. 5,no. 3, May/June 2001.
6. R. Breu, M. Hafner, B. Weber, A. Nowak: Model Driven Security for Inter-Organizational Workflows in e-Government. Accepted for TCGOV 2005.
7. R. Breu, G. Popp: Actor-Centric Modeling of User Rights. In: M.Wermelinger, T. Margaria-Steffen (Eds.): Proc. FASE 2004, Springer LNCS Vol. 2984,p. 165-179, 2004.
8. IBM, Microsoft, BEA Systems, SAP AG, Siebel Systems, "Specification: Business Process Execution Language for Web Services Version 1.1". See http://www-128.ibm.com/developerworks/library/ws-bpel/
9. IBM, "Business Process Execution Language for Web Services Java™ Run Time (BPWS4J)". See: http://www.alphaworks.ibm.com/tech/bpws4j
10. F. Casati and M. Shan. Event-based Interaction Management for Composite E-Services in eFlow. Information Systems Frontiers 4(2), 2002.
11. P. Grefen, K. Aberer, Y. Hoffner, H. Ludwig: CrossFlow: cross-organizational workflow management in dynamic virtual enterprises. In: International Journal of Computer Systems Science & Engineering 15 (2000) 5, pp. 277-290.
12. M. Hafner, R. Breu, M. Breu: A Security Architecure For Inter-organizational Workflows - Putting Web Service Security Standards Together. Accepted for ICEIS 2005.
13. W.K. Huang, V. Atluri. SecureFlow: A secure Web-enabled Workflow Management System. ACM Workshop on Role-Based Access Control, 1999.
14. I. Jacobson, G. Booch, J. Rumbaugh. The Unified Software Development Process. Addison-Wesley, 1999.

15. T. Lodderstedt, D. Basin, J. Doser. Secureuml: A uml-based modeling language for model-driven security. In: J.-M. Jézéquel, H. Hussmann, S. Cook (eds.): UML 2002. Lecture Notes in Computer Science, vol. 2460, Springer, 2002.
16. K. Mantell, "From UML to BPEL", IBM-developerWorks, 2003. See: http://www-106.ibm.com/developerworks/webservices/library/ws-uml2bpel/
17. T. Moses (ed.), et al., "XACML Profile for Web-Services", XACML TC Working draft, Version 04. September 29, 2003.
18. A. Nadalin, C. Kaler, P. Hallam-Baker, R. Monzillo, 2004. Web Services Security: SOAP Message Security 1.0 (WS Security 2004), OASIS Standard 200401, March 2004. See: http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf
19. UML 2.0 OCL Final Adopted specification. http://www.omg.org/cgi-bin/doc?ptc/2003-10-14
20. G. Starke. Web Service Engineering. Objekt-Spektrum 01/2002, SIGS DATACOM
21. I. Schieferdecker, B. Stepien. Automated Testing of XML/SOAP based Web Services. 13. Fachkonferenz der Gesellschaft für Informatik (GI) Fachgruppe "Kommunikation in verteilten Systemen", Leipzig, 2003.
22. J. Wainer, P. Barthelmess and A. Kumar: W-RBAC – A Workflow Security Model Incorporating Controlled Overriding of Constraints In International Journal of Cooperative Information Systems. Vol. 12, No 4 (2003) 455-485.
23. W3C Recommendation XML Schema Part 2: Datatypes. 02 May 2001