Towards Modeling Role-Based Pageflow Definitions within Web Applications

Ernst Oberortner, Martin Vasko and Schahram Dustdar
{e.oberortner|m.vasko|dustdar@infosys.tuwien.ac.at}

Technical University of Vienna
Distributed Systems Group
Argentinierstraße 8/184-1
A-1040 Vienna
Austria

Abstract. Model-Driven Software Development (MDSD) can be used to enhance developing and maintaining web applications. Furthermore, security plays a crucial role in the area of web applications. A seamless integration of access control and modeling web applications becomes important. This work introduces model-driven integration of security concerns into the development life cycle of web applications. In this paper, a static design model is introduced which enables the assignment of Role-Based Access Control (RBAC) to the pageflow at design time. The approach is demonstrated by a generated web application with defined security constraints which can be deployed on the Apache Tomcat web server.

1 Introduction

Due to the increasing complexity of web applications, their development and maintenance becomes more difficult [1]. To find a remedy, Model-Driven Software Development (MDSD) can be used. It addresses the reduction of development time, the quality of the software, a high level of abstraction for better maintenance, as well as to achieve better portability and interoperability [2].

Besides rising demands on the web application architecture, the seamless integration of access control becomes important. Role-Based Access Control (RBAC) is a technique for multiple user systems and networks and was introduced 1992 by Ferriaolo and Kuhn [3]. RBAC is used by organizations to protect their information resources from unauthorized access [4]. Nowadays, RBAC is a favored technique because it provides an easy way of administrating security [5]. RBAC provides possibilites to model security requirements of today's web applications through complex role hierarchies and restricted access to predefined users.

A common problem of securing web applications lies in the late integration of security at the test phase or at the end of the development process. It is recommended to integrate security into the life cycle of a web application [6]. The integration of security with MDSD leads to a high level formulation of

security aspects [7]. The model-driven integration of RBAC models into system design models has been proved to be a feasible approach [8].

This work introduces a static design model which enables role-based securing of web applications based on the pageflow. Our approach is focused on the design of secure web applications. For the time being, no models are provided for modeling security which is established during runtime. Our approach is demonstrated by a generated JavaServer Faces (JSF) [9] web application with defined security constraints, and which is deployed on the Apache Tomcat web server [10].

This paper is organized as follows: Section 2 demonstrates a motivating example for a model-driven approach for securing web applications. Section 3 shows the background of used technologies and methodologies. An architectural overview gives Section 4. Section 5 demonstrates the implementation of our approach and an example. Section 6 highlights related work. Finally, Sections 7 and 8 complete the paper with future prospects and a conclusion.

2 Motivating example

A sample web application is introducing the motivation of our approach. The application implements a user administration where visitors[1] can list all users, request detailed information on each user, add new users, delete existing users, and change user details.

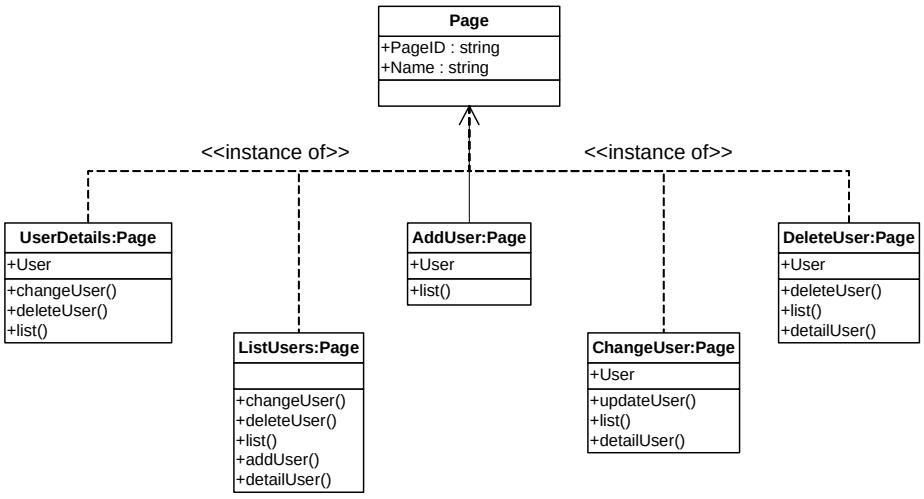

Fig. 1. An Object Diagram of the Web Pages Structure

[1] As the term users might be ambiguous, we refer to web application users as visitors, and the term users refers to the design model.

Figure 1 illustrates a UML [11] object diagram of the motivating web application. Each `Page` class represents a web page, implementing the defined operations. Beyond the stated functionality, the following security policy must be applied:

1. Deny access to unregistered visitors to all web pages
2. Registered visitors need to be assigned to one of the following roles: `Admin`, `GroupLeader`, or `GroupMember`
3. Members of the role `Admin` may access all web pages
4. Members of the role `GroupMember` may request only user details
5. Members of the role `GroupLeader` may access all information concerning the members of the role

Derived from these security constraints, the UML statechart models, illustrated in Figure 2, demonstrates the resulting pageflows of each role.

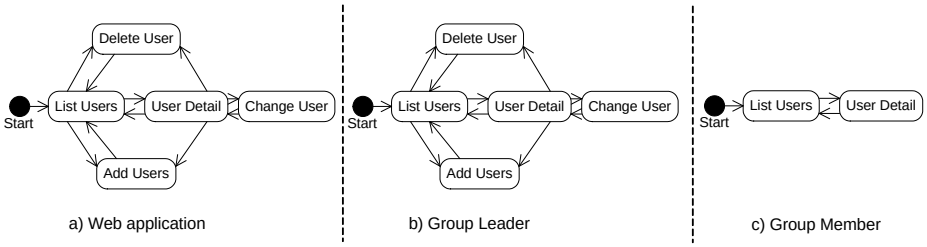

Fig. 2. UML Statechart Models of the Role

Figure 2 a) defines all web pages of the web application. Figure 2 b) lists all pages reachable for members of the role `GroupLeader`. Whereas from a functional point of view, the state charts a) and b) are equivalent, the result of the requests needs to be different (refer to security policy 5). The content of `ListUsers` web page depends on the role of the visitor. If the visitor is assigned to the `GroupAdmin` role, the list contains all registered users. If the visitor is assigned to the `GroupLeader` role, the list includes only users assigned to his/her role. Figure 2 c) presents the state chart for members of the `GroupMember` role. Visitors assigned to this role are only allowed to request information about users concerning to their role.

This sample web application indicates two restrictions:

1. **Restricted Web Pages**: Pages of the web application are restricted to a predefined role of visitors. E.g., members of the `GroupMember` role do not have access to the `Add-`, `Change-` or `Delete User` web pages.
2. **Restricted Content**: Results of requests to the application depend on the requester's role membership. E.g., members of the `GroupLeader` role, requesting a list of users, retrieve only users assigned to their role.

The stated restrictions can be divided into two categories: restrictions that can be defined (1) at design time, and (2) at runtime of the web application. E.g., at design time it can be defined that only users of the `Admin` or `GroupLeader` roles can access the `AddUsers` web page. But, the content of the web page depends on the role of the visitor. The role of the visitor is known only at runtime. Hence, the content must be generated at runtime. This is established by the business logic of the web application.

We identified the definition of restrictions at design time as an ideal candidate to integrate an RBAC model into the static design model and generate web applications from these models. Throughout the paper, the approach of applying RBAC on the pageflow is introduced.

3 Background

For a better understanding, this section provides an overview of used technologies and methodologies. First, a definition of our understanding of a pageflow is given. Afterwards, the application of MDSD on our approach is demonstrated.

3.1 Definition of Pageflow

The Model-View-Controller (MVC) [12] pattern is a design pattern which divides the responsibilities clearly into Model, View and Controller. The Model encapsulates data and behavior of web applications, independent of their representation. The View represents the presentation layer and is responsible for rendering the data, according to the type of client, by processing the results of the Model. The Controller has to select the subsequent web page or view which should be displayed to visitors by handling visitor's interactions, e.g., submitting the entered data of a form. These interactions are performed by calling actions on the Model. Based on the interactions and the outcomings of the performed actions, the Controller selects the next web page which is displayed to the visitor.

The basis of selecting the appropriate web page is defined in the pageflow. Hence, the pageflow describes to which web pages visitors can navigate, dependent on the current page. In our work, the visitor can only navigate to other web pages by interactions with hyperlinks or buttons. The subsequent web page depends on the hyperlink or button which the visitor clicks and in which roles the visitor is member of. A well arranged readability of the pageflow can be achieved by defining the pageflow with Java-like IF-ELSE statements.

3.2 Model-Driven Software Development (MDSD)

Figure 3 demonstrates the functioning of MDSD based on our approach of securing web applications. The `Pageflow- & RBAC Metamodel` provides constructs for modeling the pageflow and RBAC for web applications, i.e., the abstract syntax. `Web Application Models` are instances of the metamodel. The validity of models of web applications is checked by a `Model Validator` which consists

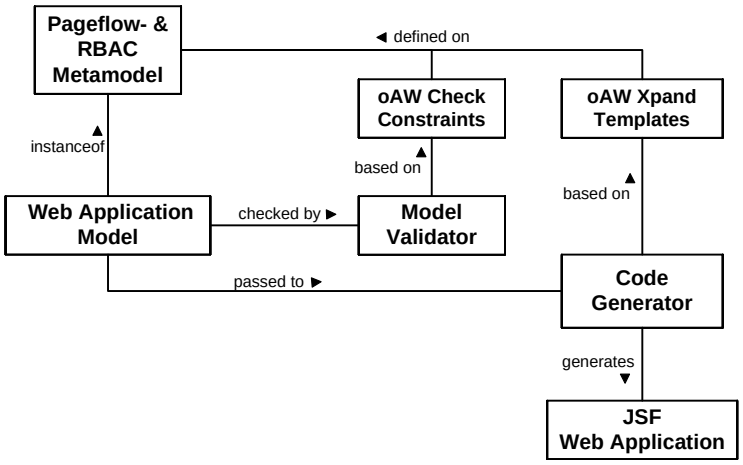

Fig. 3. The approach in the MDSD chain

of `Constraints` which are defined on the constructs of the metamodel. Valid models are passed to the `Code Generator` which generates executable code and the needed configuration files, i.e., in our case a JSF web application for the Apache Tomcat web server. In our work, the code generation process is based on `Templates` which are also defined on the constructs of the metamodel.

4 Architectural Overview

Figure 4 demonstrates the activity chain of developing web applications based on MDSD. First, the model of a web application is defined by developers or domain experts (activity 4(a)). The rectangles describe the accessible web pages for the `GroupLeader` and `GroupMember` roles. The big rectangle depicts accessible web pages for `GroupLeaders`, and the small one depicts accessible web pages for `GroupMembers`. Generating modeled web application through MDSD is demonstrated in activity 4(b). Activity 4(c) shows the generated web pages and platform-specific configuration files, e.g., the deployment descriptor *web.xml* for web applications of the Apache Tomcat web server [10]. Activity 4(d) refers to the deployment phase. In the deployment phase, developers have to add hand-written code or platform-specific configurations to the generated files of the web application. Afterwards, the web application can be deployed and executed on a web server.

5 Our Approach

This section presents implementation details to achieve joint modeling of RBAC and pageflow of web applications. First, the used technologies are mentioned.

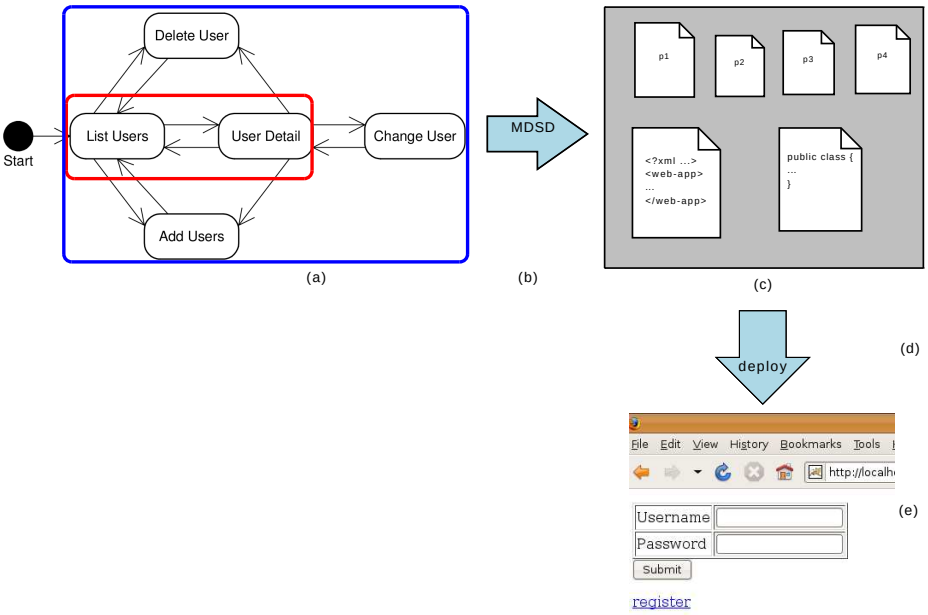

Fig. 4. The Procedures of a Model-Driven Approach

Then, the most important implementation details are presented: the metamodel, the model validation constraints and the code generation templates. Afterwards, an insight to the generated code is given. Finally, the feasibility of our approach is demonstrated on the motivating example presented in Section 2.

5.1 Used Technologies

The metamodel is defined through an Eclipse Modeling Framework (EMF) Ecore [13] model. For the time being, models are described in the XML Metadata Interchange (XMI) [14] format which is used for interchanging models between different modeling tools. Model validation and code generation are implemented by the use of openArchitectureWare (oAW) [15]. oAW is an Eclipse [2] based framework that affords model validation and template based code generation. Model validation is done through constraints which are defined in the oAW Check Language. Also, oAW provides the Xpand language which is used for the definition of the code generation templates. For the time being, only secured JavaServer Faces (JSF) [9] web applications for the Apache Tomcat web server [10] are generated.

[2] http://www.eclipse.org

5.2 A Proposed Metamodel

The metamodel is presented by a UML class diagram in Figure 5. Each `WebApplication` consists of a number of `Pages`, and one page is depicted as the `startPage`. `Pages` contain `NavigationRules` which define the pageflow. The classes `If`, `ElseIf` and `Else` are defined to achieve a Java-like `IF-ELSE` pageflow definition. These classes are derived from the `Decision` class which contains a reference to the subsequent web page through the `gotoPage` association. The referenced web page is displayed if the outcome of the performed actions is equivalent to a corresponding `outcome` attribute, specified in the `If` and `ElseIf` classes. If no corresponding `outcome` attribute is found, the web page specified by the `gotoPage` reference of the `Else` class is displayed to the visitor.

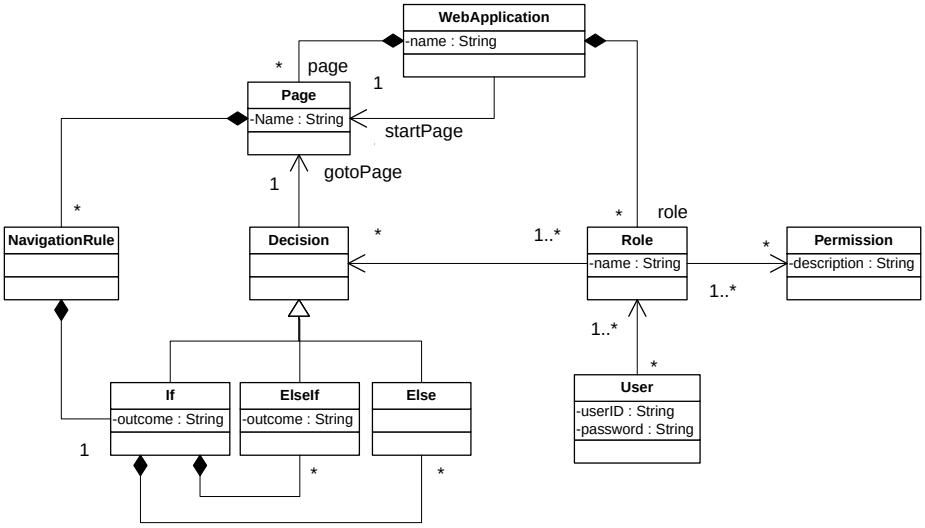

Fig. 5. Pageflow- & RBAC Metamodel

The assignment of RBAC to the definition of the pageflow is provided through the association between the `Decision` and `Role` classes. Hence, an `IF-ELSE` definition of a rolebased pageflow definition is achieved, e.g.,

```
IF outcome="..." AND role="..." THEN gotoPage="..."
```

As introduced by Sandhu et. al. [5], a `Role` consists of one or more `Users` and of one or more `Permissions`. For the time being, a `Permission` is responsible to define if a user has access to a certain web page or not. It is planned that more permissions will be introduced, e.g., write or publish web pages.

It is possible to use Aspect Oriented Modeling (AOM) for modeling the pageflow and RBAC. AOM supposes a clear separation into multiple models. This means, that our metamodel can be splitted into two separate metamodels,

one for modeling the pageflow, and one for modeling RBAC. But, AOM needs clear defined integration points for merging the models. Our approach defines an association between the classes `Decision` and `Role`, instead of integration points. But, the use of AOM seems to be an alternative modelling technique. This approach is subject of further research and evaluation and is out of the scope of this work.

5.3 A Model of a Role-Based Pageflow

Figure 6 shows an excerpt of a UML object diagram which represents the motivating web application introduced in Section 2. Page `ListUsers` contains one `NavigationRule` which consists of one `If` which refers to the `UserDetails` page via the `gotoPage` reference. The specified `outcome` attribute in the `if` object contains the string `gotoUserDetails`. Furthermore, a `roles` reference exists, which references to the `GroupMember` role. Hence, the `IF-ELSE` definition of the rolebased pageflow looks like:

```
IF outcome="gotoUserDetails" AND role="member1" THEN
                gotoPage="UserDetails"
```

All navigation rules of all web pages and their associations to roles are defined similar.

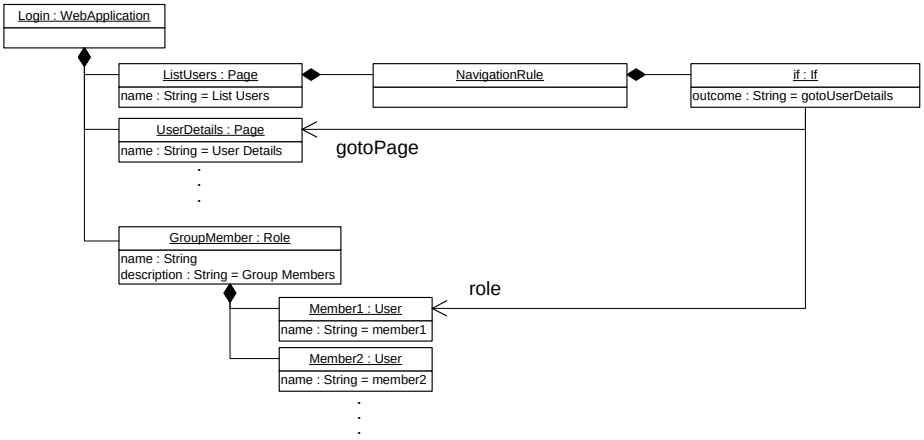

Fig. 6. An Object Diagram of a Web Application

5.4 Model Validation

Figure 7 shows an excerpt of defined constraints by the oAW Check language. The first constraint defines that each `WebApplication` must have a `startPage`.

The second one defines that each `Role` must have a `name`. The last one defines that each `NavigationRule` must consist of at least one `if`. If models do not fulfill these constraints, an error is raised, e.g., `'no name for role defined !'`.

```
context WebApplication ERROR 'no start page defined!':
    startPage!=null;

context Role ERROR 'no name for role defined!':
    this.name!=null;

context NavigationRule ERROR 'no navigation rule defined!':
    this.ifDecision!=null;
```

Fig. 7. oAW Check Constraints

5.5 Code Generation Templates

Figure 8 displays an excerpt of the oAW Xpand code generation template to generate the deployment descriptor *web.xml*. The code between the guillemets (« and ») controls the code generation.

```
«DEFINE Root FOR RoleBasedPageflow::WebApplication»
«FILE name+"/WEB-INF/web.xml"-»
...
    «FOREACH this.role AS r»
        «EXPAND GenRole FOR r-»
    <security-constraint>
        <display-name></display-name>
        «EXPAND GenRoleAccessPages (this) FOR r-»
        <auth-constraint>
            <description>«r.description»</description>
            <role-name>«r.name»</role-name>
        </auth-constraint>
    </security-constraint>
    «ENDFOREACH-»
...
«ENDFILE»
«ENDDEFINE»
...
```

Fig. 8. oAW Xpand Template

First, a `Root` label is defined for objects of type `WebApplication`, i.e., that each `WebApplication` gets its own *web.xml* file. For each `Role`, a `<security-constraint>` element is generated which contains all web pages that can be accessed by the members of this role. The `<role-name>` element is stated within the `<auth-constraint>` element. The information of `Pages` is denoted by `<web-resource-collection>` elements.

For the time being, only templates for web applications of the Apache Tomcat web server [10] were implemented. For other web servers, new code generation templates must be implemented.

5.6 Generated Code

Figure 9 contains excerpts of a generated *web.xml* file. The upper portion contains that members of the `GroupMember` role can access the web page `UserDetails`. Underneath, it is described that the web page `DeleteUser` can be accessed by users of the `GroupLeader` role.

```
    <security-constraint>
        <web-resource-collection>
            <web-resource-name>UserDetails</web-resource-name>
            ...
        <auth-constraint>
            <role-name>GroupMember</role-name>
        </auth-constraint>
    </security-constraint>
...
    <security-constraint>
        <web-resource-collection>
            <web-resource-name>DeleteUser</web-resource-name>
            ...
        <auth-constraint>
            <role-name>GroupLeader</role-name>
        </auth-constraint>
    </security-constraint>
...
```

Fig. 9. Generated *web.xml* file

5.7 Example

Figure 10 demonstrates our approach on the basis of the motivating example of Section 2. The example is shown for the visitor `Member1` which is a member of the `GroupMember` role.

First, a login mask appears where the visitor has to enter username and password. The system checks if the visitor is authorized to access. If the visitor is authorized, the system knows the role of the current visitor. Otherwise, the visitor has no access to the web application. The first web page, the `ListUsers` page, offers hyperlinks to other pages. Independent of the visitor's role, all possible links are displayed, because restricted contents are out of scope of this work. If the current visitor clicks the `UserDetails` link, the appropriate web page is displayed. Otherwise, if the current visitor clicks the `DeleteUser` or `AddUser` link, an "Access denied" web page is displayed. The depicted "Access denied" web page is a predefined web page of the Apache Tomcat web server. On the other hand, if the current visitor were member of the `GroupLeader` role, the `DeleteUser` or `AddUser` web page would be displayed.

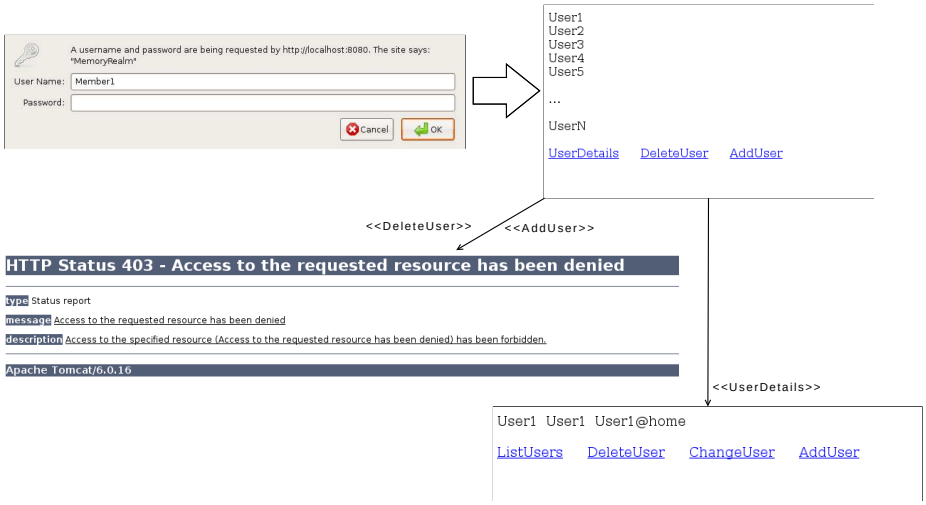

Fig. 10. An Example Web Application

6 Related Work

This section discusses similar approaches which deal with RBAC and web applications. It is divided into two parts: works dealing with (1) modeling web applications and (2) securing web applications.

6.1 Modeling Web Applications

Baresi et. al. [17] present the W2000 MOF metamodel which is divided into four higher level packages: (1) Information, (2) Navigation, (3) Presentation, and (4) Dynamic Behavior. The first three packages define models, i.e., (1) Information Model, (2) Navigation Model, and (3) Presentation Model. The Dynamic Behavior package is spread over all models. The Information Model contains the contents available to the user. The Navigation Model is based on the Information Model. In contrast to our approach, which controls the navigation between web pages, the W2000 Navigation Model controls the navigation between information elements on web pages.

Distante et. al. [18] introduce a model-driven approach for combining the development of the Ubiquitous Web Applications (UWA) design framework, the MVC pattern, and JavaServer Faces (JSF). UWA provides conceptual models at a high level of abstraction which can be used by the stakeholders. To provide useful information to the application developers, the UWA conceptual models are transformed to UML-MVC logical models. Both, the conceptual and logical models are platform independent. They are transformed to platform specific models, i.e., JavaServer Faces.

Schwabe et. al. [20] introduce the Object-Oriented Hypermedia Design Method (OOHDM) which provides multiple models for modeling web applications. The application domain is modeled with a conceptual design model. The navigational model is defined through different views which are build upon the conceptual model. This methodology allows the construction of different views for different user profiles.

The Orbeon Page Flow Controller [22] provides a separation of describing the site logic and the pageflow. Hence, the simplicity gets enhanced because there is no need to write custom logic to perform redirects between pages or to pass arguments. Furthermore, the maintainability is enhanced through the independent implementation of pages and the pageflow. Hence, it is easier to define the pageflow without effecting pages or vice versa. Security aspects are not yet considered in the Oreon Page Flow Controller, i.e., the administrator has to define the security manually in the *web.xml* deployment descriptor. In contrast, our approach provides the modeling of those security aspects. Hence, it will be an interesting research area how to combine our approach with the Orbeon Page Flow Controller for future work.

6.2 Securing Web Applications

J.D. Meier [6] discusses common mistakes in the area of security engineering for web applications: mistakes that are done (1) at the beginning (do-it-all-up-front), (2) at the end (bolt-on), (3) during the test phase (big-bang), (4) by defining security without targets, e.g., firewalls, SSL (buckshot) and, (5) by addressing security after a failure (all-or-nothing). Meier gives the advice that the most effective approach is to "bake" security into the application's lifecycle.

Ceri et.al. introduce the Web Modeling Language (WebML) [19]. The approach provides multiple models for modeling different concerns of web applications. Due the relation to our work, we concentrated on the Navigation and Personalization models. Contextual and non-contextual links are provided. Furthermore, not only the user chooses what content to see, e.g., by clicking hyperlinks, but also the system determines which page and/or content should be show automatically. Regarding security of web applications, the personalization model is introduced, for defining users and user groups, as well as customizations. Users and user groups are mapped to WebML users and groups. At this point we can engage the creation of code generation templates which generate WebML users and groups. Furthermore, we can consider the WebML solution for modeling runtime security concerns in our future works.

Zang, Baumeister, Koch and Knapp [23] propose an aspect-oriented technique for access control in web applications. The approach is associated to the Navigation Model of UML-based Web Engineering (UWE) [24]. UML state machines are used to specify access control. Each navigation node is extended by a state machine that specifies the behavior. Furthermore, constraints are introduced which check that each navigation node has exactly one state machine. To introduce aspect oriented access control, the stereotype « `aspect` » is defined. Aspects are sets of web pages. Access control rules are defined on aspects. Hence,

rules need to be specified once for a number of web pages. Furthermore, aspects can contain other aspects.

Bammigatti and Rao introduce the GenericWA-RBAC approach [25] which is based on the constrained RBAC model, introduced by Sandhu et. al. [5]. The original RBAC model is extended by an ORGN component that represents a set of organizations that have access to the system. A modular system architecture is introduced that is divided into: (1) An *Access Control Module* which is a process responsible for authentication of roles and authorizations, (2) the *ORGN Manager* which determines the origin of the requested organization, (3) a *Role-to-Role mapper* that maps roles of the external organization to native organization roles, (4) a *Role-to-Access right mapper* which decides access rights for the mapped role, and (5) an *Object Management* component that is responsible for checking requested queries against sensitive data.

Lodderstedt et. al. introduce the SecureUML approach [7]. SecureUML is an extension to the UML metamodel for RBAC. The SecureUML model is bound to the `ModelElement` class of the UML metamodel. It provides possiblities to define roles, users, and permissions. Furthermore, additional support for specifying authorization constraints and action types is given. An authorization constraint expresses a precondition imposed on every call to an operation of a particular resource. An action type is a class of security operations to protect particular types of resources. SecureUML is an approach for integrating security concerns into UML based model-driven application engineering.

7 Future Work

Our current approach is able to assign RBAC to the pageflow of web applications based on MDSD. But, there is still a lot of work to do in the future. One of the major future prospects is to have a closer look to existing model-driven approaches for securing web applications, e.g., WebML, OOHDM, or UWE. We have to consider how they can be combined with our work. Furthermore, we need to consider web applications that can be deployed on other web servers than the Apache Tomcat web server. Another future prospects regards the modeling access control of restricted contents, i.e., access control at runtime.

Our approach assigns a simple RBAC model in contrast to the RBAC models introduced by Sandhu et. al. [5]. First, we have to define many-to-many relations for user and permission assignments to roles. Besides we have to define sessions that are also included in RBAC models. Furthermore we want to do more research for applying Static Separation of Duty (SSD) as well as Dynamic Separation of Duty (DSD) to the meta-model. When we apply the complete RBAC models to our approach, we have to evaluate if applying Aspect Oriented Modeling (AOM) on the metamodels is advisable, i.e., a separation into navigation and access metamodels. A separation into multiple models can bring better maintainability as well as a better separation into the development team. But good integration points for merging have to be defined.

A lot of work has been done in the area of modeling security for web services. Certainly, it is important to have a closer look into this approaches for a possible application to our approach. Also, further work can be applied to have a closer look to model actions. Actions are executed if a violation was prevented, e.g., sending a mail to an administrator or raising an alarm.

8 Conclusion

This paper gave an overview of our approach of assigning RBAC to the pageflow of web applications based on MDSD. Thereby, the security concerns can be applied at a high level of abstraction, independent of the desired platform. Our approach concentrated on defining RBAC at the design time of web applications. Security concerns which must be checked at runtime, were out of scope. A static design model, model validation constraints, and a code generation component were introduced. Finally, an example of a generated web application demonstrated our approach.

This work has shown that it is feasible to use MDSD for integrating RBAC to the pageflow of web applications. Hence, this approach facilitates a seamless integration of access control and the modeling web application.

References

1. Sam Chung and Yun-Sik Lee: Modeling Web Applications Using Java and XML Related Technologies. In: HICSS '03: Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS'03) - Track 9, Washington, DC, USA, IEEE Computer Society (2003) 322
2. Thomas Stahl and Markus Voelter: Modellgetriebene Software Entwicklung: Techniken, Engineering, Management. dpunkt.verlag GmbH (2005)
3. D. Ferraiolo and R. Kuhn: Role-Based Access Controls. In: 15th NIST-NCSC National Computer Security Conference. (1992) 554–563
4. Indrakshi Ray and Na Li and Robert France and Dae-Kyoo Kim: Using UML to Visualize Role-Based Access Control Constraints. In: SACMAT '04: Proceedings of the ninth ACM symposium on Access control models and technologies, New York, NY, USA, ACM (2004) 115–124
5. Ravi S. Sandhu and Edward J. Coyne and Hal L. Feinstein and Charles E. Youman: Role-Based Access Control Models. Computer **29**(2) (1996) 38–47
6. Meier, J.D.: Web Application Security Engineering. Security & Privacy, IEEE **4**(4) (July-Aug 2006) 16–24
7. Torsten Lodderstedt and David A. Basin and Jürgen Doser: SecureUML: A UML-Based Modeling Language for Model-Driven Security. In: UML '02: Proceedings of the 5th International Conference on The Unified Modeling Language, London, UK, Springer-Verlag (2002) 426–441
8. David Basin and Jürgen Doser and Torsten Lodderstedt: Model Driven Security: From UML Models to Access Control Infrastructures. ACM Trans. Softw. Eng. Methodol. **15**(1) (2006) 39–91
9. Mann, K.D.: JavaServer Faces in Action (In Action series). Manning Publications Co., Greenwich, CT, USA (2004)

10. Apache Software Foundation: Apache Tomcat. Website (2008)
Available online at http://tomcat.apache.org/.
11. James Rumbaugh and Ivar Jacobsen and Grady Booch: The Unified Modeling Language Reference Manual. Addison - Wesley (1998)
12. Robert Eckstein: Java SE Application Design With MVC (March 2007)
Available online at http://java.sun.com/developer/technicalArticles/javase/mvc/.
13. Eclipse Modeling Framework Project: (2008)
Available online at http://www.eclipse.org/modeling/emf/.
14. Object Management Group (OMG): XML Metadata Interchange (XMI), v2.1.1 (2007)
Available online at http://www.omg.org/technology/documents/formal/xmi.htm.
15. openArchitectureWare: (2008)
Available online at http://www.openarchitectureware.org/.
16. Pierre-Alain Muller and Philippe Studer and Jean Bézivin: Platform Independent Web Application Modeling. In Stevens, P., Whittle, J., Booch, G., eds.: UML. Volume 2863 of Lecture Notes in Computer Science., Springer (2003) 220–233
17. F. Garzotto, L. Baresi, and M. Maritati: W2000 as a MOF Metamodel. (2002) In The 6th World Multiconf. on Systemics, Cybernetics and Informatics-Web Engineering track.
18. Damiano Distante and Paola Pedone and Gustavo Rossi and Gerardo Canfora: Model-Driven Development of Web Applications with UWA, MVC and JavaServer Faces. In Baresi, L., Fraternali, P., Houben, G.J., eds.: ICWE. Volume 4607 of Lecture Notes in Computer Science., Springer (2007) 457–472
19. Stefano Ceri and Piero Fraternali and Aldo Bongio: Web Modeling Language (WebML): a modeling language for designing Web sites. Comput. Netw. **33**(1-6) (2000) 137–157
20. Daniel Schwabe and Gustavo Rossi: An Object Oriented Approach to Web-Based Application Design. Theor. Pract. Object Syst. **4**(4) (1998) 207–225
21. Natacha Güell and Daniel Schwabe and Patricia Vilain: Modeling Interactions and Navigation in Web Applications. In: ER '00: Proceedings of the Workshops on Conceptual Modeling Approaches for E-Business and The World Wide Web and Conceptual Modeling, London, UK, Springer-Verlag (2000) 115–127
22. Orbeon: Orbeon Forms User Guide - Page Flow Controller (May 2008)
Available online at http://www.orbeon.com/ops/doc/reference-page-flow.
23. Zhang, G., Baumeister, H., Koch, N., Knapp, A.: Aspect-Oriented Modeling of Access Control in Web Applications. In: Proc. 6th Int. Wsh. Aspect Oriented Modeling (WAOM'05), Chicago (2005)
24. Christian Kroiss and Nora Koch: UWE Metamodel and Profile: User Guide and Reference. (2008) LMU Technical Report.
25. Bammigatti, P.H. and Rao, P.R.: GenericWA-RBAC: Role Based Access Control Model for Web Applications. Information Technology, 2006. ICIT '06. 9th International Conference on (18-21 Dec. 2006) 237–240