Aspect-Oriented Modeling of Access Control in Web Applications

Gefei Zhang Hubert Baumeister Nora Koch Alexander Knapp

Institut für Informatik, LMU München
Oettingenstr. 67, 80538 München, Germany

{zhangg, baumeist, kochn, knapp}@pst.ifi.lmu.de

ABSTRACT

Access control is only inadequately supported by the common design methods for Web applications. We propose an aspect-oriented technique for solving this problem. Our approach is an extension of UML-based Web Engineering. UML state machines are used to specify the access control rules of navigation nodes. Aspect-oriented modeling helps modularize the design.

Keywords

Aspect-oriented modeling, Web engineering, access control, UML

1. INTRODUCTION

Both classic Web applications — information systems with Web interface — and Web applications of the new generation, which include implementation of complex business processes, face the problem of access control, i.e., which user may access which page. While nowadays there are mature techniques of implementing access control in Web applications, modeling access control, however, is only inadequately supported by the common Web engineering methods [4, 6, 8]. Access control is commonly modeled as part of the navigation structure of an application and thus the same navigation structure, e.g. the authentication process, is repeated for every element that needs access control. This introduces redundancy to the models.

Access control is — in the same way as most of the other user modeling features are — a cross cutting feature in Web applications. They are not only orthogonal to the functionality of a Web application, but also often apply to several classes of Web pages [3]. A modular modeling would therefore be an important improvement to the reusability of the design and the flexibility of the system.

We propose an aspect-oriented technique for modeling access control in Web applications. Our approach uses UML state machines to specify access control rules for navigation nodes and benefits from the modularity of aspect-oriented modeling. It is UML 1.x [14] compliant since the model elements are defined by an extension of the UML metamodel. The approach is explained in the context of the UML-based Web Engineering method (UWE) [10, 11], but can be easily embedded in other Web engineering methods.

The remainder of this paper is organized as follows: after a brief introduction to UML-based Web Engineering in the following Section 2, Section 3 describes our approach of modeling access control in Web applications by means of an example. Section 4 discusses some related work, before Section 5 outlines some future possibilities of aspect-oriented modeling of Web applications.

2. UML-BASED WEB ENGINEERING

Similar to other Web engineering methods, UML-based Web Engineering (UWE) separates the concerns of a Web application in different points of view: the content, the navigation structure, the business processes, and the presentation. The distinguishing feature of UWE is its UML compliance since UWE is defined in the form of a UML profile and an extension of the UML metamodel (for more details see [11]).

In UWE, the content of Web applications is modeled in a conceptual model where the classes of the objects that will be used in the Web application are represented by instances of ≪conceptual class≫ which is a subclass of the UML Class. Relationships between contents are modeled by UML associations between conceptual classes.

The navigation model is based on the conceptual model and represents the navigation paths of the Web application being modeled. A ≪navigation class≫ represents a navigable node in the Web application and is associated to a conceptual class containing the information of the node. Navigation paths are represented by associations: An association between two navigation nodes represents a direct link between them. Additional navigation nodes are access primitives used to reach multiple navigation nodes (≪index≫ and ≪guided tour≫) or a selection of items (≪query≫). Alternative navigation paths are modeled by ≪menu≫s.

A navigation model can be enriched by the results of the process modeling which deals with the business process logic of a Web application and takes place in the process model. The presentation model is used to sketch the layout of the Web pages associated to the navigation nodes. The interested reader is referred to [10] for more details on the UWE notation and method.

As the navigation structure laid down in the UWE navigation model specifies the access paths to the content of the Web application, the navigation model is the appropriate starting point to investigate how to integrate access control in UWE.

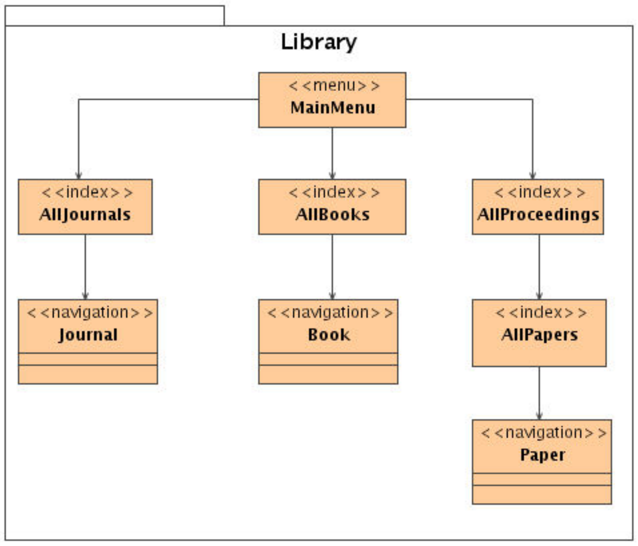


Figure 1: Navigation diagram of the online library.

As an example, Fig. 1 shows the navigation model of an online library which contains three kinds of publications: journals, books, and conference proceedings. For each kind of publication there is an index from which the publications of this kind can be reached. A main menu from which the three indexes can be reached is also provided; this is the home page of the Web application.

3. THE APPROACH

Based on the navigation model of UWE and sticking to its principles of UML compliance, our approach uses UML state machines to model access control in Web applications. In the following, we illustrate our approach by the online library example.

Suppose the business policy of the online library is that the access to the publications themselves should be reserved to registered users only, while their indexes and the main menu should be accessible by everyone. Therefore, access to the publication must be protected by a designated `LogOn` process.

Note that an access control strategy built on such a business policy requires the existence of a user model. Depending on the business policy and the access control strategy, different types of user models can be built, e.g. role models, cognitive model, task models, etc. Overlay models offer a simple and powerful structure to model user characteristics [3], associating them directly to the domain information. For more details see the reference model for adaptive hypermedia systems [12].

3.1 Using State Machines for Access Control

Simply adding the `LogOn` process to all navigation paths leading to the publication nodes may not be sufficient. A characteristic of Web applications is the possibility of navigation nodes being directly accessed via external links, thus undermining link-based access control. Therefore access control should be part of the behavior of the protected nodes.

We extend UWE by associating to each navigation node a state

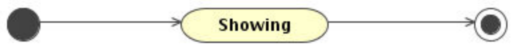


Figure 2: Default state machine of every navigation node.

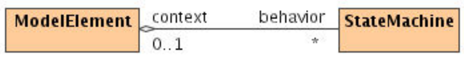


Figure 3: Excerpt of the UML metamodel: ModelElement and StateMachine.

machine which specifies the detailed behavior the navigation node. The idea is that when a navigation node is to be shown, its corresponding state machine is executed. Only when the state machine reaches state `Showing`, the navigation node is shown to the user. Each navigation node in the navigation model has a default state machine which is shown in Fig. 2.

In the UML 1.5 metamodel [14] each model element can be associated with some state machine. The model element is called context of its state machines and the state machines are the behaviors of their context (see Fig. 3). We introduce a constraint that in the navigation model each navigation node must have exactly one state machine:

```
context NavigationModel
inv:
  self.ownedElement ->
  select(e | e.oclIsKindOf(NavigationNode))->
  forAll(a | a.behavior->size() = 1)
```

For navigation nodes that are to be protected, we can now refine the default state machine by a state machine modeling the access protection. The classes `Journal`, `Book`, and `Paper`, for example, are subject to the same access control rules specified by the state machine in Fig. 4 which describes the following behavior:

1. in the state `VerifyLogOn`, the application checks if the user has already logged on;
2. if so, the state changes to `Showing`, the required page is shown;
3. if not, the state changes to `LogOn`, which means, as indicated by the stereotype ≪node≫, that the user is led to another navigation node where he can input his user name and password;
4. after `LogOn`, the state changes to `VerifiyPassword` and the system checks if the user name and the password are valid;
5. if they are valid, the state changes to `Showing` and the protected navigation node is shown;
6. otherwise the user is led to another node `Error` which informs him that the password is wrong and where he has the choice between trying to log on again or canceling the action.

This way, we obtain a UML compliant specification of the access control rules. Note that this specification is on a highly abstract

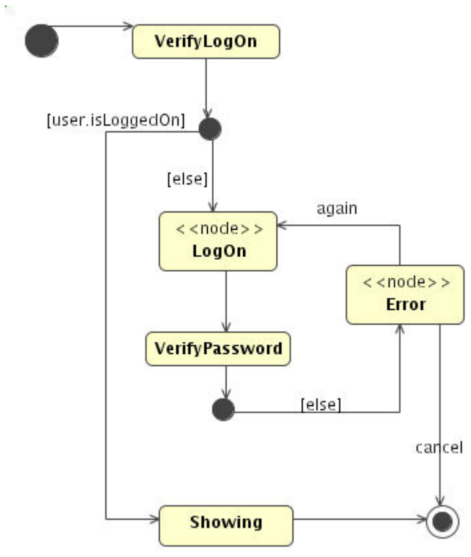


Figure 4: Refined state machine, specifying the access control mechanism of a single navigation node.

level. On this level, we do not really want to design implementation details, e.g. how to determine if a user is already logged in. The characteristic of this (naïve) approach is that those navigation nodes that are not subject to access control have a default state machine that contains only one non-trivial state while the protected navigation nodes have more complex state machines which are very similar to each other.

3.2 Making State Machines Aspect-Oriented

The naïve method works well as long as only one single navigation node is subject to access control. However, in a real application usually much more than one single page needs to be protected. In the example of the online library, access to all three classes `Journal`, `Book`, and `Paper` is reserved for registered users only. Moreover, all of them have the same access rule given by the state machine shown in Fig. 4.

However, since a state machine can have at most one context (cf. Fig. 3), we have to replicate the state machine in Fig. 4 for each class `Journal`, `Book`, and `Paper`.

To avoid this redundancy, we need to organize model elements in a new dimension where reusability is not only achieved my means of "vertical" generalization but also some "horizontal" relations. Aspect-Oriented Modeling (AOM) provides such a horizontal relation: aspects. All elements of an aspect have common features.

Extension of the UWE Metamodel

We introduce the concept of aspects into UWE by extending the UWE metamodel (see Fig. 5). Since an aspect is supposed to contain navigation nodes, we define an Aspect to be a subclass of Package. The notation for an aspect is a package with the stereotype ≪aspect≫. Between Aspect and NavigationNode a new association contains is introduced. Note that contains is an $m{:}n$ relationship: An aspect contains one or more navigation nodes, and a navigation node can be contained in several aspects. The aspects of a navigation node are ordered. In addition, also aspects may be

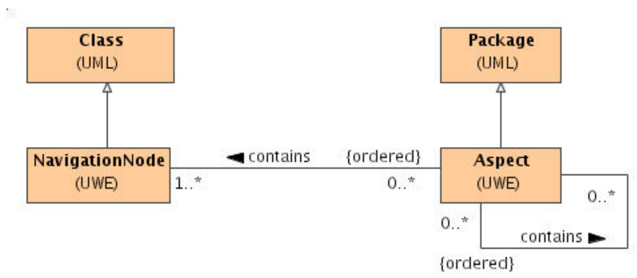


Figure 5: Extension of the UWE metamodel by Aspect

contained in aspects. The notation used for navigation nodes and aspects contained in aspects is the same as the notation for UML classes and packages imported by a package, i.e. navigation nodes and aspects can be drawn inside an aspect (e.g. Fig. 7).

When several navigation nodes are subject to the same access rules, an instance of Aspect that contains all of them is created. Fig. 6(a) shows how aspect `AccessControl` contains the classes `Journal`, `Book`, and `Paper`. In order to specify the common access rules, we employ the association shown in Fig. 3 again: Since Aspect is a subclass of Package, an aspect is also a model element and can therefore have a state machine as its behavior, which we use to define the common access control rules of all of the navigation rules contained in the aspect. Note how the access rules are specified by the aspect resp. its state machine, the state machines of the contained navigation nodes are trivial again.

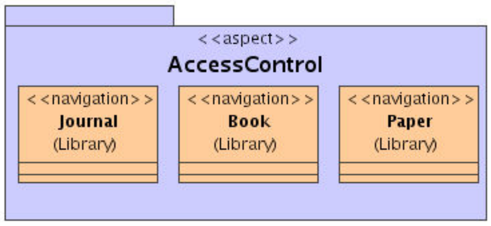
In our running example, aspect `AccessControl` has a very similar state machine, which is shown in Fig. 6(b), to the one that each of the access controlled navigation nodes had in the naïve approach (see Fig. 4). The only difference is that instead of `Showing`, which meant that the context of the state machine should be shown to the user, the state machine of an aspect has the state `OK`.

The (informal) semantics is as follows: if a navigation node is contained in an aspect, then before the state `Showing` in its state machine can be reached and the node can be shown to the user the state machine of the aspect is run. When the state `OK` is reached, the (trivial) state machine of the navigation node itself is run. In other words, the navigation node specific state machine can be seen as a substate of `OK` and when the state `Showing` is reached, the navigation to this node is possible. This semantics is depicted in Fig. 6(c).

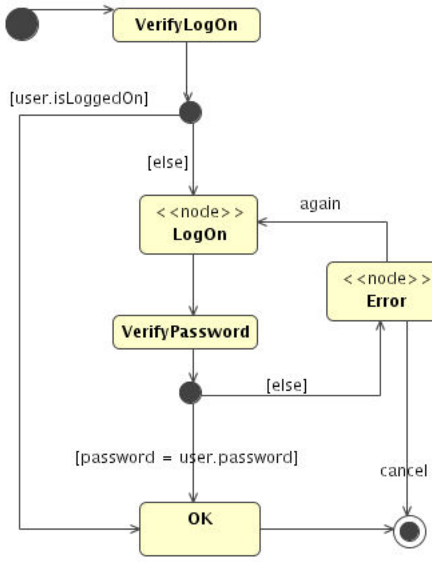
Compared to the naïve method, the characteristic of the aspect-oriented approach is therefore that the access control rule is no longer defined in single nodes, but in the aspect containing them and that all navigation nodes contained in the same aspect have the same access control rules. This way the modeling of access control in Web applications is modularized. The rules thus need to be specified only *once* and redundant specification can be avoided.

The state machine of a navigation node does not need to be trivial. Since a node specific state machine is understood as a substate machine of the state `OK` in the state machine of its aspect, complex access control rules can as well be specified in it, as long as the rule is also navigation node specific.

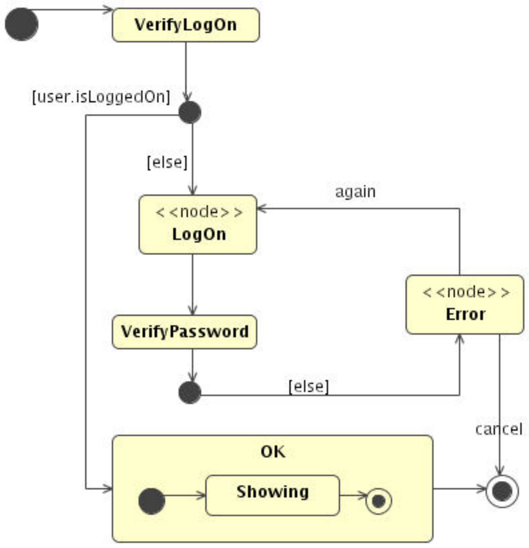
Suppose, e.g., that the policy of the online library is that while pa-



(a) Aspect `AccessControl` contains the concerned nodes.



(b) State machine of the aspect.



(c) The semantics.

Figure 6: Aspect `AccessControl`, specifying the rules of access control for all three navigation classes: `Journal`, `Book` and `Paper`.

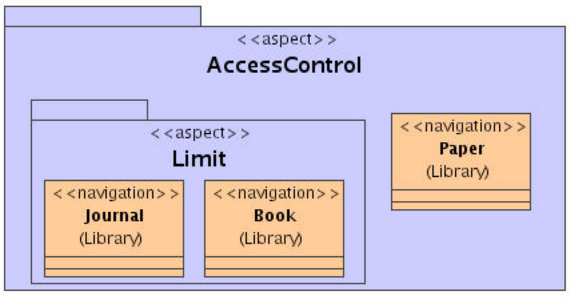


Figure 7: Aspect `Limit`, contained in `AccessControl`.

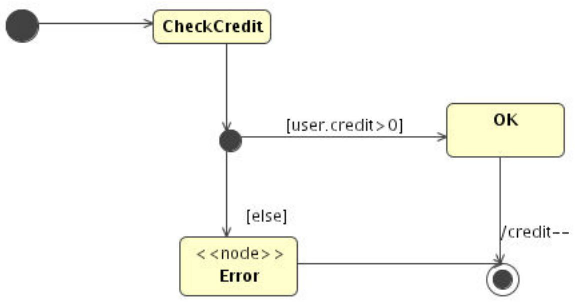


Figure 8: State machine of Aspect `Limit`, specifying limiting of downloads of books and journals.

pers and journals can be downloaded as much as a registered user likes, book downloads should be limited to, e.g., to a couple of books per month. Then the state machine of the class `Book` may be extended to specify counting of downloaded books and checking if the quota has been already exhausted.

Aspects of Aspects

Aspects can themselves be contained in other aspects (cf. Fig. 5). For example, suppose that the online library limits the download of both books and journals. Since this aspect refers to two classes, we introduce a new aspect `Limit` that contains them and is itself contained in aspect `AccessControl`. We define a state machine for `Limit` to specify the limiting behavior. Figure 7 shows this aspect and Fig. 8 shows its state machine. It is assumed that books and journals are equally expensive, otherwise the state machine of the aspect `Limit` can be extended so that different values are subtracted from the user's account.

The (informal) semantics of aspects of aspects is similar to that of aspects of navigation nodes: when a navigation node contained in an aspect is requested by the user, the system first looks recursively for its aspect and its aspect's aspect. The state machine of the most upper aspect is first run, until its `OK` state invokes the state machine of its sub-level aspects to run, and so on, until in the last step the state machine of the navigation node itself is run, where in the state `Showing` the Web page is really shown to the user. In our example, first the state machine of `AccessControl` is run, making sure that the current user is logged in, and, in the `OK` state, the state machine of `Limit` is run, checking that the user still has sufficient credit to download the publications, and, only in the OK state of Limit, the state machine of Book resp. Journal can be run, where in the state Showing the publication is shown to the user. Note that if a user cancels the logon process the state machine stops without having visited the state OK. This means that the state machine of the contained aspect, in this case aspect Limit, will not be run.

Multiple Aspects

According to the metamodel in Fig. 5, the behavior of a navigation node or an aspect can be modified by multiple aspects. The order in which the aspects are applied is given by the order of the aspects in the contains association.

4. RELATED WORK

The OO-H [7] approach proposes the use of personalization rules to model access control aspects. These rules are associated to the filters included in the navigation model. Filters are applied to navigation links. Thus each link requiring access control needs the definition of the corresponding filter. This means that if a protected navigation node is reachable by several links, each link needs to define a filter for access control. This introduces additional redundancy, which opens the door for bugs, like forgetting a required filter on a link accessing the protected node. Conversely, our approach avoids this redundancy because the access control is associated with the navigation node and not with navigation links.

Other approaches with similar redundancy problems are OOHDM [15] and WebML [6]. OOHDM separates access control from navigation by adding a wrapper class for each navigation node which requires access control, but does not treat access control as a cross-cutting feature. WebML provides a predefined login operation, which implements the verification of the identity of a user accessing the site. More complex rules for access control, however, require definition of externally defined operations.

In WSDM [5] an adaptation specification language is defined that allows designers to specify at the level of the navigation model which adaptations of the navigation structure can be performed at runtime. Although a visual representation of the rules is missing, rules are defined orthogonally to the navigation functionality as designers are allowed to define rules on one single element (node, link) and on group of elements, but these rules only pertain to the navigation structure and not to the general behavior of the navigation node. In particular, the rules do not seem to be able to express the triggering of some action when a node is accessed.

SecureUML [13, 2] is a UML profile extending UML with a declarative description of the role-based access policies of protected objects. In SecureUML the access control policy of protected objects is described by authorization constraints. An authorization constraint contains a precondition that must hold before a protected object can be accessed. In contrast to SecureUML, our approach uses UML state machines to visualize the steps to be done to satisfy the authorization constraints, e.g. by modeling the login process. Thus, our approach is complementary to SecureUML.

Jürjens [9] defines a UML profile, UMLsec, for model based security engineering. While UMLsec is focused on data confidentiality and integrity, our approach concentrates on modeling access control in Web applications.

Other researchers have given graphical notations for modeling aspect-oriented programms [1, 16, 17]. In contrast, the focus of our work is aspect-oriented modeling of software systems.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we have used aspect-oriented techniques to model access control in Web applications. Access control is associated with navigation nodes where each navigation node has a state machine describing how to gain access to that node, e.g. by first authenticating to the Web application. Access control is a cross-cutting feature of Web applications. Navigation nodes using the same rules for authentication are grouped under the same aspect. This facilitates modification of access rules for navigation nodes without having to interfere with the functional model of the Web application. There are other cross-cutting concerns in Web-applications besides access-control. We plan to apply the techniques developed in this paper to requirement elicitation, process modeling, and personalization.

Currently we are working on an implementation of the aspect-oriented extensions of UWE in ArgoUWE, a UML 1.x based CASE tool that we have developed to support modeling Web applications with UWE [10].

Furthermore, we plan to integrate the presented approach with the authorization constraints of SecureUML [13, 2] and to develop a formal semantics of the aspect-oriented extension of UWE. This will allow us to use model-checking techniques to verify that the access rules satisfies the corresponding authorization constraints.

6. ACKNOWLEDGMENTS

This work has been partially supported by the EU project AGILE (IST-2001-32747). We would also like to thank Axel Rauschmayer for fruitful discussions related to this paper.

7. REFERENCES

[1] M. Basch and A. Sanchez. Incorporating Aspects into the UML. In *Proc. 3rd Wsh. Aspect-Oriented Modeling with UML, Boston*, 2003.

[2] D. Basin, J. Doser, and T. Lodderstedt. Model driven security for process-oriented systems. In *Proc. 8th ACM Symp. Access Control Models and Technologies (SACMAT 2003)*, pages 100–109. ACM Press, June 2003.

[3] P. Brusilovsky. Methods and Techniques of Adaptive Hypermedia. *International Journal of User Modeling and User-Adapted Interaction*, 6(2–3):87–129, 1996.

[4] S. Casteleyn, O. De Troyer, and S. Brockmans. Design Time Support for Adaptive Behavior in Web Sites. In *Proc. 18th ACM Symp. Applied Computing*, pages 1222–1228. ACM Press, 2003.

[5] S. Casteleyn, O. De Troyer, and S. Brockmans. Design time support for adaptive behaviour in Web sites. In *Proc. 18th ACM Symposium on Applied Computing, Melbourne, USA*, pages 1222 – 1228. Publ. ACM, 2003.

[6] S. Ceri, P. Fraternali, A. Bongio, M. Brambilla, S. Comai, and M. Matera. *Designing Data-Intensive Web Applications*. Morgan Kaufmann, 2003.

[7] I. Garrigós, J. Gómez, and C. Cachero. Modelling Dynamic Personalization in Web Applications. In J. M. C. Lovelle,

B. M. G. Rodríguez, L. J. Aguilar, J. E. L. Gayo, and M. del Puerto Paule Ruíz, editors, *Proc. 3rd Int. Conf. Web Engineering*, volume 2722 of *Lect. Notes Comp. Sci.*, pages 472–475. Springer Verlag, 2003.

[8] J. Gómez, C. Cachero, and O. Pastor. Extending a Conceptual Modelling Approach to Web Application Design. In B. Wangler and L. Bergman, editors, *Proc. 12th Int. Conf. Advanced Information Systems Engineering (CAiSE'00)*, volume 1789 of *Lect. Notes Comp. Sci.*, pages 79–93. Springer Verlag, 2000.

[9] J. Jürjens. *Secure Systems Development with UML*. Springer Verlag, 2004.

[10] A. Knapp, N. Koch, G. Zhang, and H.-M. Hassler. Modeling Business Processes in Web Applications with ArgoUWE. In T. Baar, A. Strohmeier, A. Moreira, and S. J. Mellor, editors, *Proc. 7th Int. Conf. Unified Modeling Language (UML'04)*, volume 3273 of *Lect. Notes Comp. Sci.*, pages 69–83. Springer Verlag, 2004.

[11] N. Koch and A. Kraus. Towards a Common Metamodel for the Development of Web Applications. In J. M. C. Lovelle, B. M. G. Rodríguez, L. J. Aguilar, J. E. L. Gayo, and M. del Puerto Paule Ruíz, editors, *Proc. 3rd Int. Conf. Web Engineering (ICWE'03)*, volume 2722 of *Lect. Notes Comp. Sci.*, pages 497–506. Springer Verlag, 2003.

[12] N. Koch and M. Wirsing. The Munich Reference Model for Adaptive Hypermedia Applications. In P. De Bra, P. Brusilovsky, and R. Conejo, editors, *Proc. Adaptive Hypermedia and Adaptive Web-Based Systems (AH'02)*, Lect. Notes Comp. Sci., pages 213–222. Springer Verlag, 2002.

[13] T. Lodderstedt, D. Basin, and J. Doser. SecureUML: A UML-Based Modeling Language for Model-Driven Security. In J.-M. Jézéquel, H. Hußmann, and S. Cook, editors, *Proc. 5th Int. Conf. Unified Modeling Languageng Language (UML'02)*, volume 2460 of *Lect. Notes Comp. Sci.*, pages 426–441. Springer Verlag, 2002.

[14] Object Management Group. Unified Modeling Language Specification, Version 1.5. Specification, OMG, 2003. `http://www.omg.org/cgi-bin/doc?formal/03-03-01`.

[15] G. Rossi, A. Fortier, J. Cappi, and D. Schwabe. Seamless Personalization of E-Commerce Applications. In *2nd Int. Wsh. Conceptual Modeling Approaches for e-Business (eCOMO'01)*, volume 2465 of *Lect. Notes Comp. Sci.*, pages 457–470. Springer Verlag, 2001.

[16] D. Stein, S. Hanenberg, and R. Unland. An UML-based Aspect-Oriented Design Notation For AspectJ. In *Proc. 1st Int. Conf. Aspect-Oriented Software Development*, pages 106–112. ACM, 2002.

[17] D. Stein, S. Hanenberg, and R. Unland. On Representing Join Points in the UML. In *Proc. 2nd Wsh. Aspect-Oriented Modeling with UML*, 2002.