UNIVERSITY OF TARTU
Institute of Computer Science
Cyber Security Curriculum

Andrey Sergeev

Role Based Access Control as SecureUML Model in Web Applications Development with Spring Security

Master's Thesis (30 ECTS)

Supervisor(s): Raimundas Matulevičius

Tartu 2016

Role Based Access Control as SecureUML Model in Web Applications Development with Spring Security

Abstract:

Nowadays fast and successful development of a web application is one of the keys to effective business. However, modern world requirements define the complex approach in definition of access control and user groups' interoperability. The software development process typically involves different responsible members for the application assessment, planning, development, deployment and support, as a consequence, increasing the complexity and information losses between target groups. In order to mitigate possible risks in software development misinterpretation and security violation, teams should use tools that allow fast and accurate interpretation of the web application through a model. Modelling will help with minimization of possible problems and ensure the functionality needs with respect to desired RBAC model. In order to support and simplify the model-driven approach for a web application development with Spring platform, realization of a concept plugin for Eclipse IDE is proposed. This plugin supports the recognition of Spring Security notations with capability to visualize the RBAC model on top of them. The generation of visual model is achieved in two main steps: recognition of Spring Security configuration and generation of representation with SecureUML modeling language. The concept of contributed plugin was validated within case studies that demonstrated the acceptance of this plugin by software developers due to its integrated solution for faster development and help in understanding of RBAC model for the selected web application.

Keywords:

RBAC, SecureUML, Eclipse plugin, Spring MVC, Spring Framework, Spring Security

CERCS: T120 Systems engineering, computer technology.

Rollipõhine juurdepääsukontroll kui SecureUML mudel veebirakenduste arenduses kasutades Spring Security platvormi

Lühikokkuvõte:

Tänapäeval on efektiivse äri üheks võtmeks kiire ja edukas veebirakenduste arendus. Samas, uudsed maailmas levinud nõuded eeldavad keerukat lähenemist juurdepääsukontrolli ja kasutajate rühmade koostöövõime määratlemisel. Tavapäraselt hõlmab tarkvara arenduse protsess erinevaid vastutavaid osalisi rakenduse hindamisel, plaanide koostamisel, arendusel, rakendamisel ja kasutajatoe tagamisel, mille tulemusena suureneb informatsiooni kadu ja keerukus sihtgruppide vahelises suhtluses. Arendusmeeskonnad peaksid sellise olukorra vältimiseks ja väärtõlgendustest ning turvalisuse nõuete rikkumisest tulenevate võimalike riskide leevendamiseks tarkvara arenduses kasutama vahendeid, mis võimaldavad kiiret ja täpset veebirakenduse interpreteerimist läbi mudeli. Modelleerimine aitab tunduvalt vähendada võimalikke probleeme ja tagab funktsionaalsuse vajadused arvestades soovitud rollipõhise juurdepääsukontrolli mudeliga. Antud töös pakutakse välja kontseptuaalne Eclipse IDE lisandmooduli realisatsioon, et toetada ja lihtsustada mudelil baseeruvat lähenemist veebirakenduste arendamisel kasutades Spring platvormi. Loodud lisandmoodul toetab Spring Security esitusviiside tuvastamist koos võimega visualiseerida nende üle rollipõhise juurdepääsukontrolli mudelit. Visuaalse mudeli genereerimine toimub kahe peamise astmena: Spring Security konfiguratsiooni tuvastamine ja esituse genereerimine kasutades SecureUML modelleerimise keelt. Loodud lisandmooduli kontseptsioon valideeriti juhtumiuuringutega, mis näitasid lisandmooduli sobivust tarkvara arendajate jaoks tänu integreeritud lahendusele, et tagada kiiremat arendust ja abi valitud veebirakenduse rollipõhise juurdepääsukontrolli mudelist arusaamisel.

Võtmesõnad:

RBAC, SecureUML, Eclipse plugin, Spring MVC, Spring Framework, Spring Security

CERCS: T120 Süsteemitehnoloogia, arvutitehnoloogia.

Table of Contents

1 Introduction

Nowadays, cyber-attacks are on the way to become almost a common event of our lives and reality [1]. Attacks are targeting different software vulnerabilities on the application layer and bring concerns about software quality and possible measures to react. However, it is difficult to improve and address these vulnerabilities because of software complexity and huge number of developers participating in applications creation. Although, the security ultimately depends on the general quality of a software product, having only working application is not enough today; in addition it has to be secure.

The modern world requirements define the complex approach in definition of access control and user groups' interoperability, in particular, bringing additional problems with security and its configuration. Typical vulnerabilities or implementation patterns causing weaknesses are well-known, for example, The Open Web Application Security Project (OWASP) publishes a list of common flaws, called the OWASP top ten [2]. Nevertheless, the top 10 remains active with insignificant changes from year to year. Moreover, checking the OWASP top ten [2] risks more precisely: A2, A4, A7, A8 and A10 – we see the mitigation strategy is in proper validation and allocation of a role. However, at the same time OWASP [3] defines the term of Anti-Pattern for roles – "Hard Coded Roles" that spread the issues: Difficulties to audit and test; Update of the code in case of the access control policy changes; simplicity of making a mistake in configuration. We found these terms are related. In order to address this issue and offer the solution we determine one of the most popular frameworks [4] [5] [6] for the web development is Spring [7].

In order to offer a solution that helps solving the fact and need for better understanding of security and better secured software we state the research questions:

RQ1: How is it possible to support a Spring web application and improve the understanding of its usage?

RQ2: How to validate an existing Spring web application for the roles configuration?

RQ3: How is it possible to support the software developer and help him/her to understand the security of a Spring web application?

To solve these questions, we should organize the support for software developers at the early stage of the software development life cycle; and supply them with proper tools allowing to review the application and to identify the insufficiently secure code. On the other hand, introduction of a completely new tool for a developer may bring the annoying effect of usage of different environments and tools for different purposes, thus, the more beneficial way to solve this problem in integration of our tool into existing tools of software development. Another important aspect of the solution is in the way of presentation of a web application; therefore, we should visualize it using the modeling that could represent the required things in a proper and clear way.

The thesis is organized in five chapters. In chapter 2 we introduce the theoretical and technological prerequisites for development of this thesis. Chapter 3 presents the overview and brief outline of a software contribution concept that was designed and implemented to offer a solution for the stated problems. Chapter 4 introduces the validation researches and analysis confirming the acceptance and perspectives of this software contribution. Finally, in chapter 5 we discuss the findings, give an overview of the related work, conclude this thesis and define directions for future work and research.

2 Background

This chapter gives an overview about theoretical and technological prerequisites for the thesis topic. First, we present the main concept of access policy definition and its compatible modeling language. Second, we visualize the Role Based Access Control levels by means of selected modeling language. Third, we describe technologies and frameworks used for implementation of software contribution. Finally, we introduce the empirical research methods to implement for validation of thesis contribution.

2.1 Role Based Access Control and Modeling

Access control is a security service responsible for defining which subjects can perform what type of operations on which objects. A subject is typically an active entity such as a user or a process, and an object is an entity, such as a file, database table or a field, on which the subject can perform some authorized operations. Permission indicates the mode of operation on a particular object [8].

Role is a prevalent organizational concept and it identifies the various job functions and responsibilities within an organization. As organizational information systems has become a crucial component for carrying out organizational job functions, it has motivated the use of roles that users within the organization play to define what accesses should be authorized to them so that they can carry out their job functions and responsibilities efficiently. Based on the premise that the role represents the set of permissions that are needed for carrying out the job functions, various Role Based Access Control (RBAC) approaches have been proposed [8].

2.1.1 RBAC Family

Modern information systems are required to ensure the appropriate level of access to resources in respect of user's position and job duties. There also has to be a sufficient protection against accidental and intentional security breaches. When choosing an appropriate access control policy, organizations have to find a balance between implementation costs and security. Role Based Access Control provides a clear pattern and affordable way to achieve this balance and nowadays, it has become the predominant model for advanced access control foremost because of its simplicity and reduced administration costs [9].

In RBAC, users are assigned to roles and roles are associated with permissions. Permission determines what operations a user assigned to a role can perform on information resources. Essential RBAC principle includes the requirement that user-role and permission-role assignment can be many-to-many. Thus, the same user can be assigned to many roles and a single role can include many users. Similarly, for permissions, a single permission can be assigned to many roles and a single role can be assigned to many permissions. In addition to user, roles and permissions, various kinds of constraints can be specified in RBAC. Evolution and research of RBAC initial definitions – a.k.a. *Flat RBAC* – produced concepts of *Hierarchical RBAC*, *Constrained RBAC* and *Symmetric RBAC* that are defined and standardized in the NIST[1] model [10]. The overview on key RBAC requirements depending on the RBAC type is illustrated in the Table 1 [10] [11].

Table 1. The NIST model of RBAC generations and requirements (adapted from [3]).

Level	Name	RBAC Functional Capabilities
1	*Flat RBAC*	• users acquire permissions through roles • must support many-to-many user-role assignment • must support many-to-many permission-role assignment • must support user-role assignment review • users can use permissions of multiple roles simultaneously
2	*Hierarchical RBAC*	In addition to *Flat RBAC*: • must support role hierarchy (partial order) • level 2a requires support for arbitrary hierarchies • level 2b denotes support for limited hierarchies
3	*Constrained RBAC*	In addition to *Hierarchical RBAC*: • must enforce separation of duties (SOD) • level 3a requires support for arbitrary hierarchies • level 3b denotes support for limited hierarchies
4	*Symmetric RBAC*	In addition to *Constrained RBAC*: • must support permission-role review with performance effectively comparable to user-role review • level 4a requires support for arbitrary hierarchies • level 4b denotes support for limited hierarchies

The *Hierarchical RBAC* adds a requirement for supporting role hierarchies. A hierarchy is mathematically a partial order defining a seniority relation between roles, whereby senior roles acquire the permissions of the juniors. In other words, role r_1 inherits role r_2 only if all permissions of r_2 are also permissions of r_1 and all users of r_1 are also users of r_2. The NIST model [10] recognizes two sub-levels in the hierarchy definition: *General Hierarchical RBAC* – In this case there is support for an arbitrary partial order to serve as the role hierarchy; and *Restricted Hierarchical RBAC* – Some systems may impose restrictions on the role hierarchy. Most commonly, hierarchies are limited to simple structures such as trees or inverted trees [10] [11].

[1] http://www.nist.gov

The *Constrained RBAC* in addition to *Hierarchical RBAC* requires for constraints including static separation of duty (SSOD), dynamic separation of duty (DSOD), prerequisite roles and cardinality constraints. The enforcement for separation of duties (SOD) is a time-honored technique for reducing the possibility of fraud and accidental damage, known and practiced long before the existence of computers. SOD spreads responsibility and authority for an action or task over multiple users, thereby raising the risk involved in committing a fraudulent act by requiring the involvement of more than one individual. The RBAC respects SSOD – based on user-role assignment and DSOD – based on role activation during the same session. The concept of prerequisite roles is based on competency and appropriateness. Prerequisite constraints require that a user can be assigned to a role only if the user is already assigned to the role's prerequisites. Another constraint type is cardinality constraints. Cardinality constraints can be used to restrict, for example, the number of users that can be assigned to a role, the number of roles a user can play, the number of roles permission can be assigned to, or the number of sessions a user is allowed to activate at the same time [10] [11].

The *Symmetric RBAC* extends the requirements of *Constrained RBAC* by adding a necessity for permission-role review similar to user-role review introduced in level. Thus, the roles to which a particular permission is assigned can be determined as well as permissions assigned to a specific role. The performance of permission-role review must be effectively comparable to that of user-role review. This requirement of *Symmetric RBAC* comes from the need of role transfer or substitution. For example, by identifying permissions of a single user, the manager of policy should have possibilities to revoke all permissions of this user and then to reassign all revoked rights to another user with same or different set of permissions [10].

2.1.2 SecureUML

There are several advantages to integrating security engineering in the software development lifecycle. To begin with, this approach allows security requirements to be incorporated into system designs at a high level of abstraction. This further facilitates the development of security aware applications that avoid the violation of security policies in respect of work duties and needs. Moreover, the utilization of access control infrastructure model can prevent errors during the implementation of access control requirements and enable the technology abstracted design of a secure application [12].

The unified modeling language (UML)[2] is a standard modeling language maintained by the Object Management Group (OMG)[3]. The UML defines notations for building many diagrams that each presents a particular view of the artifact being modeled [11]. SecureUML defines the construct of notations to UML-based models with information relevant to access control, expressing different aspects of access control, like roles, role permissions and user-role assignments. Due to its general access-control model, extensibility, visual notation and the possibility to define designs at a high abstraction level, SecureUML is well suited for designing secure systems, business analysis as well as design models for different stack of technologies. Therefore, SecureUML enables even develop-

[2] http://www.uml.org
[3] http://www.omg.org

ers without a strong security background to develop secure systems implementing the model definition [12] [13].

SecureUML is based on the extension of RBAC model, allowing support expression of access control conditions coupled with system conditions, by introduction of authorization constraints in SecureUML. An authorization constraint defines the precondition for granting access to an action. SecureUML offers a significant design benefit combining the simplicity of graphical notation for RBAC with the power of logical constraints on models. Policies with different level of complexity can be expressed using role-based permissions and, if needed, by adding effective authorization constraints. Access control infrastructure can be generated from SecureUML models and thereby prevent possible errors during the development and implementation of access control policies and enable the technology independent creation of secure systems [12] [13].

Typical workflow for SecureUML model creation is [12]:

- Identify Users;
- Identify application roles;
- Map users into roles;
- Identify resources;
- Identify actions;
- Identify authorization constraints;
- Account for cardinality and complex relations (inheritance);
- Combine relevant diagrams to document security policy.

The SecureUML metamodeling (shown in Figure 1) is defined as an extension of the UML metamodeling.

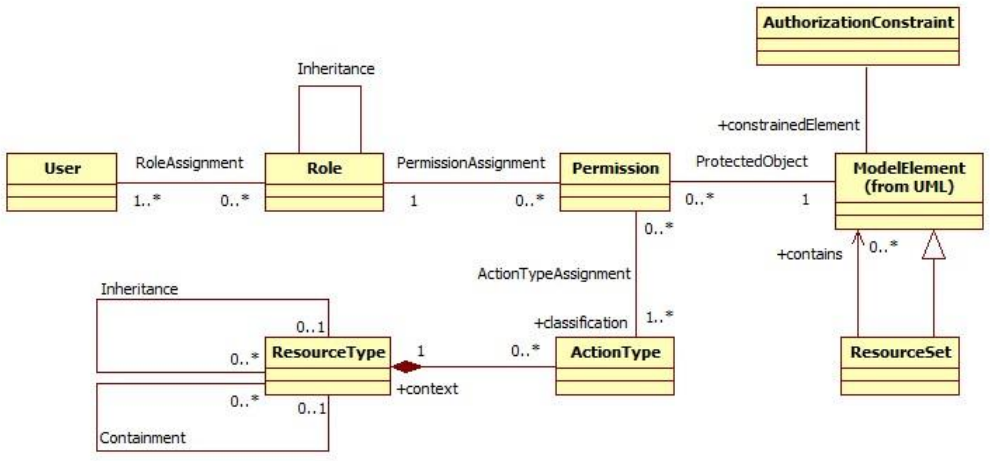


Figure 1. SecureUML Metamodel (adapted from [13]).

The concepts of RBAC are represented directly as metamodeling types. SecureUML introduces the new metamodeling types and relations between them: *User*, *Role* and *Permission*. Protected resources are represented in a different way. Instead of defining a dedicated metamodeling type to represent them, SecureUML allows every UML model element to take the role of a protected resource. Additionally, SecureUML introduces the type *ResourceSet*, which represents a user defined set of model elements used to define permissions or authorization constraints. Permission is a relation object connecting a role to a *ModelElement* or a *ResourceSet*. The semantics of permission is defined by the *Action-*

Type elements used to classify the permission. Every *ActionType* represents a class of security relevant operations on a particular type of protected resource. In SecureUML, there is a corresponding action type for every class of such actions. Action types may also represent more conceptual classes of operations at a higher abstraction level [12].

2.2 Using SecureUML to Model RBAC

In order to prove the conformity and compatibility of SecureUML with different RBAC levels we use an example web application having different access constraints, roles and activities. The selected example should demonstrate the abilities to model and visualize the implementation of Role Based Access Control mechanisms and define initial requirements for software contribution and its validation. The selection for task demonstration is a part of forum software allowing users to share and exchange information by creating threads and topics. The common forum has users with various responsibilities and privileges. For example, the hierarchy of forum users might be the following:

- Administrators are in charge for forum's sections (threads), their definition and management;
- Moderators are for forum's sub-sections (topics), their management, content formation and validation;
- Users are typically the biggest target group of a forum application, they are consumers and actors of forum's knowledge base with abilities to propose the information and to evaluate its quality;
- Guests are all unregistered users with limited privileges or without access rights, depending on the forum's target community.

However, the example of forum application for this chapter is minified to achieve better illustration of RBAC rather than expose possible complexity of the software. The conditions for access control escalate from simple model to complex one in order to demonstrate the evolution of RBAC level requirements.

First of all, we identify possible actions over forum application shown in Figure 2.

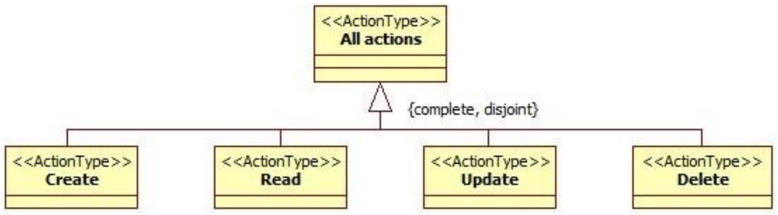


Figure 2. Security action types in forum application.

It is possible to identify four prime actions on an abstract resource: Create – creation of resource component in the system; Read – an ability to use resource content for reading; Update – a capability to modify the content of resource data; Delete – a possibility to remove the resource from the system.

2.2.1 Flat RBAC

The initial requirement contains the need of multiple users and roles having permissions. Relations and access dependencies between resources and roles are shown in Figure 3. First, we determine possible users: Adam, Alice, Bob and David. Next, we identify roles for our application. So, Administrator and Guest roles are selected: Administrator represents the role inside organization with authentication required; while, Guest *Role* stands for public usage of forum's *Resource*, this *Role* is assigned to every visitor of the forum application by default. As a result, Adam and Alice assigned to Administrator *Role* after success in authorization procedure. Bob and David are not registered or use the forum anonymously without authentication; nevertheless, they are able to participate in forum activities with public access having Guest *Role*. *Resources* include the objects of Forum Thread and Forum Topic with their attributes and operations. Depending on the *Roles*, we have different set of allowed actions defined by *Permission*. Guest *Permission* allows only read access (`readThread()` and `readTopic()`) on forum *Resources* Forum Thread and Forum Topic. While Administrator *Permission* gives additional possibilities – creation (`createThread()` and `createTopic()`), update (`updateThread()` and `updateTopic()`) and deletion (`deleteThread()` and `deleteTopic()`) of *Resources* in addition to reading.

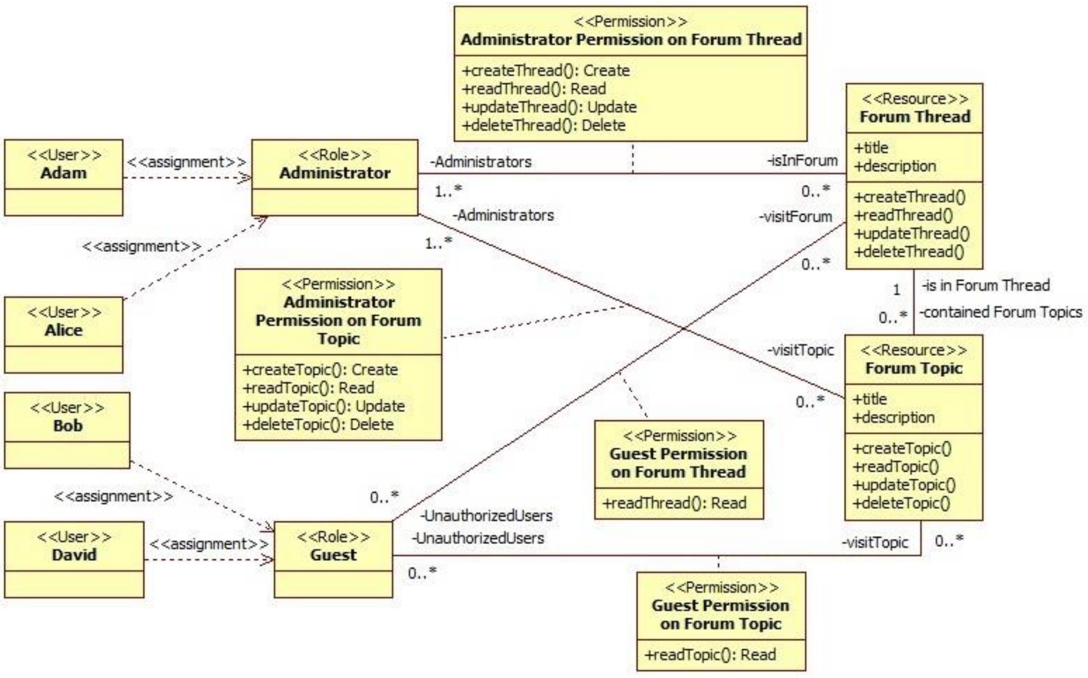


Figure 3. An example SecureUML model of *Flat RBAC*.

2.2.2 General Hierarchical RBAC

General Hierarchical RBAC gives us advantages of usage Role hierarchies shown in Figure 4. In comparison to *Flat RBAC*, the *General Hierarchical RBAC* includes all principles of *Flat RBAC*, and in addition allows the consumption of another *Role's Permission*. The Figure 4 demonstrates the hierarchy of roles – Administrator *Role* implies the Guest *Role*, thus the *Permission* of Administrator *Role* does not require for separate definition of read access on Forum Thread and Forum Topic objects, as it is occurred from Guest *Role*. In practice, in scope of this example, for software implementation it means the public access for reading for all type of *Roles*, thus does not require implementation of security mechanisms for read operations.

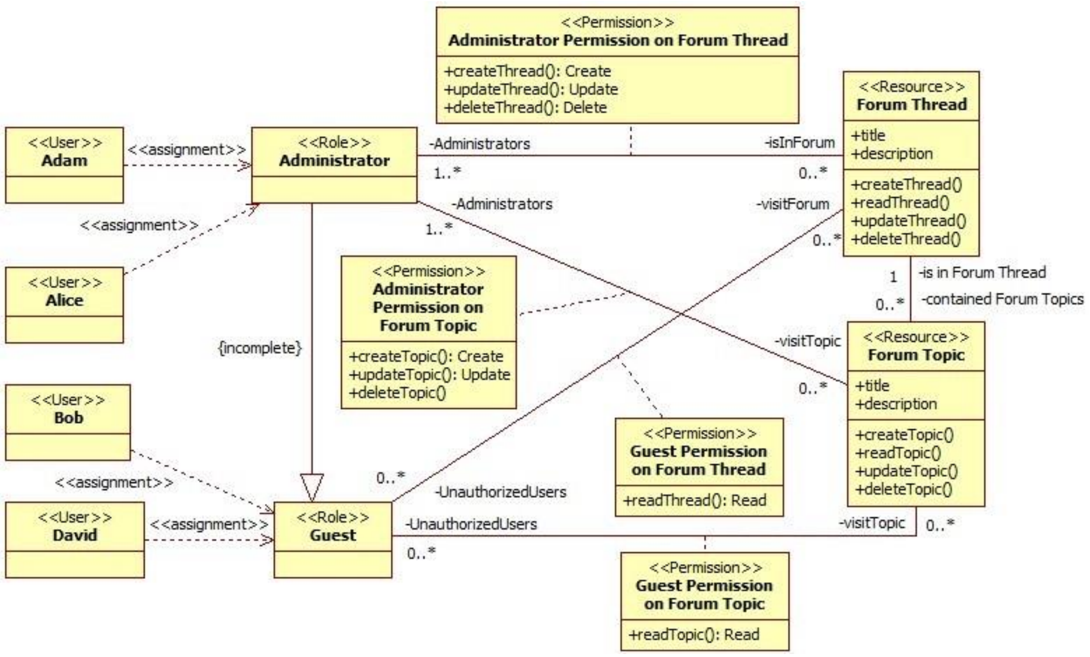


Figure 4. An example SecureUML model of *General Hierarchical RBAC*.

2.2.3 Constrained RBAC

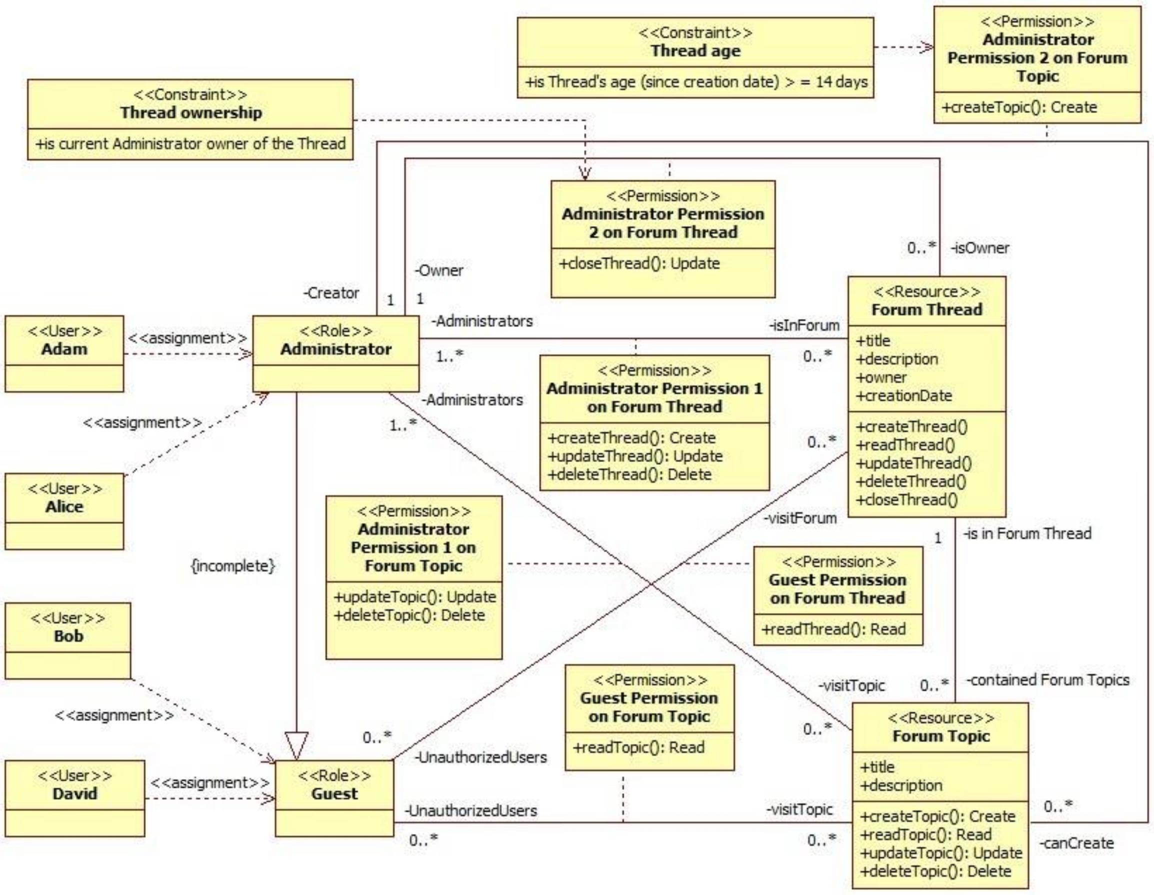


Figure 5. An example SecureUML model of *Constrained RBAC*.

The *Constrained RBAC* extends the abilities of *General Hierarchical RBAC* and brings the concept of separation of duties (SOD). In order to demonstrate this approach, we add extra attributes to Forum Thread *Resource* – `owner`, `isOpen`, `creationDate`; and operation `closeThread()`, see Figure 5. The addition of new operation `closeThread()` comes with Thread ownership *Constraint* and forms new *Permission* – Administrator *Permission* 2 on Forum Thread. The functionality requirement behind *Constraint* is in possibility to end submission of Forum Topics dedicated to specific Forum Thread by setting its attribute `isOpen` to false. However, this possibility is not allowed to all Administrators, but only to initial author of the Forum Thread, who sets the Ownership over Thread at its creation time (attribute `owner`) and acquires the possibility to update the Forum Thread by closing it. Another *Constraint* is Thread age; it defines the pre-requirement for Forum Topic creation. This constraint sets the precondition by adding requirement for date comparison between current date and Forum Thread's creation date. If the difference is less or equal to 14 days, the creation of Forum Topic is allowed for Administrator, otherwise, the creation is not allowed.

2.2.4 Symmetric RBAC

The *Symmetric RBAC* appends requirements to *Constrained RBAC* for review and determination of role-permission relations.

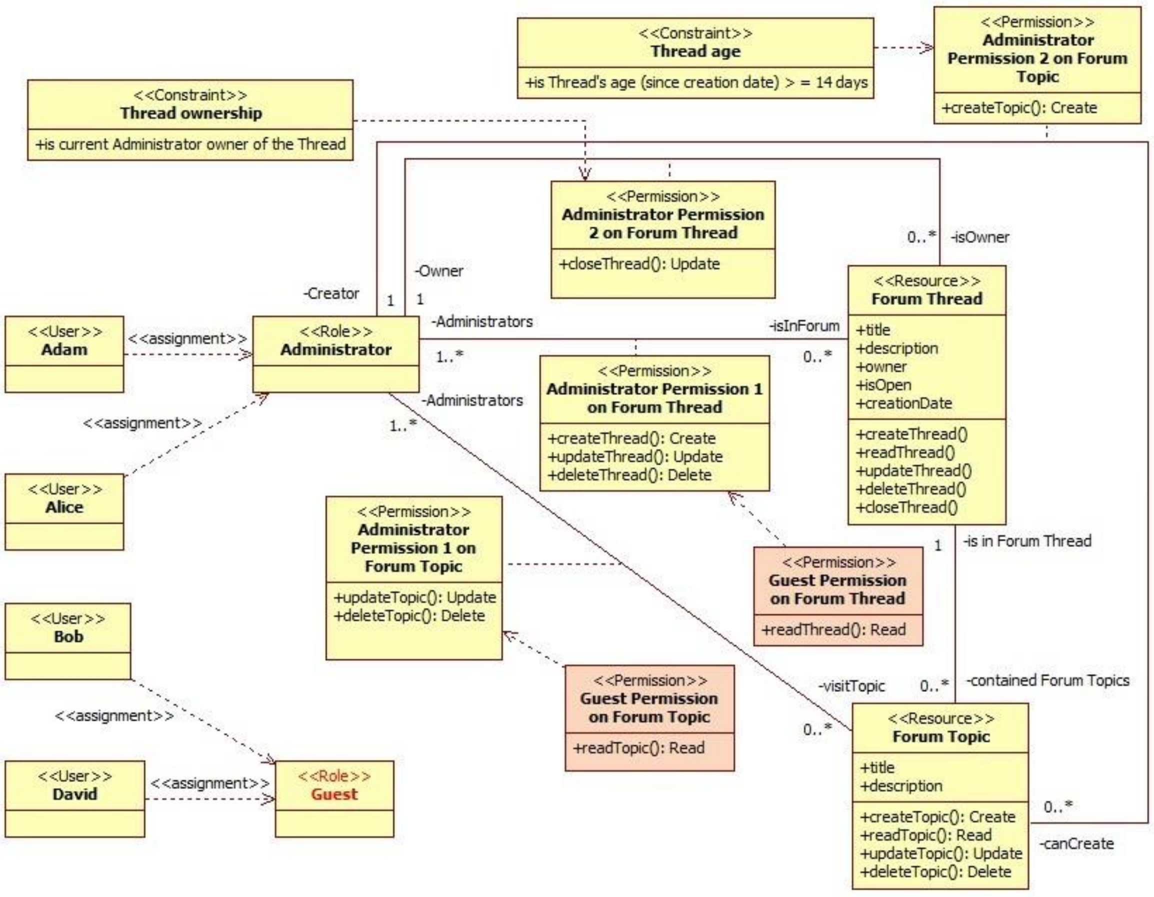


Figure 6. An example SecureUML model of *Symmetric RBAC* with role-permission removal preview.

In order to illustrate this requirement we remove the Guest *Role*, as the stakeholder of forum application, may force all users of this forum to have registration within forum system and end the public access. The middle-step for role-permission of *Rol*e removal is illustrated in Figure 6. The Figure 6 demonstrates the remaining dependencies of Guest *Permissions* upon removal of relations between Guest *Role* and *Resources*. The determination of two remaining permissions Guest *Permission* on Forum Thread and Guest *Permission* on Forum Topic should be merged into upper level of *Resource Permissions.* The delegation of *Permissions* takes into account the hierarchy between Administrator and Guest *Roles*. In order for Administrator *Role* to function as before removal of Guest *Role*, Guest *Permissions* have to be merged into proper *Permissions* of denoted *Resources*. The final result of Guest *Role* removal is demonstrated in Figure 7. Within Guest *Role* removal the presence of *Users* Bob and David become obsolete, unless they are not assigned to any valid *Role*. *Permissions* of the removed *Role* that were shared between Guest and Administrator *Roles*, moved to Administrator *Permission* on Forum Thread and Administrator *Permission* on Forum Topic.

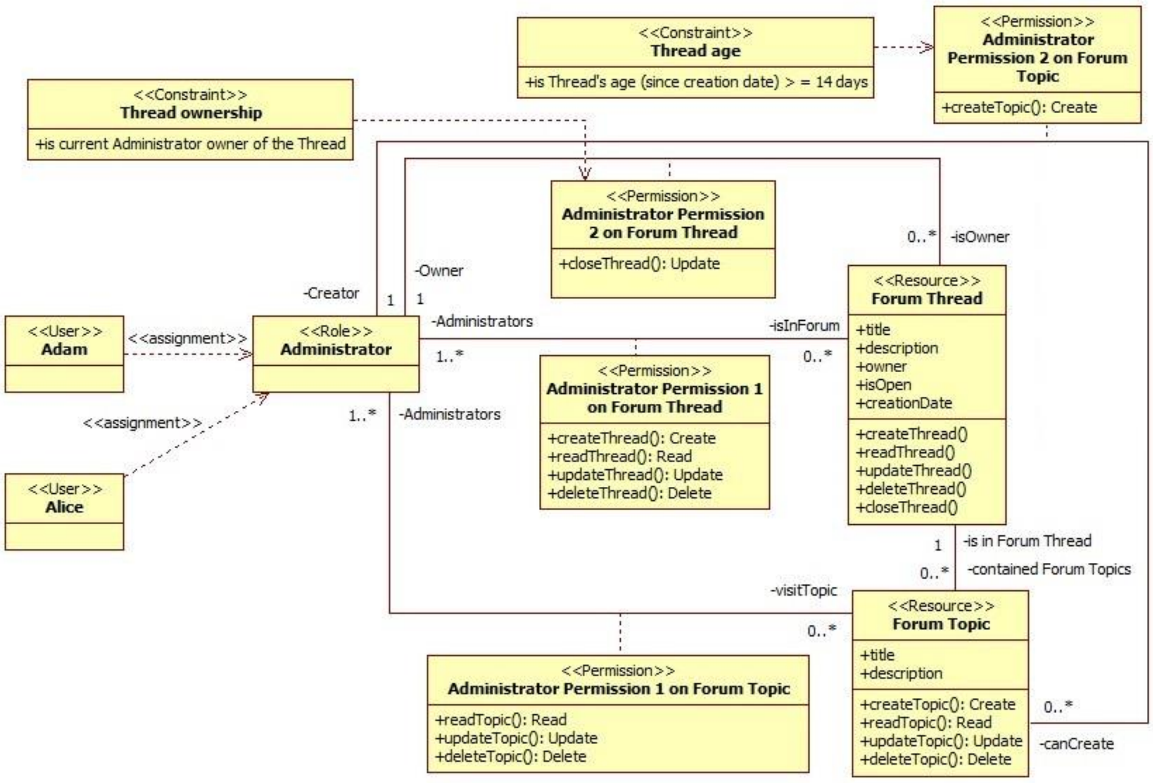


Figure 7. An example SecureUML model of *Symmetric RBAC* with Guest role-permission removed.

2.3 Technology

The concept of a model based approach for software design and development appeared at the end of the nineties. The definition of the UML and its release by the OMG in 1997 caused the interest in modeling of software development communities. The OMG initiative of Model Driven Architecture (MDA)[4] brought a lot of expectations as the Object oriented Programming (OOP) was showing its limitations in terms of optimization. Its universal and abstract principles were to make use of patterns in order to easily implement the same model for different versions and platforms. Furthermore, these terms and definition formed the principles of Model-Driven Engineering (MDE). MDE is the software engineering approach that is built around the concept that everything is a model. The benefits of modeling languages, especially UML, drove the development of MDE techniques as well as development of sophisticated tools to support the automated generation of code and its visualization. One of the most important motivations for applying MDE techniques are in improved software correctness and transparency [14] [15].

As a result of MDE, the Model-Driven Security (MDS) has emerged more than a decade ago as a specialized topic in the MDE approach for supporting the development of security requirements. MDS applies the modeling paradigm to information security engineering, bringing several benefits to the domain. First, MDS integrates the security concerns explicitly from the beginning and throughout the development lifecycle. The combined outcome of MDE and MDS approaches deliver a complete system implementation, not only the functionality or security limitations. Second, MDS may inherit the abstraction principles of MDA and concentrate on business functionality and security requirement promoting the platform independence as well as cross-platform interoperability. This approach allows security experts to focus on security related issues, instead of dealing with the technical problems of target system infrastructure. Finally, MDS relies on MDE automation reducing the human interference, which is naturally error-prone [15] [16].

2.3.1 Spring Framework

The Spring Framework started from the code written for the book Expert One-on-One J2EE Design and Development by Rod Johnson [17]. This publication demonstrated and explained some of the existing complexities in Java EE. This book offered the possible solution to overcome difficulties and stimulated the ongoing evolution of Java EE technologies [18] [19].

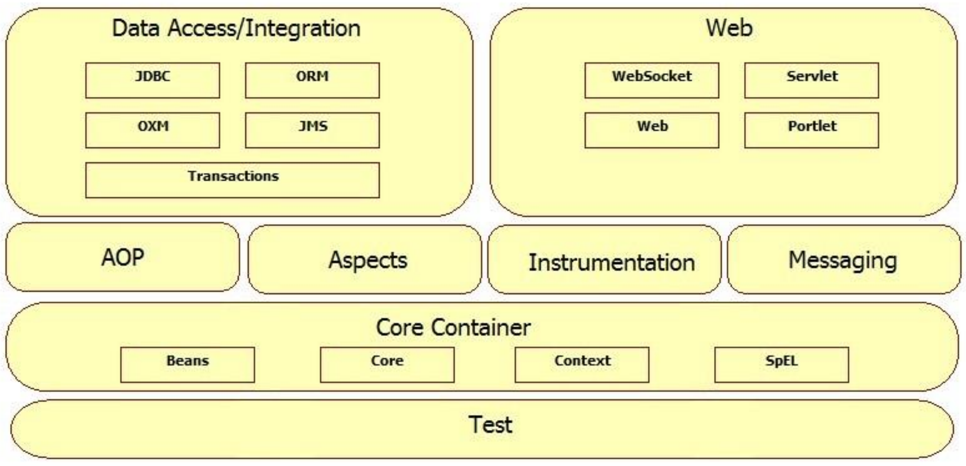


Figure 8. Spring Framework organization (adapted from [20]).

[4] http://www.omg.org/mda/

Next, the Spring Framework has emerged as a popular choice for the middle, or business tier of enterprise application development. Its lightweight paradigm for developing enterprise applications offered significant benefits over standard Java EE platform development [18] [19].

Nowadays, the Spring Framework is an open source application framework and inversion of control container for the Java platform. The framework's core features can be used by any Java application, but there are extensions for building web applications on top of the Java EE platform. The Spring Framework consists of features organized into about 20 modules as shown in the Figure 8 [20] [7].

These modules (Figure 8) are grouped into Core Container, Data Access/Integration, Web, AOP (Aspect Oriented Programming), Instrumentation, Messaging, and Test providing the infrastructure for application development and integration. Finally, the Spring Framework offers consistent programming extensions for the huge stack of popular technologies, allowing developers to target discrete problems and build solutions specifically for them [19].

2.3.2 Spring Security

Spring Security is a production-ready framework that allows implementing and customizing the authentication and access-control mechanisms of an application. It is dedicated to provide a full array of security services to Java applications in a flexible and developer-friendly way. Although the majority of applications that use the framework are web based, the core of Spring Security can also be used in standalone applications [21] [22].

The modular framework consists of loosely coupled components, which are connected using dependency injection. The core classes and their dependencies are shown in Figure 9. The `Authentication` class stores user information. It is a part of a `SecurityContext` class for every authenticated user in an application. An `AuthenticationManager` loads this data and which verifies the authenticity of users using offered credentials and information from a user store. To intercept secured resource access, classes extend the `AbstractSecurityInterceptor` class, which is the central class in terms of authorization. Thereby, the `SecurityContext` and `SecurityMetadataSource` classes offer information about the current user and the secured object respectively. Access decisions are performed by the `AccessDecisionManager`, which is also called by the `AbstractSecurityInterceptor`. The `AccessDecisionManager` calls voters, which decide whether access is granted or not and which can be added dynamically to the application. Thus, the voter system abstracts from an access control mechanism [23].

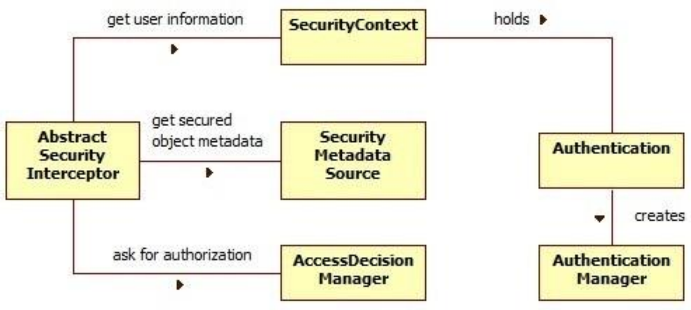


Figure 9. The main classes of the Spring Security (adapted from [23]).

Finally, Spring Security fills a gap in the universe of Java third-party libraries integration. Standards such as Java Authentication and Authorization Service (JAAS), or Java EE Security do provide some ways of performing similar authentication and authorization functions, but Spring Security offers a 'ready to configure' integration with many common enterprise authentication systems. This advantage makes Spring Security adaptable to most situations with little effort (beyond configuration) on the part of the developer [22] [23].

2.3.3 Eclipse

“The Eclipse Project was originally created by IBM in November 2001 and supported by a consortium of software vendors. The Eclipse Foundation was created in January 2004 as an independent not-for-profit corporation to act as the steward of the Eclipse community. The independent not-for-profit corporation was created to allow a vendor neutral and open, transparent community to be established around Eclipse. Today, the Eclipse community consists of individuals and organizations from a cross section of the software industry.” [24]

The purpose of Eclipse is to provide a highly integrated core tool platform. This framework itself divided into four main subprojects: Equinox, the Platform, the Java Development Tools (JDT), and the Plugin Development Environment (PDE) illustrated in Figure 10. Collectively, the four subprojects provide everything needed to extend the framework and develop tools based on Eclipse using Eclipse Software Development Kit. Other projects extend the core framework to support specific kinds of tools and development environments via plugins. The projects in Eclipse are implemented in Java [25].

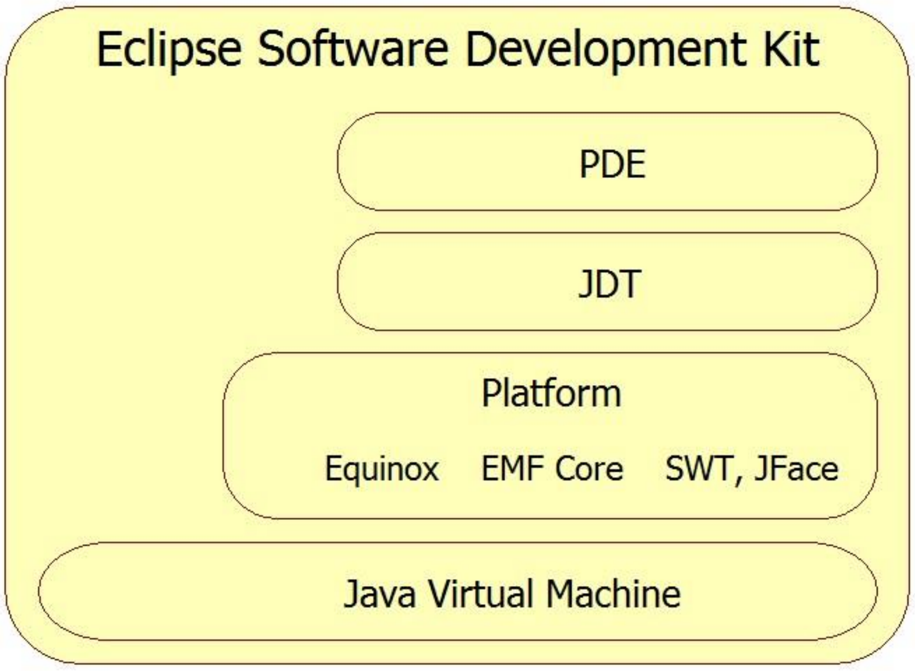


Figure 10. The overview of Eclipse Software Development Kit architecture (adapted[5]).

[5] https://wiki.eclipse.org/Eclipse4

In Eclipse, the basic unit of function or component is called a plugin. The Eclipse Platform itself and the tools that extend it are both composed of plugins. A simple tool might consist of a single plugin, but more complex tools are typically divided into several. Each plugin contributes functionality that can be invoked by the user or reused and extended by other plugins [21].

The Eclipse plugins can be divided into 3 main categories: Core plugins (ECP), Extension plugins (EEP) and Third-party plugins (ETP). ECPs are plugins present in and shipped as part of the Eclipse Software Development Kit (SDK). This essential collection of plugins provides core functionality upon which other plugin extensions are built. The plugin also provide the runtime environment in which other plugins are loaded, integrated, and executed. EEPs are plugins built with the main goal of extending the Eclipse platform. Most EEPs are large, generic, applications frameworks with tool plugins to build other specialized applications. Finally, ETPs are the remaining plugins. The common scenario for ETPs to reuse at least some functionality provided by the ECPs but also may use functionality provided by EEPs [22].

The JDT is the core part of Eclipse; it provides the full-function Java development environment built using Eclipse. Its tools are highly integrated and representative of the power of the Eclipse Platform. It can be used to develop Java programs for Eclipse or other target platforms and applications. The JDT provides tools for java model extraction and recognition, representing a source code as structured element tree. The Java element tree defines a project oriented view of the java files, dependencies and content of internal constructs like package fragments, compilation units, binary classes, types, methods and fields [26] [21].

2.3.4 Eclipse Modeling

The Eclipse Modeling Project is one of the useful services and extensions provided by Eclipse Foundation. The product of this project focuses on the evolution and promotion of model-based development technologies within the Eclipse community. This extension supplies a unified set of modeling frameworks, tooling and standards implementations for the creation of textual and graphical editors. The Eclipse Modeling Project consists of three major Eclipse plugins [27] [28]:

- Eclipse Modeling Framework (EMF) [29];
- Graphical Modeling Framework (GMF) [30];
- Graphical Editing Framework (GEF) [31].

“The EMF is a modeling framework and code generation facility for building tools and other applications based on a structured data model. From a model specification described in XML Metadata Interchange (XMI)[6], EMF provides tools and runtime support to produce a set of Java classes for the model, along with a set of adapter classes that enable viewing and command-based editing of the model and a basic editor.” [29]

The GMF is a framework for developing domain specific languages (DSL). GMF is built on the Eclipse Modeling Framework (EMF) and the Graphical Editing Framework (GEF). The domain model often needs to be visualized in a graphical way, such as in a diagram editor. This is especially important when implementing tools and models representations.

[6] http://www.omg.org/spec/XMI/

The following frameworks provide support to implement graphical views for the EMF-based domain model that can be embedded into a tool or an application [30].

The GEF is a framework that allows developers to take an existing application model and simplifies the creation of a rich graphical editor for it. It can be hooked to EMF Metamodel. GEF is a framework that supports the development of graphical editors. In the context of GMF, GEF is used to implement the concrete graphical syntax of languages and for editing of concrete syntax [27] [31].

2.3.5 Obeo Designer

The implementation of editors and representations via pure Eclipse Modeling set of plugins is complicated task. Therefore, the Obeo Designer[7] was used to design artifacts, their interconnections and their representation for SecureUML diagrams. This tool provides a setting environment to configure viewpoints and their various representations. For each, it is possible to define display criteria and their graphic features, styles, tool palettes, and their associated behaviors. The graphical aspects are dissociated from behavior [27].

The Obeo Designer is based on the EMF, GEF and GMF frameworks that independently offer sufficient collection of elements to build modelers. However, the difference in indirect usage of Eclipse Modeling Project is in upper layer allowing specifying modelling without deep expertise and coding of complicated frameworks like EMF, GEF and GMF. The output of Obeo Designer model design is .odesign file representing the Domain Specific Model (DSM). This style sheet file allows the definition of several kinds of diagrams and point of views of the model input data. Each point of view may include several kinds of diagram elements, their relations and focus on different view concerns [14] [27].

2.3.6 Eclipse Sirius

The Sirius is an Eclipse project allowing creating graphical modeling workbench.

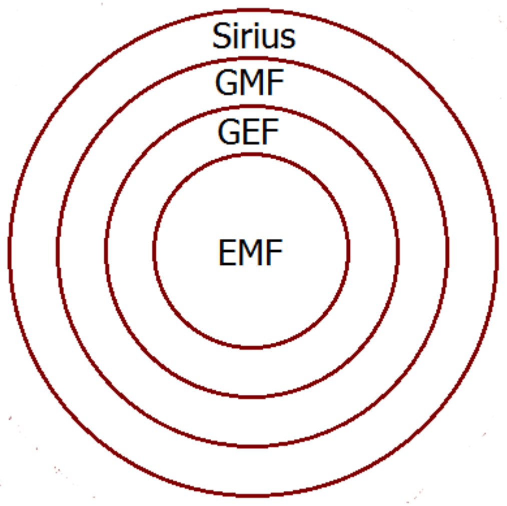


Figure 11. Hierarchy of Graphical Model-Driven Engineering Tools (adapted from [32]).

[7] http://www.obeodesigner.com/

It unifies the Eclipse Modeling technologies, including EMF, GEF and GMF as illustrated in Figure 11. This workbench is based on the viewpoints definition specified via DSM file compatible with .odesign file. Initially Sirius supports three different kinds of representations: diagrams (graphical modelers), tables and trees (hierarchical representations). However, new representations can be added through programming. The editors are defined by the DSM file which defines the complete structure of the modeling workbench, its behavior, editors and navigation tools. This description of a Sirius modeling workbench is dynamically interpreted by a runtime within the Eclipse. Furthermore, the runtime interpretation allows DSM specification to call Java code to interact with Eclipse or any other system. Finally, the representation's appearance is not statically fixed allowing the end user to create, modify and manipulate elements recording the state into .aird file in the background [33] [32].

2.3.7 Eclipse Layout Kernel

Another important aspect of modeling tools usage is in readability of diagrams. Previously discussed graphical editors and their extension give the possibility to visualize the elements on diagram; nevertheless, only having elements on the diagram is not sufficient. In general a visual representation gives the possibility to understand and consume information better than textual view, but it requires and strongly depends on the layout of diagram's components According to [34] the manual layout of diagram elements takes around 30% of user time upon manual creation or modification of diagram. Since diagrams have 2-dimensions, the placement of an element usually affects the position and interrelations with other diagram elements, as a result may require a revision of existing elements' placement and connections. In case of automatic generation of diagram's components the time needed for manual layout increasing dramatically, furthermore, may require new layout iteration upon fresh synchronization of the model. Thus, in order to save the time and increase the performance of diagram understanding, automatic layout should be applied [34].

In order to solve the problem with layout and apply automatic layout with proper layout algorithm the Eclipse Layout Kernel[8] (ELK) project was selected. This project provides the library of layout algorithms and a bridge interface to connect with diagram editors. The pre-included bridge library of this plugin set is compatible with GMF-based editors, thus integration of this plugin does not require attention on source code level, but only configuration[9].

2.4 Validation Concepts and Methodology

Empirical studies [35] [36] [37], often in the form of survey or case study, compose the reliable method to evaluate and prove the merits of a new software engineering tool. In this section we present the formal software verification concept, which is based on the software survey and on the case study with elements of reverse engineering. The proposed approach of this thesis validation is a hybrid method of software evaluation and functional-structural testing.

[8] https://projects.eclipse.org/projects/modeling.elk

[9] https://projects.eclipse.org/proposals/elk-eclipse-layout-kernel

2.4.1 Design and Conduct of Surveys in Empirical Software Engineering

An empirical research forms the precondition for the success of the discipline of software engineering. Therefore, a survey as one of empirical research strategies for the collection of information from heterogeneous sources is required to collect data for practical validation of hypotheses developed by implementation of software concept. In this way, survey results may prove the external approval and improve the understanding of software perspectives. The surveys are classified by types: Descriptive, Explanatory and Explorative surveys. First type of survey makes assertions about some characteristics and their distribution across the set of objects, however, does not provide the explanation for the distribution itself. Second type of survey focuses on the explanation of preference of one object over another in the set of objects. Last, an Explorative survey defines a pre-study to determine opportunities and risks for a more thorough empirical investigation on later stages of research [35].

The survey process itself consists of six major steps [35]:

- Study definition – determining the goal of the survey;
- Design – adaption of survey goal into a set of questions;
- Implementation – adjustment of design to make it executable;
- Execution – the actual data collection and data processing;
- Analysis – interpretation of the data;
- Packaging – reporting the survey results.

The initial task for any kind of research survey is in goal definition; it determines the research area and sets the subjects of a survey. Upon formalization of a survey goal, it is possible to continue with its design. The design process requires the consideration of issues [35]:

- Definition of the target group and the survey sample;
- Conceptual model of the objects and variables of the survey;
- Approach for data collection;
- Questionnaire design;
- Approaches for data analysis.
- Validity concerns and limitations.

At the implementation phase of the survey, its owner produce, collect and prepare all the material that is required to conduct the survey. The effort to implement a questionnaire can be significant, as possible defects in the questionnaire can compromise the validity of the collected data, low level in quality and comprehension of questions may annoy and disorientate respondents. Upon the execution step the survey holder organizes the presentation of materials to the audience and collects required data. Next step, the interpretation and analysis of the gathered data according to the data qualification methods selected during the survey definition to satisfy the survey goal. The techniques for data handling can range from common sense analysis (using standard descriptive techniques) to sophisticated statistical analysis techniques. Finally, the packaging step is to publish the survey and its result so that external parties are able to understand the results and their contexts [35].

2.4.2 Design and Conduct of Case Studies in Empirical Software Engineering

Another approach in the field of research in software engineering is a Case Study. Its methodology is well suited for different kinds of software engineering exploration and evaluation. The case study usually assumes the situation of typical or real scenario and does not require the arrangement of a control as for experiment, allowing the observation and accumulation of specific data. Data is collected for a specific purpose throughout the study or at the final point of scenario outcome. Based on the gathered data, statistical analyses can be carried out. In addition, the case study operates not only with evaluation of reasons for certain phenomena occurrences, but also evaluates the differences between results and conditions of a case study. This task is completed through tracking of a specific attribute or establishing relationships between different attributes. This way, allows the comparison of certain parameters, results and techniques involved in the case study research scenario [37] [38].

When conducting a case study, there are five major process steps to follow [38]:

- Case study design: objectives are defined and the case study is planned;
- Preparation for data collection: procedures and protocols for data collection are defined;
- Collection of data: execution with data collection on the studied case;
- Analysis of collected data;
- Reporting.

A plan for a case study should clearly define the following elements [38]:

- Objective: what to achieve?
- The case: what is studied?
- Research questions: what to know?
- Methods: how to collect data?
- Selection strategy: where to seek data?

The objective of the study may be, for example, exploratory, descriptive, explanatory, or improving. The case of the study usually represented by a project, an individual, a group of people, a process, a product, a policy, a role in the organization, an event, a technology. Research questions set the entries for clarification and resolution during the case study. The principal decisions on methods for data collection are defined at design time for the case study, however, the adjustment and preciseness of methods might be adapted at later stages. The purpose of the selection is in the classification of a case that is expected to be typical, critical, revelatory or unique [38].

The principles and strategy for data collection of a case study is essential for evaluation and analysis of results. It is important to organize and use several data sources in a case study in order to limit the effects of single data source interpretation. Multiple sources allow the usage of a comparative research strategy, comparing the results of using one method or some form of manipulation, to the results of using another approach. Another possibility arises from fixation of a baseline of one solution characteristics, and then it is possible to compare the results from the usage of different solution with the figures from the baseline. Data collection techniques can be divided into three levels [37] [38]:

- Direct method – the researcher is in direct contact with subjects and collect data in real time;
- Indirect method – the researcher collects the raw data without actual interaction with subjects during data collection;
- Existing data – the researcher analyzes the existing compilation of data.

At the analysis stage the data is processed depending on the nature of input – quantitative or qualitative data. For quantitative data, the analysis typically includes analysis of descriptive statistics, correlation analysis, development of predictive models and hypothesis testing. In contrast, the qualitative analysis derives conclusions from the data, keeping a clear chain of evidence. The chain of evidence means that a reader should be able to follow the derivation of results and conclusions from the collected data [38].

Finally, the reporting, as any empirical study cannot be distinguished from its reporting. The report demonstrates the findings of the study, but is also the main source of information for judging the quality of the study [38].

2.5 Summary

In this chapter we introduced the Role Based Access Control architecture and SecureUML modelling language. We discussed the abilities and relations between RBAC and SecureUML, and proved the compatibility and conformity of these methods. Next, we presented the stack of technologies and tools that will be used for the implementation of RBAC model capturing and visualization through SecureUML for the Spring web application. Finally, we introduced the validation methods and concepts to use for evaluation of the thesis contribution. In the next chapter we concentrate on the contribution and solution.

3 Contribution

This chapter gives a brief overview about software contribution for the thesis topic. First, we introduce the concepts of software contribution. Next, we present an experimental web application we use for plugin concept implementation and validation. Finally, we describe the plugin concept implementation intended to process the web application and build the SecureUML model of RBAC on top of it.

3.1 Introduction

In this section we highlight the key concepts of a Spring web application with Spring Security that define the criteria for recognition and mapping of our contribution to achieve the representation of RBAC model via SecureUML modeling language. The common layout for a Spring web application is demonstrated in the Figure 12. We have 4 major layers: Web Layer – this layer is handling a web request, usually has a logical separation of client end and server end that form the Model-View-Controller paradigm [19]; Service Layer – represents the middle layer between the Web and Data, it organizes the operations on objects like Create-Read-Update-Delete (CRUD); Data Access Layer (or sometimes called Persistence Layer) is in charge for the actual storage of the data, with possibility to form data into the object (for example it combines together an entity class and its database representation); and finally, Domain Layer – is a collection of entities the application operate with across all the layers.

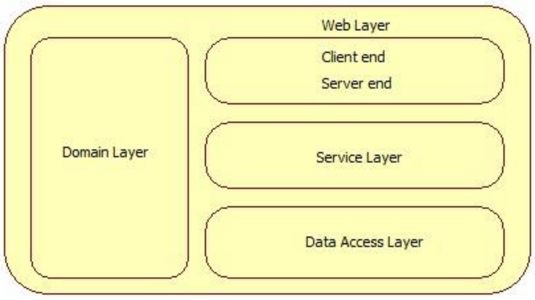


Figure 12. The layout of a Spring web application.

This layered division forms the stereotypes of a web application and defines the annotation level for every single java class that helps to separate and distinguish a purpose of a class as, for example, illustrated in the Figure 13.

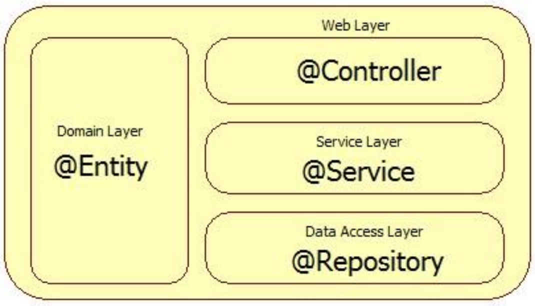


Figure 13. An example of stereotypes layout for a Spring web application.

Another opportunity to map a stereotype of a class in a Spring web application is in definition of bean in xml configuration file (illustrated in Code listing 1.).

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:tx="http://www.springframework.org/schema/tx"
       xmlns:jpa="http://www.springframework.org/schema/data/jpa"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
         http://www.springframework.org/schema/beans/spring-beans.xsd
         http://www.springframework.org/schema/data/jpa
         http://www.springframework.org/schema/data/jpa/spring-jpa-1.3.xsd
         http://www.springframework.org/schema/tx
         http://www.springframework.org/schema/tx/spring-tx-4.0.xsd">

  <jpa:repositories base-package="ee.ut.cs.msc.as.webapp.demo.repository"/>

</beans>
```

Code listing 1. An example of a stereotype assignment via xml configuration file.

This configuration (Code listing 1.) binds all the classes of the package `ee.ut.cs.msc.as.webapp.demo.repository` with `@Repository` stereotype.

In addition to usage of stereotype annotations on the Java class definition level, it is possible to use binding annotations for a class or a method, as it is illustrated in the Code listing 2.

```
package ee.ut.cs.msc.as.webapp.demo.controller;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

@Controller
@RequestMapping("/")
public class IndexWebController {

  @RequestMapping(value = "index/", method = RequestMethod.GET)
  public String index() {
    return "index";
  }
}
```

Code listing 2. An example of various annotations.

The combination of annotations (Code listing 2.) for this class shows the following information:

- `@Controller` – defines the stereotype of this class – Controller. It means this class may handle a web request.
- `@RequestMapping("/")` – this annotation of the class defines the value of a Uniform Resource Locator (URL) triggering the execution of this class upon a request with path `"/"`.
- `@RequestMapping(value = "index/", method = RequestMethod.GET)` – this annotation defines the binding of the method `index()`, executing method upon request on path `"/" + "index/"` (concatenation of the path for this class and this method); and request's definition is – Hypertext Transfer Protocol (HTTP) request of the type GET.

These conditions actually form the RBAC precondition, even though, we do not have the definition of a role, but we have a definition of the action and the control (in this certain scenario the control is not applied, it simply allows the execution of this operation upon any request). Nevertheless, we continue with our example and add missing components - a Role. To do so we add the support for Spring Security defining the complete RBAC model - we allow the usage of a control (whether allow or restrict triggering of a method action) upon the certain Role usage, assuming the default role allocation (Anonymous) unless it is not defined otherwise. Thus, if we extend our previous code example (Code listing 2), with Spring Security in mind, to have an access limitation we get an illustration of example - Code listing 3.

```
package ee.ut.cs.msc.as.webapp.demo.controller;

import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

@Controller
@RequestMapping("/")
public class IndexWebController {

  @RequestMapping(value = "index/", method = RequestMethod.GET)
  public String index() {
    return "index";
  }

  @PreAuthorize("hasRole('ROLE_ADMINISTRATOR')")
  @RequestMapping(value = "secret/", method = RequestMethod.GET)
  public String secret() {
    return "secret";
  }
}
```

Code listing 3. An example of various annotations with security configuration.

The analysis of this example (Code listing 3) shows as the following:

- `@Controller` – defines the stereotype of this class – Controller. It means this class may handle a web request.
- `@RequestMapping("/")` – this annotation of the class defines the value of a Uniform Resource Locator (URL) triggering the execution of this class upon a request with path `"/"`.
- `@RequestMapping(value = "index/", method = RequestMethod.GET)` – this annotation defines the binding of the method `index()`, executing method upon request on path `"/" + "index/"` (concatenation of the path for this class and this method) **for ANY Role**; and request's definition is – Hypertext Transfer Protocol (HTTP) request of the type GET.
- `@RequestMapping(value = "secret/", method = RequestMethod.GET)` – this annotation defines the binding of the method `secret()`, executing method upon request on path `"/" + "secret/"` (concatenation of the path for this class and this method) **for Administrator Role Only**; and request's definition is – Hypertext Transfer Protocol (HTTP) request of the type GET.

As we can see now, we have an example of RBAC model implemented on the top of Spring Controller with Spring Security responsible for access limitations in case of security rules definition.

Similar to definition of a stereotype via xml configuration, we can configure the Spring Security engine for protection of resources denoted by path in xml configuration - Code listing 4.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:sec="http://www.springframework.org/schema/security"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/security
http://www.springframework.org/schema/security/spring-security.xsd">
  <sec:global-method-security proxy-target-class="true" pre-post-
annotations="enabled" secured-annotations="enabled" >
    <sec:expression-handler ref="expressionHandler"/>
  </sec:global-method-security>
<sec:http disable-url-rewriting="false">
<sec:intercept-url pattern="/secret/**" access="hasRole('ROLE_USER')"/>
</sec:http>
</beans>
```

Code listing 4. An example of security configuration via xml configuration file.

Moreover, we can define the access settings via configuration class as it is illustrated in the Code listing 5.

```
package ee.ut.cs.msc.as.webapp.demo.config;

import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityConf
igurerAdapter;

@Configuration
public class SecurityConfiguration extends WebSecurityConfigurerAdapter {

  @Override
  protected void configure(HttpSecurity httpSecurity) throws Exception {
    httpSecurity
      .authorizeRequests()
        .antMatchers("/secret/**").hasRole("ROLE_ADMINISTRATOR");
  }

}
```

Code listing 5. An example of security configuration via configuration class.

Consequently, the security constraint for accessing the resource of "/secret/**" for users having Administrator role – might be defined in 3 ways (as it is illustrated in the Code listing 3, Code listing 4 and Code listing 5).

The validation of Access within Spring Security is organized via middle container (`AbstractSecurityInterceptor` as illustrated in the Figure 9) that proxies the call for method and apply controls if any specified. If security constraint is not satisfied, the Spring Security ends the flow of method call. The visual model of the RBAC behavior on the top of Spring Security is presented in the Figure 14.

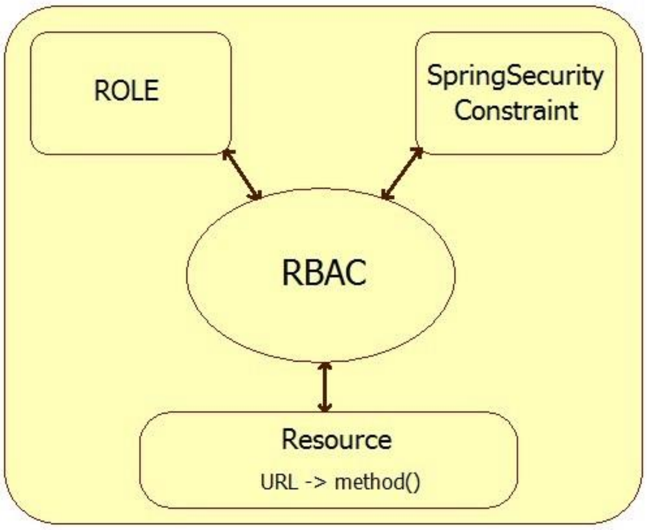


Figure 14. Role-based Access Control with Spring Security (adapted [23]).

In addition to Role validation and interception before the actual execution of a method, notated with Spring Security constraint, the Spring Security supports the roles hierarchy. This is equal to SecureUML term – generalization of roles. This functionality requires for configuration of roles dependencies as it is illustrated in the Code listing 6.

```
…
<bean id="roleVoter"
class="org.springframework.security.access.vote.RoleHierarchyVoter">
    <constructor-arg ref="roleHierarchy" />
</bean>
<bean id="roleHierarchy"

class="org.springframework.security.access.hierarchicalroles.RoleHierarchyImpl">
    <property name="hierarchy">
        <value>
            ROLE_ADMINISTRATOR > ROLE_MODERATOR
        </value>
    </property>
</bean>
…
```

Code listing 6. Configuration of Roles hierarchy with Spring Security.

The notation `"ROLE_ADMINISTRATOR > ROLE_MODERATOR"` (Code listing 6) means that Administrator may operate on behalf of Moderator and validation of `@PreAuthorize("hasRole('ROLE_MODERATOR')")` will allow to Administrator to access a method.

To summarize, in order to build the model of RBAC and represent it using SecureUML for a Spring web application with Spring Security, we define the following mapping:

- SecureUML *User* – we map to the User of the Spring Security – `UML Class`.
- SecureUML *Role* – we map to the Role of the Spring Security – `UML Class`.
- SecureUML *Permission* – we map to allowance of a Java class method triggering, assuming the *Role* limitation for performing an operation of this method – `UML Association class`.

- SecureUML *ActionType* – we map to the combination of URL and HTTP method that corresponds to the certain method of Java class representing the Resource – `UML Class`.
- SecureUML *Resource* – we map to a Java class having the support for serving's web requests (for example annotated with `@RequestMapping`) – `UML Class`.
- SecureUML *Constraint* – we map to a method call of a Java class (for example, the method `isValidInTime()` – defines the Constraint) – `UML Class`.

3.2 Web Application

In order to build and "train" our contribution, we created a model web application representing a test environment of research and implementation of the SecureUML plugin. As an example implementation we selected the simple message board application, which has different resources and roles and allows us make experiments with SecureUML plugin. The source code of this experimental web application can be found at Appendix - VI. Source Code.

3.2.1 Functionality

For the purpose to keep things simple, we designed the list of functional requirements of our model web application that is presented in the Table 2.

Table 2. The list of functional requirements for model web application.

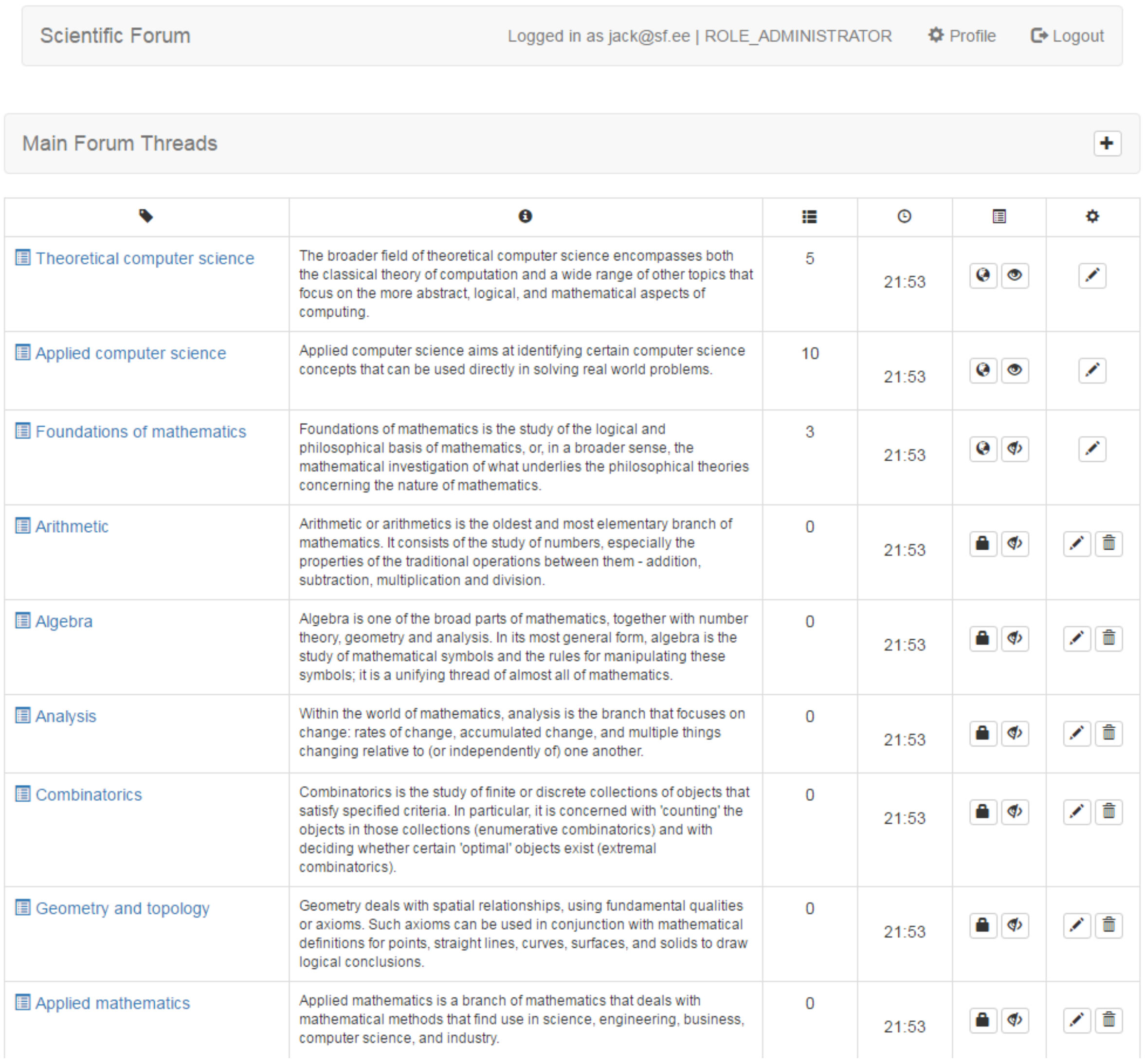
ID	Operation	Object	Condition	Role
F1	List forum threads	Forum threads	-	Administrator
F2	List forum threads	Forum threads	Forum thread is enabled.	Moderator, User
F3	List forum threads	Forum threads	Forum thread is enabled and is public.	Anonymous
F4	Select forum thread	Forum thread	-	Administrator
F5	Select forum thread	Forum thread	Forum thread is enabled.	Moderator, User
F6	Select forum thread	Forum thread	Forum thread is enabled and is public.	Anonymous
F7	List forum topics of certain forum thread.	Forum topics	-	Administrator
F8	List forum topics of certain forum thread.	Forum topics	Forum thread is enabled.	Moderator, User
F9	List forum topics of certain forum thread.	Forum topics	Forum thread is enabled and is public.	Anonymous
F10	Create forum thread	Forum thread	-	Administrator
F11	Update forum thread	Forum thread	-	Administrator
F12	Delete forum thread	Forum thread	-	Administrator

ID	Operation	Object	Condition	Role
F13	Create forum topic	Forum topic	Forum thread is enabled.	Moderator
F14	Update forum topic	Forum topic	Forum thread is enabled.	Moderator
F15	Delete forum topic	Forum topic	Forum thread is enabled.	Moderator
F16	Select forum topic	Forum topic	-	Administrator
F17	Select forum topic	Forum topic	Forum thread is enabled.	Moderator
F18	Select forum topic	Forum topic	Forum thread is enabled, forum topic is enabled.	User
F19	Select forum topic	Forum topic	Forum thread is enabled and is public; forum topic is enabled and public.	Anonymous
F20	List forum posts of certain forum topic.	Forum posts	-	Administrator
F21	List forum posts of certain forum topic.	Forum posts	Forum thread is enabled.	Moderator
F22	List forum posts of certain forum topic.	Forum posts	Forum thread is enabled; forum topic is enabled.	User
F23	List forum posts of certain forum topic.	Forum posts	Forum thread is enabled and is public; forum topic is enabled and public.	Anonymous
F24	Create forum post	Forum post		Administrator
F25	Create forum post	Forum post	Forum thread is enabled	Moderator
F26	Create forum post	Forum post	Forum thread is enabled; forum topic is enabled.	User
F27	Create forum post	Forum post	Forum thread is enabled and is public; forum topic is enabled and public.	Anonymous
F28	Update forum post	Forum post	Forum thread is enabled.	Moderator
F29	Update forum post	Forum post	Forum thread is enabled; forum topic is enabled.	Moderator, User
F30	Delete forum post	Forum post	Forum thread is enabled.	Moderator

Within our model web application we have:

- 4 Roles: Administrator, Moderator, User and Anonymous;
- 3 prime Entities: Forum thread, Forum topic and Forum post.
- 4 prime actions: Create, Read, Update and Delete.
- Built-in application server (Tomcat[10] via Spring Boot[11]) and database (H2[12]).
- Different types of Spring Controllers (`@Controller` and `@RestController`).

An example snapshot of this model web application is illustrated in the Figure 15.



Scientific Forum | Logged in as jack@sf.ee | ROLE_ADMINISTRATOR | Profile | Logout

Main Forum Threads

Theoretical computer science	The broader field of theoretical computer science encompasses both the classical theory of computation and a wide range of other topics that focus on the more abstract, logical, and mathematical aspects of computing.	5	21:53	🌐 👁	✎
Applied computer science	Applied computer science aims at identifying certain computer science concepts that can be used directly in solving real world problems.	10	21:53	🌐 👁	✎
Foundations of mathematics	Foundations of mathematics is the study of the logical and philosophical basis of mathematics, or, in a broader sense, the mathematical investigation of what underlies the philosophical theories concerning the nature of mathematics.	3	21:53	🌐 ⊘	✎
Arithmetic	Arithmetic or arithmetics is the oldest and most elementary branch of mathematics. It consists of the study of numbers, especially the properties of the traditional operations between them - addition, subtraction, multiplication and division.	0	21:53	🔒 ⊘	✎ 🗑
Algebra	Algebra is one of the broad parts of mathematics, together with number theory, geometry and analysis. In its most general form, algebra is the study of mathematical symbols and the rules for manipulating these symbols; it is a unifying thread of almost all of mathematics.	0	21:53	🔒 ⊘	✎ 🗑
Analysis	Within the world of mathematics, analysis is the branch that focuses on change: rates of change, accumulated change, and multiple things changing relative to (or independently of) one another.	0	21:53	🔒 ⊘	✎ 🗑
Combinatorics	Combinatorics is the study of finite or discrete collections of objects that satisfy specified criteria. In particular, it is concerned with 'counting' the objects in those collections (enumerative combinatorics) and with deciding whether certain 'optimal' objects exist (extremal combinatorics).	0	21:53	🔒 ⊘	✎ 🗑
Geometry and topology	Geometry deals with spatial relationships, using fundamental qualities or axioms. Such axioms can be used in conjunction with mathematical definitions for points, straight lines, curves, surfaces, and solids to draw logical conclusions.	0	21:53	🔒 ⊘	✎ 🗑
Applied mathematics	Applied mathematics is a branch of mathematics that deals with mathematical methods that find use in science, engineering, business, computer science, and industry.	0	21:53	🔒 ⊘	✎ 🗑

Figure 15. An example snapshot of the model web application.

[10] http://tomcat.apache.org
[11] http://projects.spring.io/spring-boot
[12] http://www.h2database.com/html/main.html

3.2.2 Configuration

The model web application has dependencies on different frameworks as it is illustrated in the Figure 16. The exact list of dependencies can be checked in the pom.xml describing the model of this web application.

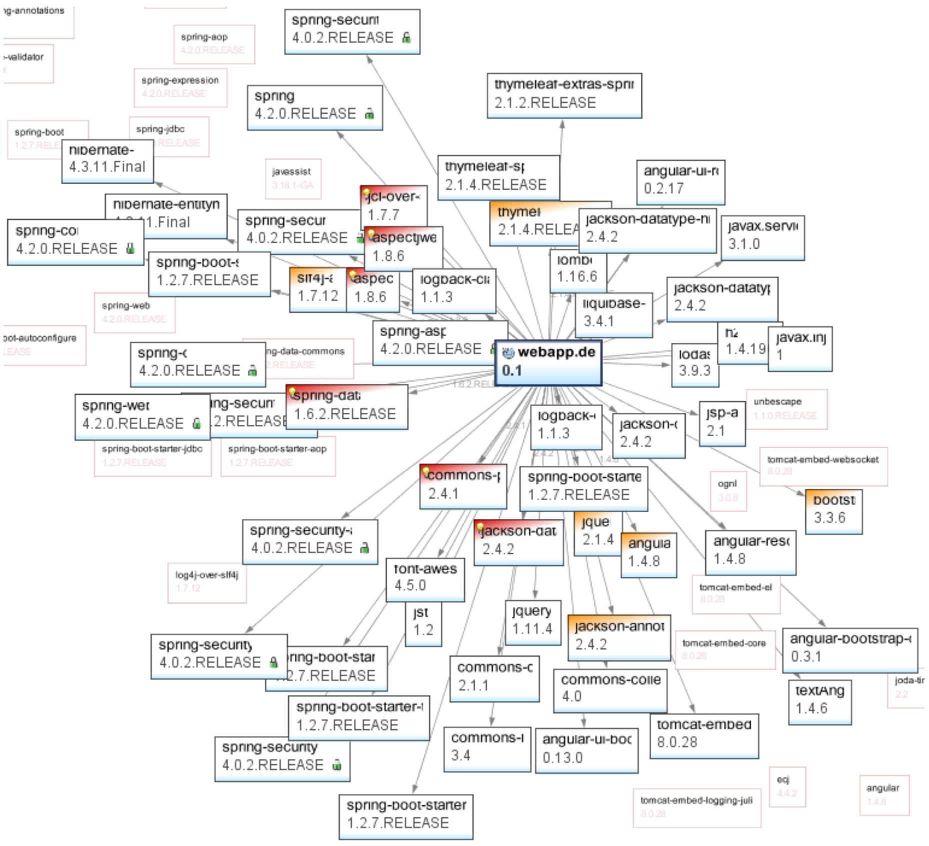


Figure 16. An overview of dependencies for the model web application.

3.2.3 Limitations

The Web application is developed to test the main contribution of this thesis. Therefore the major emphasis was placed on its rapid implementation than on its actual design. Thus this gives the following limitations:

- Application design and validation are treated as the second level citizens, thus putting the major emphasis on the implementation;
- The software quality of the application could also be seen as the limitation, since it was not properly tested. The exception, however, was done for the software parts related to the security constraints (e.g., integration test was performed to validate access violations);
- Some application scenarios could be seen unreal, however they contribute to the tests of the contribution.

3.3 SecureUML Plugin

In this section we give a brief introduction of the concept SecureUML plugin that is main contribution of this thesis. The source code of this experimental plugin can be found at Appendix - VI. Source Code.

3.3.1 Introduction

As we discussed in the section 3.1, it is possible to extract the model of RBAC of a Spring web application using the recognition of annotations. This is the main discovery strategy of this plugin. As our plugin is part of the Eclipse IDE, we use the internal resource of the workspace to access the web application project and within JDT we organize the detection of source by traversing the JDT Domain Object Model (DOM) and look for needed resources. Depending on the configuration (Figure 19) we filter out resources and inspect their methods for the Spring Security annotations and `@RequestMapping` annotations. On the top of URLs of `@RequestMapping` annotation we build a map to address the resources with relation to methods of a Java class. The detection of Roles is organized on the traverse of `@PreAuthorize` annotations + Anonymous role to map resources and their methods without security constraints. The detection of constraints is organized with the explicit configuration of these classes in the Configuration (Figure 19) + one default class - Authentication (`org.springframework.security.core.Authentication`).

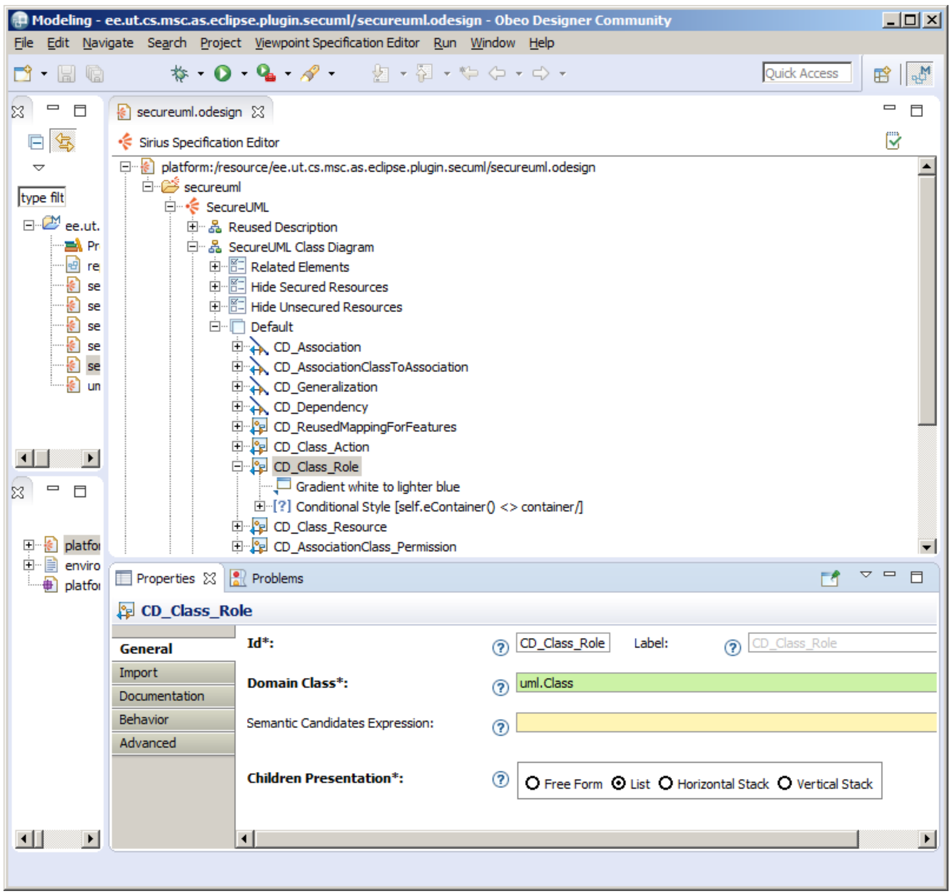
3.3.2 Use Cases

Main use cases are:

- Create SecureUML model: This process is illustrated in the Figure 23, Figure 24, Figure 25 and Figure 26.
- Synchronize SecureUML model: This process is illustrated in the Figure 27 and Figure 28.
- Configure SecureUML plugin: This process is illustrated in the Figure 19.
- Manage SecureUML Diagram: This process is illustrated in the Figure 29, Figure 30, Figure 31, Figure 32 and Figure 33.

3.3.3 Design Models

In order to design Sirius representation (.odesign) of our model resources, we used the Obeo Designer to describe the resource as it is illustrated in the Figure 17. Due to usage of EMF, we were able to reuse the UML library that describes the actual object behind its visual representation (in the Figure 17 we set the `uml.Class` definition as Domain Class of the element).


Modeling - ee.ut.cs.msc.as.eclipse.plugin.secuml/secureuml.odesign - Obeo Designer Community
File Edit Navigate Search Project Viewpoint Specification Editor Run Window Help
Quick Access
secureuml.odesign
Sirius Specification Editor
type filt
Properties
Problems

```
platform:/resource/ee.ut.cs.msc.as.eclipse.plugin.secuml/secureuml.odesign
  secureuml
    SecureUML
      Reused Description
      SecureUML Class Diagram
        Related Elements
        Hide Secured Resources
        Hide Unsecured Resources
        Default
          CD_Association
          CD_AssociationClassToAssociation
          CD_Generalization
          CD_Dependency
          CD_ReusedMappingForFeatures
          CD_Class_Action
          CD_Class_Role
            Gradient white to lighter blue
            [?] Conditional Style [self.eContainer() <> container/]
          CD_Class_Resource
          CD_AssociationClass_Permission
```

CD_Class_Role

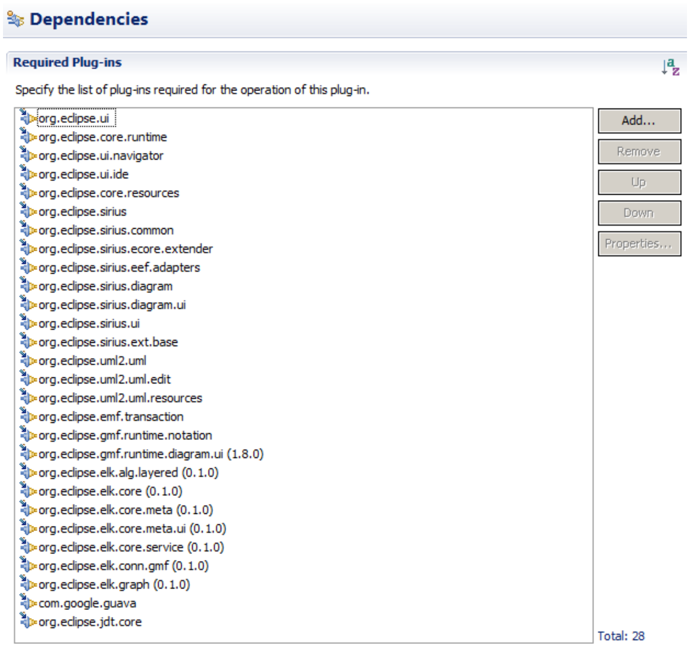
General | Import | Documentation | Behavior | Advanced

Id*: CD_Class_Role Label: CD_Class_Role
Domain Class*: uml.Class
Semantic Candidates Expression:
Children Presentation*: Free Form [✓ List] Horizontal Stack Vertical Stack

Figure 17. Design of plugin’s .odesign file.

3.3.4 Configuration

The implementation of the SecureUML plugin depends on the libraries denoted by other plugins. The overview is presented in the Figure 18.



Dependencies

Required Plug-ins

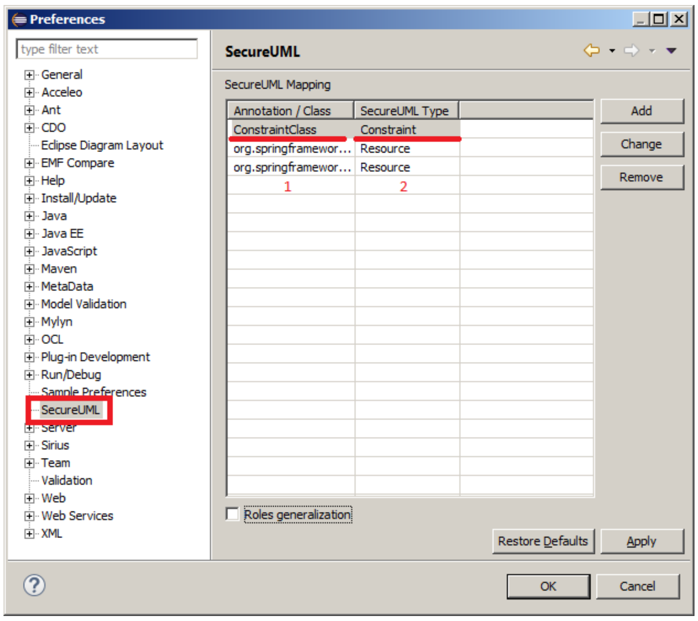
Specify the list of plug-ins required for the operation of this plug-in.

```
org.eclipse.ui
org.eclipse.core.runtime
org.eclipse.ui.navigator
org.eclipse.ui.ide
org.eclipse.core.resources
org.eclipse.sirius
org.eclipse.sirius.common
org.eclipse.sirius.ecore.extender
org.eclipse.sirius.eef.adapters
org.eclipse.sirius.diagram
org.eclipse.sirius.diagram.ui
org.eclipse.sirius.ui
org.eclipse.sirius.ext.base
org.eclipse.uml2.uml
org.eclipse.uml2.uml.edit
org.eclipse.uml2.uml.resources
org.eclipse.emf.transaction
org.eclipse.gmf.runtime.notation
org.eclipse.gmf.runtime.diagram.ui (1.8.0)
org.eclipse.elk.alg.layered (0.1.0)
org.eclipse.elk.core (0.1.0)
org.eclipse.elk.core.meta (0.1.0)
org.eclipse.elk.core.meta.ui (0.1.0)
org.eclipse.elk.core.service (0.1.0)
org.eclipse.elk.conn.gmf (0.1.0)
org.eclipse.elk.graph (0.1.0)
com.google.guava
org.eclipse.jdt.core
```

Total: 28

Figure 18. SecureUML plugin’s required dependencies.

The configuration of the plugin itself (Figure 19) is organized through standard configuration window of Eclipse IDE.



Annotation / Class	SecureUML Type
ConstraintClass	Constraint
org.springframewor…	Resource
org.springframewor…	Resource
1	2

Figure 19. SecureUML plugin configuration.

In the Figure 19: 1 is denoted for input of class (package + ‘.’ + Java Class name) and 2 for the definition of type (at the moment only Resource and Constraint types are supported).

3.3.5 Solution Illustration

The work with SecureUML plugin starts with the creation of a Modeling project that is prerequisite for SecureUML model creation. The creation of the Modeling project is organized in 3 steps. These steps are illustrated in the Figure 20, Figure 21 and Figure 22. First we select the type of project, then we select its name and finally create it.

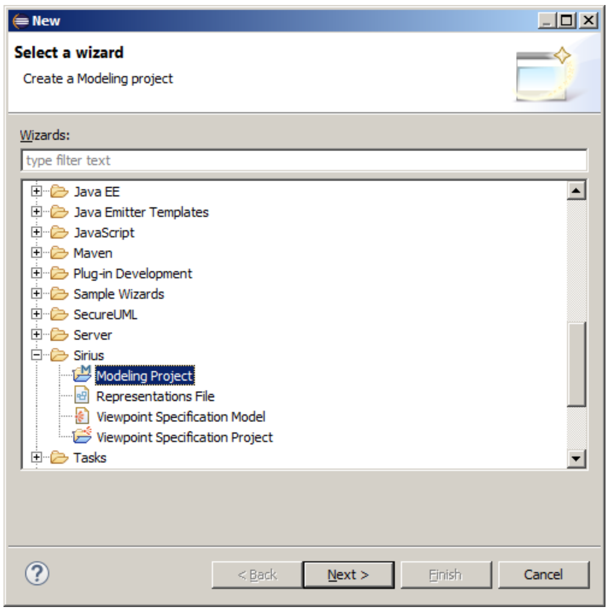
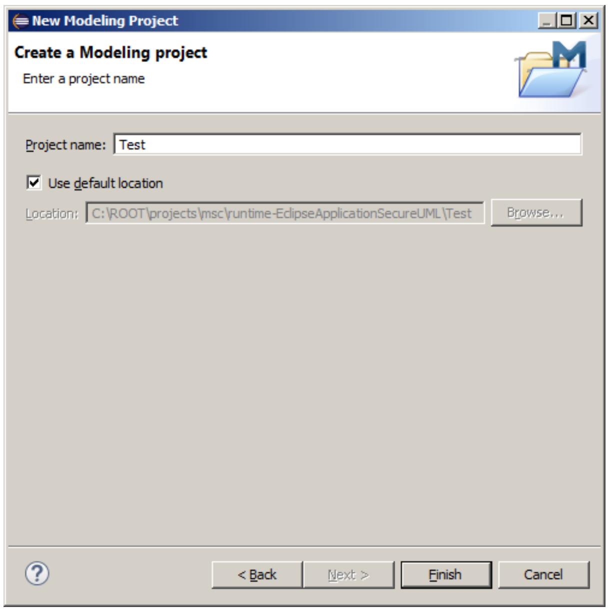


Figure 20. New modeling project creation wizard Step 1.



Project name: Test

☑ Use default location

Location: C:\ROOT\projects\msc\runtime-EclipseApplicationSecureUML\Test

Figure 21. New modeling project creation wizard Step 2.

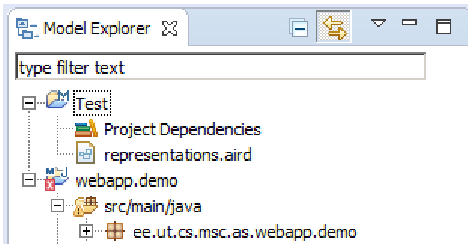
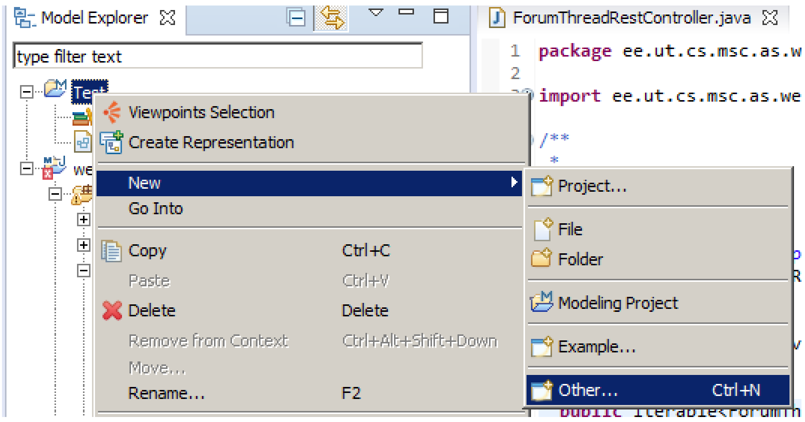


Figure 22. New modeling project creation wizard Step 3.

The second step for SecureUML plugin usage is in creation of SecureUML model. On the top of created Modeling project, we create a SecureUML model. This process is organized in 4 steps. These steps are illustrated in the Figure 23, Figure 24, Figure 25 and Figure 26. We select the creation of component, we select SecureUML model wizard, we name the file for our model and choose the location of it (previously created Modeling project), finally, we get the created SecureUML model.



```
package ee.ut.cs.msc.as.w

import ee.ut.cs.msc.as.we

/**
 *
```

Figure 23. A new SecureUML model creation Step 1.

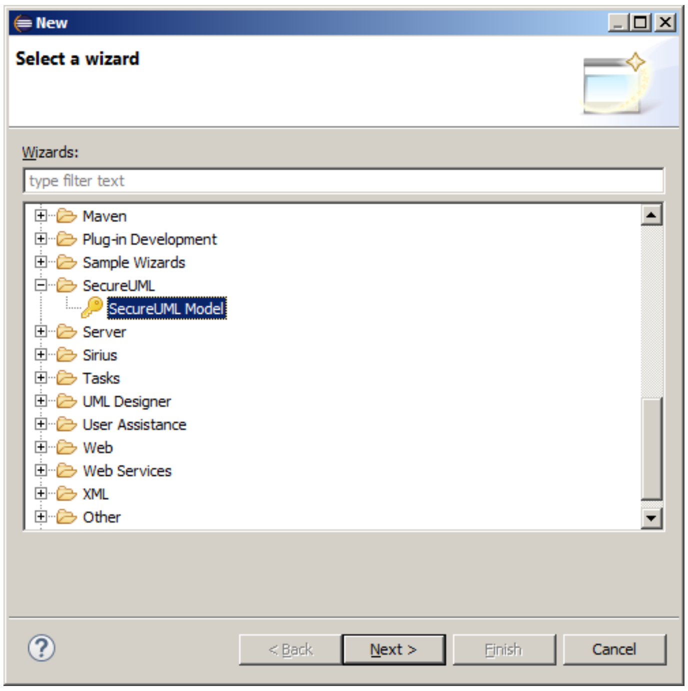


Figure 24. A new SecureUML model creation Step 2.

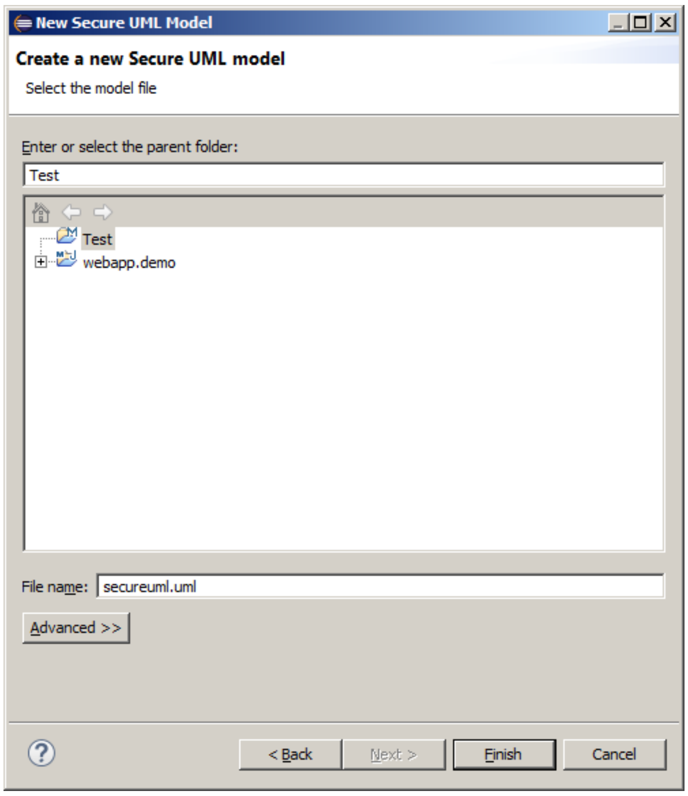


Figure 25. A new SecureUML model creation Step 3.

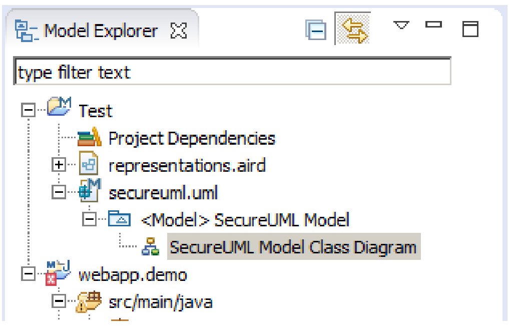


Figure 26. A new SecureUML model creation Step 4.

The newly created SecureUML model is empty (an existing SecureUML model may have existing resources; however, they might be replaced upon new Synchronization). In order to visualize the RBAC model with SecureUML we call the Synchronization as it is illustrated in the Figure 27.

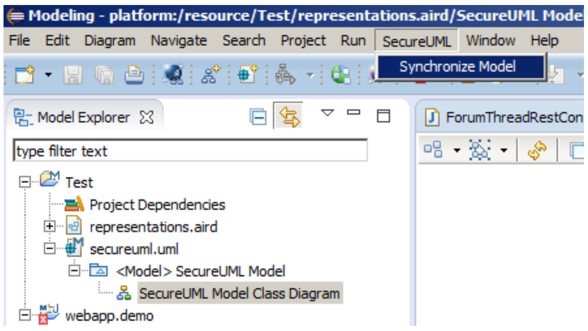

Modeling - platform:/resource/Test/representations.aird/SecureUML Mode
File Edit Diagram Navigate Search Project Run SecureUML Window Help
Synchronize Model
Model Explorer
ForumThreadRestCon
type filter text
Test
Project Dependencies
representations.aird
secureuml.uml
<Model> SecureUML Model
SecureUML Model Class Diagram
webapp.demo

Figure 27. SecureUML model synchronization Step 1.

The synchronization inspects the existing web application in the workspace of Eclipse IDE and builds the SecureUML model with visual representation. The result of synchronization is illustrated in the Figure 28.

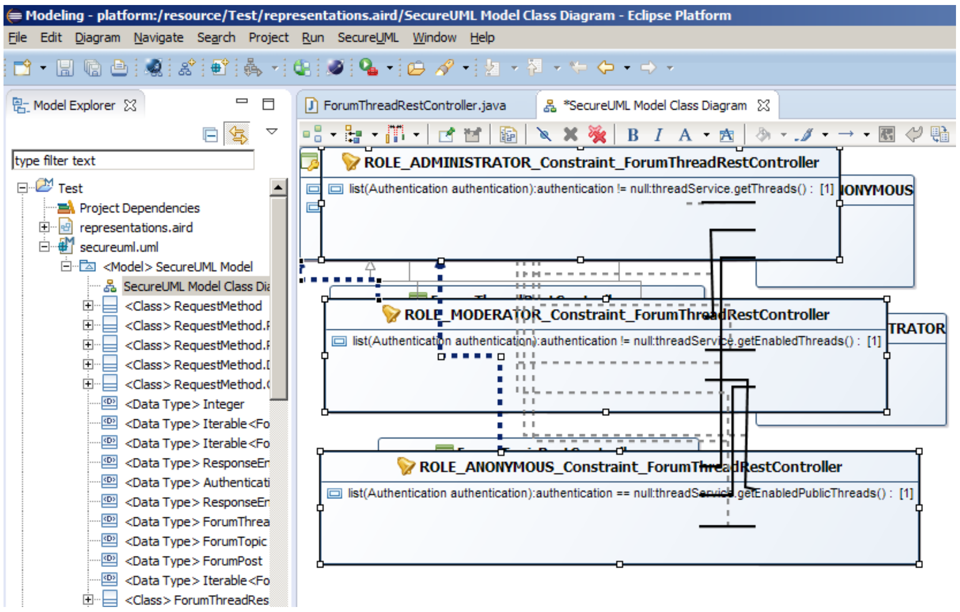
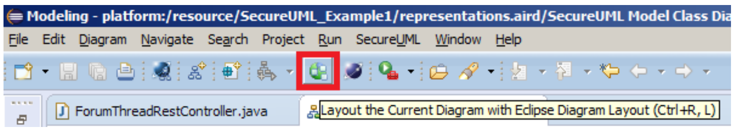

Modeling - platform:/resource/Test/representations.aird/SecureUML Model Class Diagram - Eclipse Platform
File Edit Diagram Navigate Search Project Run SecureUML Window Help
Model Explorer
ForumThreadRestController.java
*SecureUML Model Class Diagram
type filter text
Test
Project Dependencies
representations.aird
secureuml.uml
<Model> SecureUML Model
SecureUML Model Class Dia
<Class> RequestMethod
<Class> RequestMethod.F
<Class> RequestMethod.F
<Class> RequestMethod.I
<Class> RequestMethod.(
<Data Type> Integer
<Data Type> Iterable<Fo
<Data Type> Iterable<Fo
<Data Type> ResponseEr
<Data Type> Authenticati
<Data Type> ResponseEr
<Data Type> ForumThrea
<Data Type> ForumTopic
<Data Type> ForumPost
<Data Type> Iterable<Fo
<Class> ForumThreadRes
ROLE_ADMINISTRATOR_Constraint_ForumThreadRestController
list(Authentication authentication):authentication != null:threadService.getThreads() : [1]
ONYMOUS
ROLE_MODERATOR_Constraint_ForumThreadRestController
list(Authentication authentication):authentication != null:threadService.getEnabledThreads() : [1]
TRATOR
ROLE_ANONYMOUS_Constraint_ForumThreadRestController
list(Authentication authentication):authentication == null:threadService.getEnabledPublicThreads() : [1]

Figure 28. SecureUML model synchronization Step 2.

In order to apply the automatic layout of the diagram elements, the user should press the "Layout" icon as it is shown in the Figure 29. After that it is possible to check the model content using the Explorer view (Figure 30).



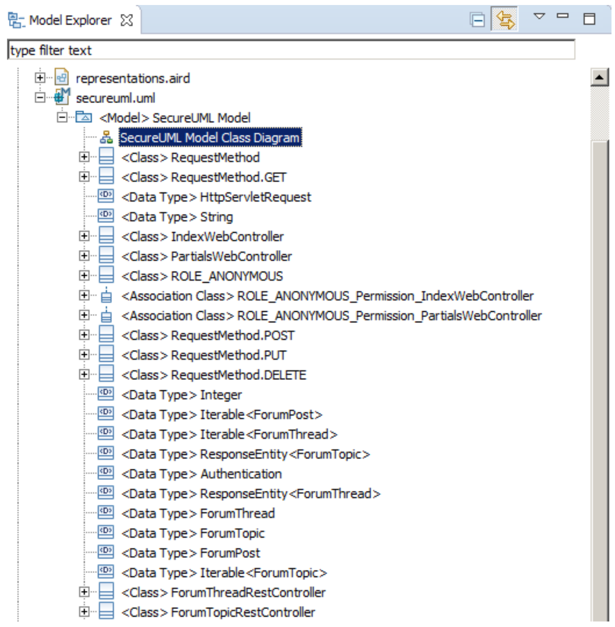
Modeling - platform:/resource/SecureUML_Example1/representations.aird/SecureUML Model Class Dia

File Edit Diagram Navigate Search Project Run SecureUML Window Help

ForumThreadRestController.java

Layout the Current Diagram with Eclipse Diagram Layout (Ctrl+R, L)

Figure 29. Automatic layout of diagram elements.



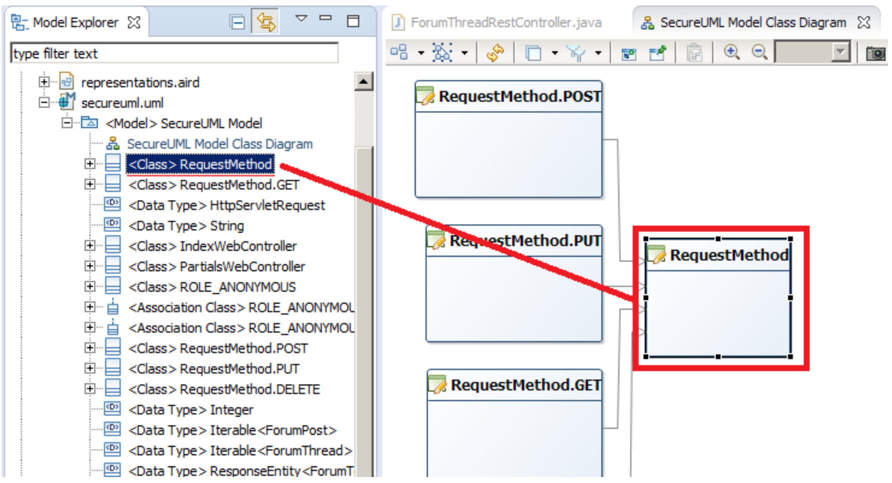
Model Explorer

type filter text

- representations.aird
- secureuml.uml
  - <Model> SecureUML Model
    - SecureUML Model Class Diagram
    - <Class> RequestMethod
    - <Class> RequestMethod.GET
    - <Data Type> HttpServletRequest
    - <Data Type> String
    - <Class> IndexWebController
    - <Class> PartialsWebController
    - <Class> ROLE_ANONYMOUS
    - <Association Class> ROLE_ANONYMOUS_Permission_IndexWebController
    - <Association Class> ROLE_ANONYMOUS_Permission_PartialsWebController
    - <Class> RequestMethod.POST
    - <Class> RequestMethod.PUT
    - <Class> RequestMethod.DELETE
    - <Data Type> Integer
    - <Data Type> Iterable<ForumPost>
    - <Data Type> Iterable<ForumThread>
    - <Data Type> ResponseEntity<ForumTopic>
    - <Data Type> Authentication
    - <Data Type> ResponseEntity<ForumThread>
    - <Data Type> ForumThread
    - <Data Type> ForumTopic
    - <Data Type> ForumPost
    - <Data Type> Iterable<ForumTopic>
    - <Class> ForumThreadRestController
    - <Class> ForumTopicRestController

Figure 30. Model explorer view of the SecureUML plugin.

The model explorer allows quick highlight of the resource in the diagram, to do so, just select the needed element in the Explorer and check the selection on the diagram (Figure 31).



Model Explorer

type filter text

- representations.aird
- secureuml.uml
  - <Model> SecureUML Model
    - SecureUML Model Class Diagram
    - <Class> RequestMethod
    - <Class> RequestMethod.GET
    - <Data Type> HttpServletRequest
    - <Data Type> String
    - <Class> IndexWebController
    - <Class> PartialsWebController
    - <Class> ROLE_ANONYMOUS
    - <Association Class> ROLE_ANONYMOU
    - <Association Class> ROLE_ANONYMOU
    - <Class> RequestMethod.POST
    - <Class> RequestMethod.PUT
    - <Class> RequestMethod.DELETE
    - <Data Type> Integer
    - <Data Type> Iterable<ForumPost>
    - <Data Type> Iterable<ForumThread>
    - <Data Type> ResponseEntity<ForumT

ForumThreadRestController.java

SecureUML Model Class Diagram

Figure 31. The highlighted selection of SecureUML plugin diagram.

The diagram might be big enough to fit on the screen, in order to understand your current location, you may use the outline view (Figure 32).

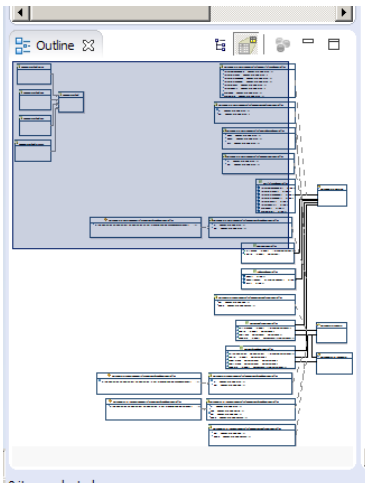


Figure 32. Outline view of the SecureUML plugin.

Within diagram you have a toolbar with icons (Figure 33). Using this icons you can optimize your view: 1 – Arrangement options; 2 – Selection options; 3 – Possible to refresh this diagram; 4 – Not used; 5 – Allows to apply a filter, to filter out the diagram elements; 6 – Allows to hide an element; 7 – Allows to pin the element; 8 – Not used; 9, 10 and 11 – Allows to configure the Zoom; 12 – Allows to export image of current view; 13 – Not used.

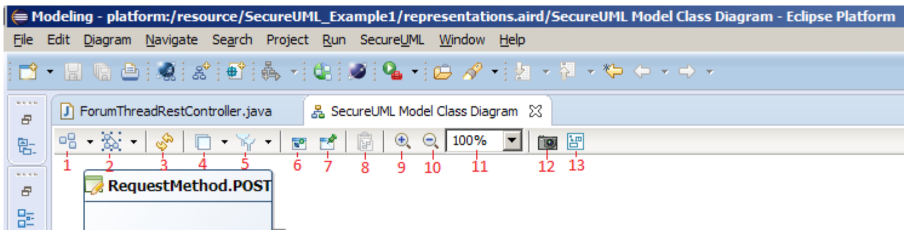

Modeling - platform:/resource/SecureUML_Example1/representations.aird/SecureUML Model Class Diagram - Eclipse Platform
File Edit Diagram Navigate Search Project Run SecureUML Window Help
ForumThreadRestController.java
SecureUML Model Class Diagram
100%
1
2
3
4
5
6
7
8
9
10
11
12
13
RequestMethod.POST

Figure 33. Toolbar of the SecureUML plugin diagram.

3.3.6 Limitations

Due to fact that SecureUML plugin is in conceptual phase of development the following limitations exist:

- It is not advised to use it for production solutions;
- The support for Spring Security annotations is limited;
- The support for Spring Security configuration reading is limited (xml and class configurations are ignored);
- Not all versions of Spring framework and Spring Security are supported.
- As it is discussed in the section 4.2.3.2 the models are different.

3.4 Summary

In this chapter we presented the brief overview about software contribution. First, we presented the main concepts we use for software contribution. Second, we presented the description of model web application we used for main contribution validation and implementation. Finally, we described the plugin concept implementation intended to process the web application and build the SecureUML model of RBAC on top of it. In the next chapter we concentrate on the validation and evaluation of software contribution.

4 Validation

This chapter gives an overview about theoretical and practical validation of thesis software contribution. First, we illustrate the conducted survey for software developers to evaluate the SecureUML plugin and its features. Finally, we provide the results and analysis of the case study that demonstrates the complexity and negative effects in manual discovery of the RBAC model of a web application, and then we highlight the benefits of automated recognition of a model.

4.1 Survey: Discover RBAC Model of Web Application via SecureUML plugin

The validation of SecureUML plugin was organized through the explorative survey and analysis of its results. The selected target group for evaluation was formed by 18 software developers, who may represent the end users of software. Survey participants were able to observe the demonstration of SecureUML plugin concept via face to face presentation at software development company Helmes[13] and via video[14] presentation for other interested software developers. After the introduction of software concept, the author answered respondent's questions. Finally, the audience was asked to complete the anonymous questionnaire in order to assess the concept of SecureUML plugin and its features.

4.1.1 Evaluation Scenario

As described in the section 2.4.1, the survey process is organized in six steps. Our first step is the identification of survey goal. In order to validate the contribution we set the goal of our survey – *We validate the acceptance and need of software developers for the SecureUML plugin*. To accomplish this goal, we collect the quantitative feedback of software developers on the concept of SecureUML plugin and the gathered data should prove the necessity of the plugin in the scope of support and audit in software development and a web application security implementation. Next, we design the survey questionnaire that is presented in the Appendix - II Survey: Questionnaire. The questionnaire was divided into 4 main blocks:

- Contextual questions of section "General" – contain the characterization questions for demonstration respondent's background in the scope of technology and experience;
- High-level questions of section "Modeling" – contain questions for showing the attitude and interest of respondent to modeling;
- Detail-level questions of section "SecureUML Plugin" – contain question for evaluation of functionality and outcome from the concept implementation of SecureUML plugin;
- Miscellaneous questions of section "Feedback" – allow respondents to provide a feed-back on presentation or video, and share the proposals for future improvements and implementations of SecureUML plugin.

[13] http://www.helmes.ee
[14] https://youtu.be/G4qp5wRmxHI

The questions of the questionnaire can be divided into four major types:

- Type I – the question assumes an integer answer;
- Type II – the question requires a Boolean answer;
- Type III – the question reflects the level of respondent's attitude to statement in the form of approval/disapproval reflected by integer answer;
- Type IV – the question with textual answer.

In order to evaluate the results of survey the standard descriptive technique was selected as analysis method for data interpretation and counting.

Our next step is to concentrate on survey implementation and execution. The execution of survey for software professional is challenging task [36], as they are busy and it is hard to schedule the presentation. This fact defined the strategy of survey implementation and execution. The survey performance was organized in two tracks:

- Live presentation: Software developers of Helmes Company were asked to participate in presentation of SecureUML plugin concept. During this presentation the author demonstrated the plugin and its features using the model web application (described in the section 3.2).
- Video presentation: Software developers were asked to watch the video containing demonstration of SecureUML plugin, and contact author for possible questions and comments.

After the demonstration, participants were asked to fill the questionnaire form. The gathered responses were collected, analyzed and packaged as it is showed in the next section.

4.1.2 Survey Results and Analysis

Depending on the type of the question I – III we acquire the quantitative results that can be handled with standard descriptive techniques. The question of type III measures the attitude of a respondent on the certain argument, his/her viewpoint was exposed through the rating scale of 0 to 4 values, where: 0 – Disagree, 1 – Almost disagree, 2 – Neutral position, 3 – Almost agree, 4 – Agree. Questions of type IV provide us freeform answers, thus, we build the collection of ideas and hints for future development and improvements on top of this feedback.

4.1.2.1 Contextual Questions

This questions block is presented by questions 1 – 6 containing Type I and Type II questions. The idea behind this section is in the characterization of a single respondent and in possibility to average the overview of experience and qualities of all respondents.

<u>Question 1: What is your development experience (years)?</u>

This question demonstrates the level of general experience in the field of software development. The average maturity of all respondents is 7 years, the minimum and maximum are 1 and 15 years.

<u>Question 2: What is your development experience with Spring (years)?</u>

This question demonstrates the level of Spring framework experience in the field of software development. The average maturity of all respondents is 2.5 years, the minimum and maximum are 0 and 5 years.

Question 3: What is your development experience with Spring Security (years)?

This question demonstrates the level of Spring Security framework experience in the field of software development. The average maturity of all respondents is 2 years, the minimum and maximum are 0 and 5 years.

Question 4: Have you had Spring Security Roles misconfiguration (yes/no)?

This question illustrates the career experience of a software developer. The result of this question is 9 respondents had experience with misconfiguration of roles; the rest of respondents didn't experience such problems. Even though, only half of respondents had problems with configuration and implementation of security, it clearly demonstrates the importance of proper configuration, as mistake in this area usually brings the security violation of an application.

Question 5: Do you have Eclipse IDE experience (yes/no)?

This question illustrates the conformity of key tool used for the SecureUML implementation. The majority of respondents (15) have an experience with required tool, thus, it will not require for additional efforts of learning the Eclipse itself, therefore, it may simplify the decision of the usage of SecureUML plugin.

Question 6: Do you have UML experience (yes/no)?

This question illustrates the conformity of key modeling technology used for the SecureUML visualization. The majority of respondents (16) have an experience with required modeling language, thus, it will not require for additional efforts of learning the SecureUML, therefore, it may simplify the decision of the usage of SecureUML plugin and its results.

4.1.2.2 High-level Questions

This questions block is presented by questions 7 – 11 containing Type III questions. The idea behind this section is in the attitude of a single respondent and in possibility to compile the overview of opinions and relations of all respondents to modeling and its usage in software development.

Question 7: The modeling is important in software development.

The majority of opinions (10 – 55%) confirmed the importance of modeling as a discipline in software development, the second group (7 – 38%) almost agrees to this statement. This result, probably, exposes the requirement for proper modeling in software development that may simplify and improve the development process by presentation of complex things through simplified viewpoints. The column chart of opinions is illustrated in the Appendix - III. Survey: Results - Figure 39.

Question 8: SecureUML is easy to understand.

The majority of opinions (9 – 50%) almost accept the simplicity of SecureUML notations; the second group (6 – 33%) confirmed this statement. This result, probably, exposes the effect of new notation and structure, as SecureUML concept is almost new to respondents, even though, it is an extension of UML they are familiar with. The column chart of opinions is illustrated in the Appendix - III. Survey: Results - Figure 40.

Question 9: SecureUML is easy to learn.

The majority of opinions (10 – 55%) almost accept the easiness of SecureUML as a subject for a study; the second group (7 – 38%) confirmed this statement. This result, probably, exposes the effect of UML knowledge that can be reused and adapted for learning the SecureUML. The column chart of opinions is illustrated in the Appendix - III. Survey: Results - Figure 41.

Question 10: Generation of models from source code is useful for me.

The majority of opinions (9 – 50%) almost accept the usefulness of models generation from existing source code; the second group (6 – 33%) confirmed this statement. This result, probably, exposes the effect of expectation of a developer to acquire a model of code or its parts that help in understanding of relations and dependencies of the software, however, the content of the models, their complexity and usefulness is unknown, thus, it bring the effect of uncertainty. The column chart of opinions is illustrated in the Appendix - III. Survey: Results - Figure 42.

Question 11: Modeling of software development can help me in everyday work.

The majority of opinions (8 – 44%) almost accept the possibility to use modeling approach on the daily basis; the second group (5 – 27%) confirmed this statement and the third group (4 – 22%) exposed the neutral position. This result, probably, demonstrates the effect of perspectives and possible differences in methods of daily software development, as every individual may have own habits and workflow that is hard to modify without reasonable motivation of change. The column chart of opinions is illustrated in the Appendix - III. Survey: Results - Figure 43.

4.1.2.3 Detail-level Questions

This questions block is presented by questions 12 – 18 containing Type III questions. The idea behind this section is in the attitude of a single respondent and in possibility to compile the overview of opinions and relations of all respondents to concept of SecureUML plugin and its perspectives in software development.

Question 12: SecureUML Plugin is easy to use.

The majority of opinions (13 – 72%) confirmed their feeling about simplicity of SecureUML plugin usage; the second group (3 – 16%) almost accepts this statement. This result, probably, demonstrates the benefits of straightforward operations of SecureUML plugin to achieve results that does not bring complexity of path flows with many operations. The column chart of opinions is illustrated in the Appendix - III. Survey: Results - Figure 44.

Question 13: SecureUML Plugin is easy to learn.

The majority of opinions (9 – 50%) almost accept the easiness of SecureUML plugin as a subject for a study; the second group (8 – 44%) confirmed this statement. This result, probably, demonstrates the effect of new software that brings some level of uncertainty due to lack of personal experience of software usage in the conditions and requirements of the end user. The column chart of opinions is illustrated in the Appendix - III. Survey: Results - Figure 45.

Question 14: SecureUML Plugin helps to understand the RBAC model.

The majority of opinions (10 – 55%) confirmed the improvement in understanding of RBAC model of a web application through the usage of SecureUML plugin; the second group (7 – 38%) almost accepted this statement. This result, probably, demonstrates the effect of model visualization that allows simplifying the way of discovery of RBAC model in comparison to manual audit of source code. The column chart of opinions is illustrated in the Appendix - III. Survey: Results - Figure 46.

Question 15: SecureUML Plugin helps to understand the usage of resources.

The majority of opinions (9 – 50%) almost agree with improvement in understanding of resources usage of a web application via SecureUML plugin output; the second group (6 – 33%) accepted this statement. This result, probably, demonstrates the effect of model visualization that allows simplifying the mapping of resources relations in comparison to manual audit of source code. The column chart of opinions is illustrated in the Appendix - III. Survey: Results - Figure 47.

Question 16: SecureUML Plugin improves the understanding of a web application.

The majority of opinions (10 – 55%) almost agree with improvement in understanding of a web application via SecureUML plugin results; the second group (5 – 27%) exposed the neutral position and the third group (3 – 16%) accepted this statement. This result, probably, demonstrates the effect of a new application that brings some level of uncertainty due to lack of personal experience and knowledge about software layout and its composition. The column chart of opinions is illustrated in the Appendix - III. Survey: Results - Figure 48.

Question 17: SecureUML Plugin helps to improve the security of a web application.

The majority of opinions (9 – 50%) confirmed the improvement in understanding of security model for a web application through the usage of SecureUML plugin; the second group (7 – 38%) almost accepted this statement. This result, probably, demonstrates the effect of model visualization that defines overall representation of security configuration for a web application in the scope of RBAC terms. The column chart of opinions is illustrated in the Appendix - III. Survey: Results - Figure 49.

Question 18: I would like to try the SecureUML Plugin.

The majority of opinions (9 – 50%) confirmed their wish to try the SecureUML plugin; the second group (7 – 38%) almost accepted this statement. This result, probably, demonstrates the interest of software developers to SecureUML plugin, as they want to try it within their implementation tasks and conditions. Another aspect of this question result is in possibility to form the experiment study within interested group of developers that helps to understand the actual usage of this plugin and provide data collection for future empirical researches. The column chart of opinions is illustrated in the Appendix - III. Survey: Results - Figure 50.

4.1.2.4 Miscellaneous Questions

This questions block is presented by 3 questions of Type IV. The idea behind this section is in the possibility to collect notes and remarks in free form that will not influence on the survey results, however, provide the feedback for research and future development. Therefore, in this section we show some answers that the author finds relevant and interesting.

Question: Did you like presentation?

The respondents were pleased with presentation and video, no complaints were reported.

Question: Do you have ideas/suggestions/feature requests?

The following list represents the feedback for this question:

- Full support of Spring Security configuration stack;
- Possibility to navigate from a diagram element to its source code;
- Support for version control with possibility to visualize differences between versions;
- Improve options for documentation support on the diagram;
- Improve options for differentiation of a diagram model: different coloring, different packaging and grouping of diagram artifacts;
- Multi-dimensional usage and manipulation of diagram – From source code to model and From model to source code;
- Creation and management of security profiles;
- Support for IntelliJ IDEA[15] development environment.

Question: Do you have a name suggestion for SecureUML Plugin?

The following list represents the feedback for this question:

- SecureRBAC Model;
- Role Visualizer;
- Spring SecureUML Plugin;
- SecureUML Plugin;
- Spring RBAC Modeler.

4.1.3 Discussion and Reporting

At this point of our survey we check our analysis (18 respondents in total) and the goal "We validate the acceptance and need of software developers for the SecureUML plugin", and make the following decisions:

The survey was successful:

This conclusion is formed on the top of answers for questions: 1 – 3. The selection of survey participants was correct, as respondents have an appropriate level of experience (average software development experience is 7 years) and knowledge (average experience of Spring and Spring Security frameworks are: 2.5 and 2 years) that represents the denoted area of software development and meets the target group of end users of SecureUML plugin.

The core methodology of SecureUML Plugin is accepted by software developers:

This conclusion is formed on the top of answers for questions: 6 – 11. The majority of respondents: (16) have experience with UML; (10) affirm the importance of modeling; (9) and (10) are optimistic in regard of SecureUML; (9) and (8) are optimistic in source code modeling and its daily usage.

[15] https://www.jetbrains.com/idea/

The core technologies of SecureUML Plugin are accepted by software developers:

This conclusion is formed on the top of answers for questions: 2, 3 and 5. Formation of experience in Spring (average 2.5 years) and Spring Security (average 2 years) frameworks arises from the usage of these technologies. The majority of respondents (15) have experience with Eclipse.

Software developers confirmed the acceptance of SecureUML Plugin:

This conclusion is formed on the top of answers for questions: 12, 13 and 18. The majority of respondents: (13) confirmed the simplicity of the plugin usage; (9) confirmed their wish to try the plugin; (9) are feeling optimistic about learning of the plugin usage, while, (8) are ready to learn it.

Software developers confirmed the need of SecureUML Plugin:

This conclusion is formed on the top of answers for questions: 4, 14 - 17. The majority of respondents: (9) admitted the problems with misconfiguration of roles during career; (10) find this plugin helpful in understanding of RBAC model; (9) accept the help of this plugin in improvement of web application security; (9) and (10) are feeling optimistic in abilities of plugin to help them with understanding of an entire web application and its resources usage.

To summarize, this survey confirmed the perspective in research area of creation and adaption of modeling tools that can help in understanding and visual processing of software. The transparency and audit of the software during development phase may improve the security of a web application via unified presentation of the application model. The integration of SecureUML plugin into development process offers to developers significant benefits and is mostly accepted by software development community that participated in this survey.

4.1.4 Threats to Validity

This section gives an overview about possible threats to validity and reliability of a given survey [35] [37]:

- The respondents are whether friends or colleagues of the researcher, that brings the effect of obligation to please an experimenter.
- The real usage of plugin by developer may change the opinion of a respondent. The opinion of a respondent was built on the top of author's experience and presentation; however, personal experience may correlate the motivation of software usage and uncover new shortcomings that worsen the satisfaction.
- A respondent's answer was inaccurate or the selection of answer was completed with mistake.
- Experimental validity of this survey – possible repetition of this survey or different selection of respondents may change the results of survey.
- Constructs validity of this survey – We assume the proper measurement of the right things; however, software's attitude measurement or ease of use of software might be in the way subjective.
- The conclusion validity of this survey – the possibility to make a mistake in conclusion and hypothesis regarding the relationship between questions and the outcome of this survey.

4.2 Case Study: Discover RBAC Model of Web Application via manual analysis

The validation and provability of usefulness [36] of software is a challenging task. However, in order to highlight possible benefits of the usage of contributed software we organize a case study. The major idea of this research is in demonstration of manual and automated discovery of RBAC Model of Web Application. On the result of case study we can compare the efficiency of every method and build the chain of evidence proving the need of software and automation. Yet, the process of software development usually involves different members like management, quality engineers, analysts, software developers and others, thus, in order to properly illustrate the manual recognition of RBAC model; we separate the recovery between two parties. This separation forms the sub-case of our research. One party for manual model recovery is a software developer; this role is delegated to author of this thesis. Another party for discovery is students of a course “Principles of Secure Software Design”[16] at University of Tartu[17]. The group of 46 students is divided to form 23 pairs of quality engineers with the task of recovery the RBAC model of a running web application and representation of their findings via SecureUML diagram. The students’ submissions are validated and compared to results of software developer. The outcome of this comparison is the quantitative data that may illustrate possible pitfalls in manual processing and help us to define aspects of manual recovery. Formulated metrics provide us with qualitative data that we can use at the upper level of our case study and make analysis of manual and automated discovery of a web application.

4.2.1 Evaluation Scenario

As described in the section 2.4.2, the case study is organized in five steps. First, we define the objectives of our case study and do planning. As case study may include elements of other research methods [38], we divide our scenario into two main stages with two separate goals:

- Stage A – we make a short case study on abilities to discover the RBAC model of a web application by hand. The goal of this case study is – *We validate the quality of SecureUML diagram that is created manually by using running application and description to reflect the RBAC model of this application*. In order to demonstrate the involvement of different parties in the software development we assume the creation of model by software developer and quality engineer. Furthermore, it helps to our case study to organize different data sources and use the comparative research strategy.
- Stage B – we make a comparison of RBAC models acquired by hand from the previous stage and generated within SecureUML plugin and name the advantages and disadvantages of every method. The goal of this case study – *We compare the consistency of manual and generated SecureUML diagrams and check limitations behind the process of diagrams creation*.

[16] https://courses.cs.ut.ee/2016/ssd/spring
[17] http://www.ut.ee/en

4.2.1.1 Stage A: Description, Planning and Collection

The definition of plan for Stage A: The discovery of model is provided by software developer having access to source code of web application and running instance of the web application. The quality engineer in his/her researches of RBAC model discovery uses the definition of the web application and running instance of the application. In order to satisfy the diversity of sources, the same task is completed by 23 quality engineers that are represented by pairs of students. On the top of the model of software developer we define the matrix of objects, attributes and operations we expect to be uncovered as required elements of the RBAC model. The model of software developer sets the baseline for our case study, as it is complete and precise due to access to all information sources. Upon submission of student task, its model is validated and results are forwarded to accumulation matrix for later analysis. The resulting matrix is represented by Boolean entries showing match or mismatch in model discovery. As a consequence of this stage we expect the qualitative output of observation and validation of models that we reuse as an input for the next stage.

The task definition is presented in the Appendix - IV. Case Study: Student Task. The test application is represented by the minified version of the model web application (described in the section 3.2.) and the list of its functions is presented in the Table 3. The reference to source code of minified web application is presented in the Appendix - VI. Source Code.

Table 3. The list of functions for student web application.

ID	Operation	Object	Condition	Role
F1	List forum threads	Forum threads	-	Administrator
F2	List forum threads	Forum threads	Forum thread is enabled.	User
F3	List forum threads	Forum threads	Forum thread is enabled and is public.	Anonymous
F4	Select forum thread	Forum thread	-	Administrator
F5	Select forum thread	Forum thread	Forum thread is enabled.	User
F6	Select forum thread	Forum thread	Forum thread is enabled and is public.	Anonymous
F7	List forum topics of certain forum thread.	Forum topics	-	Administrator
F8	List forum topics of certain forum thread.	Forum topics	Forum thread is enabled.	User
F9	List forum topics of certain forum thread.	Forum topics	Forum thread is enabled and is public.	Anonymous
F10	Create forum thread	Forum thread	-	Administrator
F11	Update forum thread	Forum thread	-	Administrator
F12	Delete forum thread	Forum thread	-	Administrator

The case scenario for Stage A has similarities to testing with elements of reverse engineering task. The testing assumes two base techniques: black-box testing and white-box testing. Black-box testing derives test cases from the specification or description of a program, while white-box testing derives test cases from the internal representation of a program [39]. In this scenario we use hybrid testing approach of black-box and white-box. The quality engineer derives test cases and model from the description and running application usage, making the reverse engineering in the sense of transformation of observations and findings into a model, accomplishing the black-box testing. While, software developer inspects the source code and validates the test cases via internal representation of the system, fulfilling the white-box testing.

In theory, in order to prove the possibility to complete this task, the discovery process of RBAC model for quality engineer is based on the steps [39] [40]:

- The reverse engineering process of quality engineer should start from building a preliminary model of the application by interacting with the existing Graphical User Interface (GUI). This process allows building a model views (representing objects and attributes).
- On capture of views, they are linked with navigations (actions) of the web page.
- Depending on the conditions (usage of credits in our case), the quality engineer should detect and observe the difference in behavior (for example, actions limitation) or views (objects and attributes) that actually demonstrates the difference in access control.
- Nevertheless, the inspection and detection of such models is usually costly [39] [40] because it requires the attention and differentiation during the usage of a web application.

Finally, we should add a remark about our case study that brings an additional threat to validity – the usage of computer science students. The review of 113 experiment publications of years 1993 to 2002, 72% of experiments reported about involvement of students [36]. However, in the review of the last decade the number of publications with students dropped to 23% of experiments [36]. This fact clearly demonstrates the recent trend - exclusion of students from studies [36].

4.2.1.2 Stage B: Description, Planning and Collection

The definition of plan for Stage B: First, we compare the consistency of diagrams created manually and generated; next, we compare limitations that may arise from manual or automated creation of diagrams. As result, we provide qualitative output of our observation and comparisons. In addition to data we discover during the comparison we may also reuse the facts and conclusions derived from the previous stage of the case study.

4.2.2 Case Study Results and Analysis

4.2.2.1 Stage A: Results and Analysis

According to the plan, the initial step for results analysis and comparison is in population of model by software developer. The discovery of raw model took the developer approximately 45 minutes, however, the formation of report and manual creation of the model using StarUML[18] software took about 1 hour and 30 minutes.

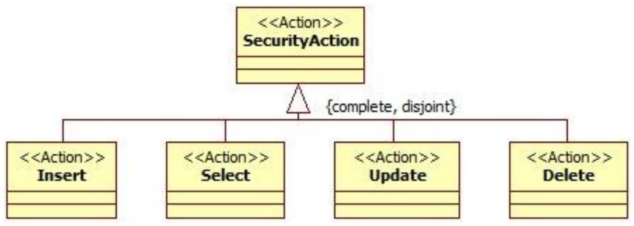


Figure 34. Security actions of student web application.

First, the developer identified the set of security actions that are illustrated in the Figure 34: Insert – creation of the object and its attributes; Select – reading of the object and its attributes; Update – modification of the object and its attributes; Delete – deletion of the object and its attributes. Next step was in creation of *Class* elements of UML with definition of stereotypes: `<<User>> Guest; <<User>> john; <<User>> jack; <<Role>> Anonymous; <<Role>> User; <<Role>> Administrator; <<Resource>> Forum Thread; <<Resource>> Forum Topic`. Third step was in definition of relations between *Class* objects and their multiplicity criteria. Next step was in determination of objects' attributes and operations. Finally, the developer discovered permissions and its operations, and added them to the model as Association Classes. During inspection of operations, the developer discovered Constraints `AC#1` and `AC#2` that sets the attribute default values upon creation of the `Forum Thread` – `Thread's topics` as 0 and `Thread's time` as creation date. The resulting diagram is presented in the Figure 35.

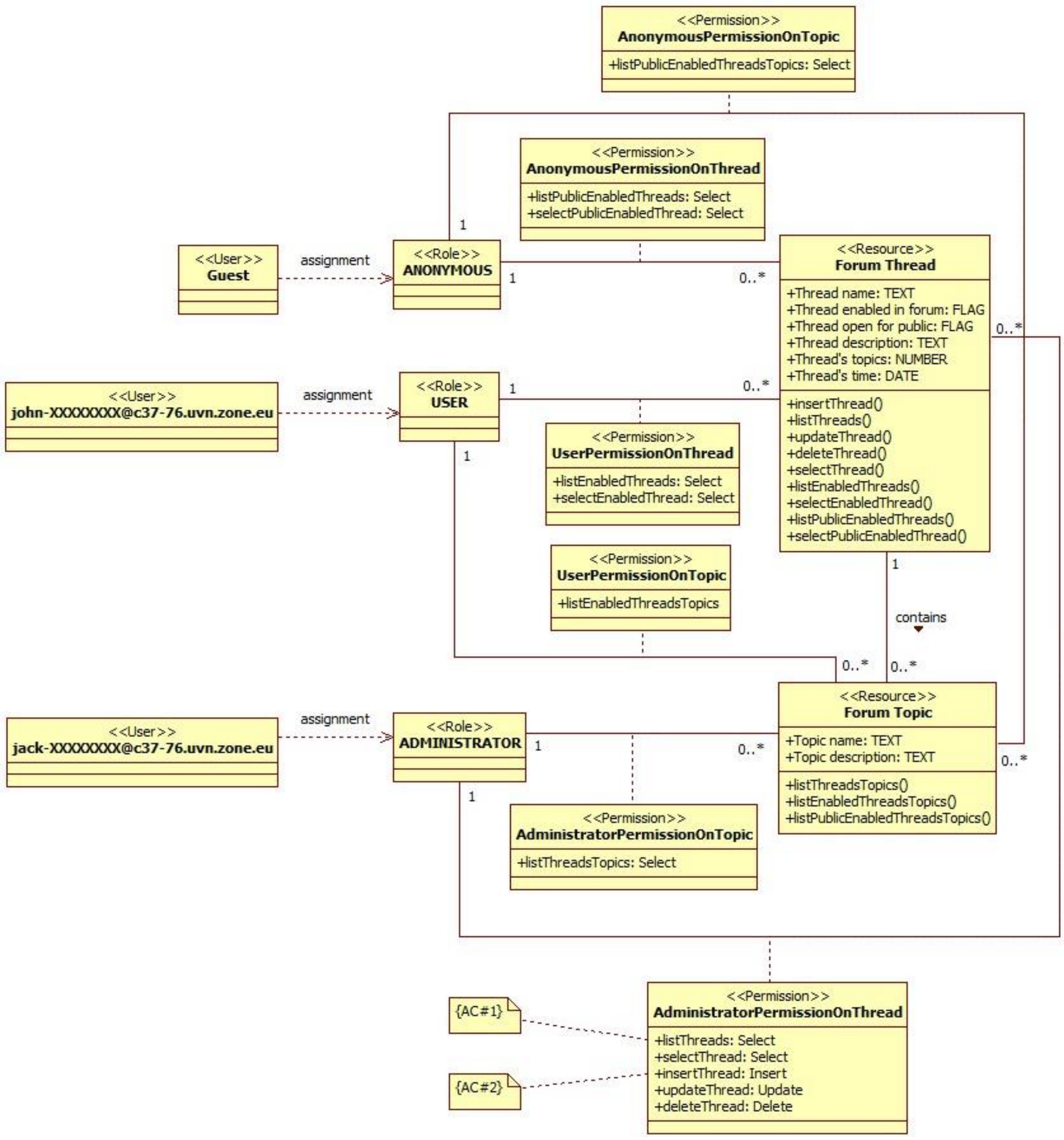
Next step, in order to accumulate the submission of students is in creation of matrix template. The x-axis of the matrix will contain the identifier of model feature to check, the y-axis will point to the student submission. The crossing of X and Y will show the 1 in case of match of model feature in the submission, 0 otherwise. Using the table of features (Table 3) and SecureUML diagram (Figure 35) we can identify the following list of model features to detect by quality engineer (QE):

- M1 – List of forum threads – QE identifies the list of objects;
- M2 – Forum thread – QE identifies the object;
- M3 – Forum thread attributes – QE identifies the attributes of object;
- M4 – List of forum topics – QE identifies the list of objects;
- M5 – Forum topic – QE identifies the object;
- M6 – Forum topic attributes – QE identifies the attributes of object;
- M7 – Users identified – QE identifies the list of users;
- M8 – Roles identified – QE identifies the list of roles;
- M9 – Anonymous permissions identified – QE identifies the permissions;
- M10 – User permissions identified – QE identifies the permissions;
- M11 – Administrator permissions identified – QE identifies the permissions;
- M12 – Enabled flag for User – QE identifies the constraint behind this flag for user;
- M13 – Enabled flag for Anonymous – QE identifies the constraint behind this flag for anonymous;

[18] https://sourceforge.net/projects/staruml/

- M14 – Public flag for Anonymous – QE identifies the constraint behind this flag for anonymous;
- M15 – Create forum thread – QE identifies the action;
- M16 – Update forum thread – QE identifies the action;
- M17 – Delete forum thread – QE identifies the action;
- M18 – Security actions identified – QE identifies the actions;
- M19 – Speculations – QE brings assumptions that do not exist in reality.

Note: The last model feature M19 is artificial, it does not reflect the correctness of detection of model, but demonstrates the noise of model by bringing superfluous artifacts – in our case it was in presence of Moderator Role or its related constructs. For this feature 0 was assigned for submissions without speculations, 1 otherwise.



<<Permission>> AnonymousPermissionOnTopic
+listPublicEnabledThreadsTopics: Select
<<Permission>> AnonymousPermissionOnThread
+listPublicEnabledThreads: Select
+selectPublicEnabledThread: Select
<<User>> Guest
assignment
<<Role>> ANONYMOUS
1
0..*
<<Resource>> Forum Thread
+Thread name: TEXT
+Thread enabled in forum: FLAG
+Thread open for public: FLAG
+Thread description: TEXT
+Thread's topics: NUMBER
+Thread's time: DATE
+insertThread()
+listThreads()
+updateThread()
+deleteThread()
+selectThread()
+listEnabledThreads()
+selectEnabledThread()
+listPublicEnabledThreads()
+selectPublicEnabledThread()
<<User>> john-XXXXXXXX@c37-76.uvn.zone.eu
<<Role>> USER
<<Permission>> UserPermissionOnThread
+listEnabledThreads: Select
+selectEnabledThread: Select
<<Permission>> UserPermissionOnTopic
+listEnabledThreadsTopics
contains
<<User>> jack-XXXXXXXX@c37-76.uvn.zone.eu
<<Role>> ADMINISTRATOR
<<Permission>> AdministratorPermissionOnTopic
+listThreadsTopics: Select
<<Resource>> Forum Topic
+Topic name: TEXT
+Topic description: TEXT
+listThreadsTopics()
+listEnabledThreadsTopics()
+listPublicEnabledThreadsTopics()
{AC#1}
{AC#2}
<<Permission>> AdministratorPermissionOnThread
+listThreads: Select
+selectThread: Select
+insertThread: Insert
+updateThread: Update
+deleteThread: Delete

Figure 35. SecureUML model of RBAC for student web application.

The raw resulting matrix of validation of student submissions on behalf of quality engineer is presented in the Appendix - V. Case Study: Results - Table 5. The adapted graphical representation as chart diagram of student submissions is illustrated in the Appendix - V. Case Study: Results - Figure 51.

The analysis of student submission allows us to state the fact – None of detected models captured all the model features, the best result was in detection of 16 model features without speculations (submission # 7) and the average level of features detection was 12 (66% out of all features except M19). Furthermore, the best detected features were:

- M15, M16, M17 – these features were detected with 100% match;
- M8, M10, M11 – demonstrated the good results were detected in 22 (95%) out of 23 submissions;
- M2 and M9 – these features were discovered in the most cases 21 (91%) out of 23 submissions.

Our analysis of the results allows determining the most problematic areas for discovery of a model:

- M1 and M4 – the detection of object lists caused the problem in 15 (65%) out of 23 submissions – this mistake, probably, shows the failure in reverse engineering of model and its components, it causes the misunderstanding of operations behind the view;
- M3 and M6 – the detection of object’s attributes caused the problem in 13 (56%) out of 23 submissions, and 10 (43%) out of 23 submissions – this mistake, probably, shows the failure in reverse engineering and low concentration on details;
- M5 – the detection of children objects caused the problem in 12 (52%) out of 23 submissions – this mistake, probably, shows the failure in reverse engineering of model and its components, it causes the misunderstanding of objects chaining and relation;
- M12, M13 and M14 – the detection of access limiting flags caused the problem in the submissions 19 (82%) out of 23, 22 (95%) out of 23, 12 (52%) out of 23 – this is *the most dramatic failure*, probably, due to low concentration and lack of experience. These features played the key role for the whole RBAC model, such as; they limited the access to objects depending on the role;
- M19 – the number of wrong assumptions is significant 8 (34%) out of 23 submissions – for the purpose, the introduction of task mentioned the moderator role, however, this caused the speculation about existence of this role with wrong assumptions about possible operations and permissions.

4.2.2.2 Stage B: Results and Analysis

The generated model (Figure 36) and manual model (Figure 35) are quite similar in notation; however, the following differences are discovered:

- The definition of Security Actions is different in manually created and generated diagrams. The manual model operates with human understandable naming; however, the generated one uses the access logic method. Even though, the meaning is similar, for example `RequestMethod.POST` is equivalent for `Insert`, it looks differently.

- The naming is different. The names of `Permissions`, `attributes`, `operations` are different. The manual model operates with human understandable naming; however, the generated one uses the naming derived from the source code. Although, the meaning is equal, for example `add()` is equivalent for `insertThread()`, it looks differently.
- The Key Resources are different. Manual diagram represents the objects, like `Forum Thread` and `Forum Topic`. While, the generated model represents the end points that actually service these objects `ForumThreadRestController` and `ForumTopicRestController`.
- The generate model is more precise in comparison to manual model, in the scope of usage input and output parameters. For example, operations `selectThread()` and `get(id : Integer)`.
- The generated model includes the hierarchy of Roles, for example, it uses the generalization of `ROLE_ADMINISTRATOR` that includes `ROLE_USER`. However, manual model does not, unless it is implemented manually.
- Generated model does not show the mapping of `Users` to `Roles`, as it assumes the dynamic assignment of `Roles` on the application level. However, the manual model may include `Users` mapping as an example.
- Generated model has limitation due to functional possibilities of SecureUML plugin. Generated model: does not define the level of multiplicity between diagram elements; does not provide supporting information like: contains, includes for the linked resources; does not recognize the complex constraints like `{AC#1}` and `{AC#2}` for the manual model.
- The addressing of manual and generated models is different. The generated model interprets the source code, thus, it is possible to track back model's elements to source code directly, while, manual model addresses the human terms, unless it is not prepared with proper usage of original names from source code.

On the level of diagram creation process we can determine the following differences:

- The generation of the model requires much less efforts for preparation and execution in comparison to the manual creation. Moreover, the time of a manual process may dramatically increase because of a manual layout of diagram elements [34].
- The generated model can be created quickly, thus, the automated process always reflects the latest state, while, manual process brings the fixation of a state.
- Automated creation of a diagram may change the level and way of thinking, as manual creation requires processing of information in depth.
- Manual processing of code for the manual model creation causes analysis activation, sometimes; this kind of inspection brings the ideas about possible optimizations and improvements.
- The generated diagram has interactive capabilities of the visualization through filtration possibilities. The filtration allows limiting the viewpoint of a diagram to a certain criteria. The illustrations of filtering are shown in the Figure 37 – the viewpoint is limited to unsecured resources; and in the Figure 38 – the viewpoint is limited to secured resources.

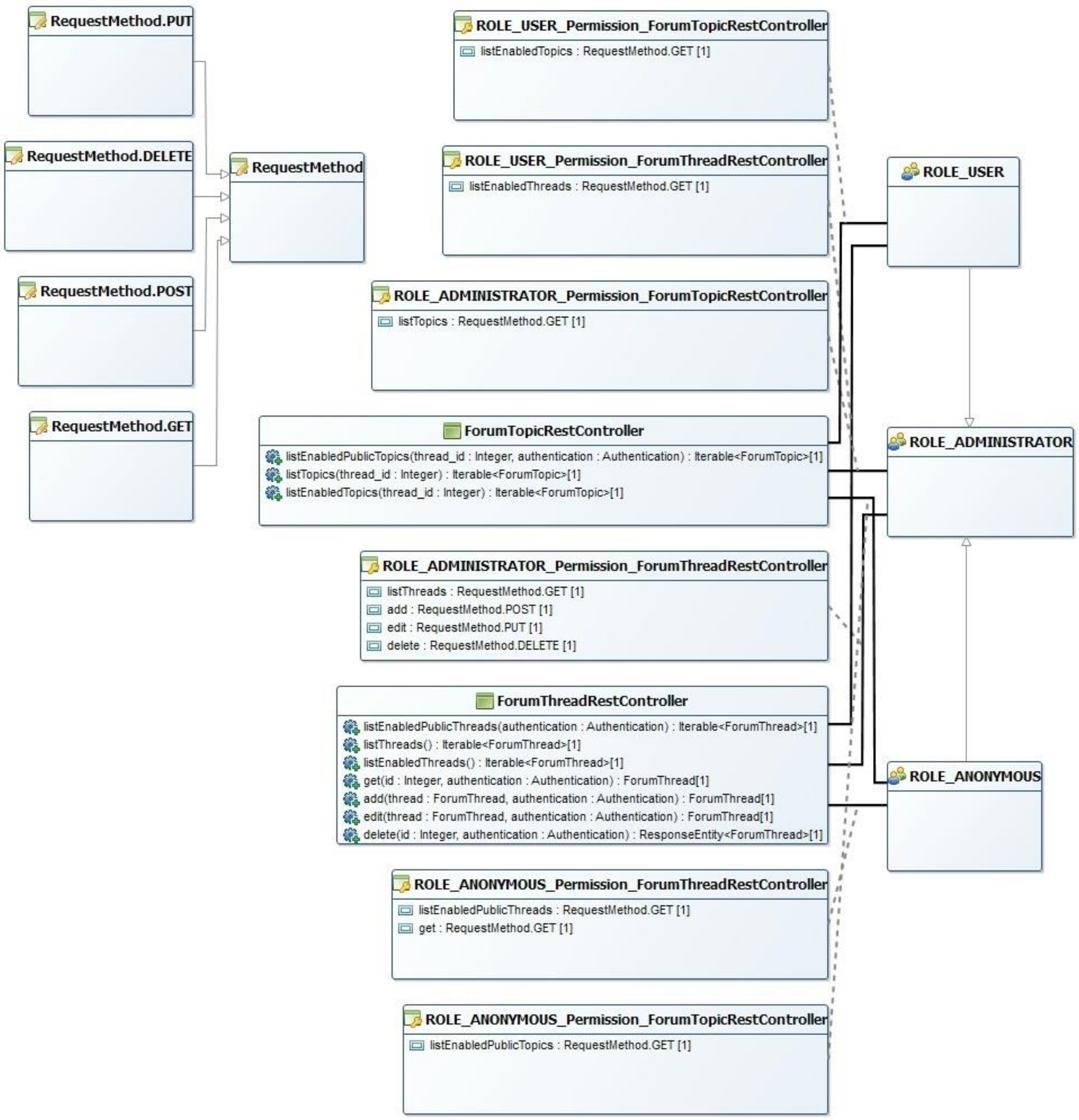

RequestMethod.PUT
RequestMethod.DELETE
RequestMethod.POST
RequestMethod.GET
RequestMethod
ROLE_USER_Permission_ForumTopicRestController
listEnabledTopics : RequestMethod.GET [1]
ROLE_USER_Permission_ForumThreadRestController
listEnabledThreads : RequestMethod.GET [1]
ROLE_ADMINISTRATOR_Permission_ForumTopicRestController
listTopics : RequestMethod.GET [1]
ForumTopicRestController
listEnabledPublicTopics(thread_id : Integer, authentication : Authentication) : Iterable<ForumTopic>[1]
listTopics(thread_id : Integer) : Iterable<ForumTopic>[1]
listEnabledTopics(thread_id : Integer) : Iterable<ForumTopic>[1]
ROLE_ADMINISTRATOR_Permission_ForumThreadRestController
listThreads : RequestMethod.GET [1]
add : RequestMethod.POST [1]
edit : RequestMethod.PUT [1]
delete : RequestMethod.DELETE [1]
ForumThreadRestController
listEnabledPublicThreads(authentication : Authentication) : Iterable<ForumThread>[1]
listThreads() : Iterable<ForumThread>[1]
listEnabledThreads() : Iterable<ForumThread>[1]
get(id : Integer, authentication : Authentication) : ForumThread[1]
add(thread : ForumThread, authentication : Authentication) : ForumThread[1]
edit(thread : ForumThread, authentication : Authentication) : ForumThread[1]
delete(id : Integer, authentication : Authentication) : ResponseEntity<ForumThread>[1]
ROLE_ANONYMOUS_Permission_ForumThreadRestController
listEnabledPublicThreads : RequestMethod.GET [1]
get : RequestMethod.GET [1]
ROLE_ANONYMOUS_Permission_ForumTopicRestController
listEnabledPublicTopics : RequestMethod.GET [1]
ROLE_USER
ROLE_ADMINISTRATOR
ROLE_ANONYMOUS

Figure 36. Generated SecureUML model of RBAC for student web application.

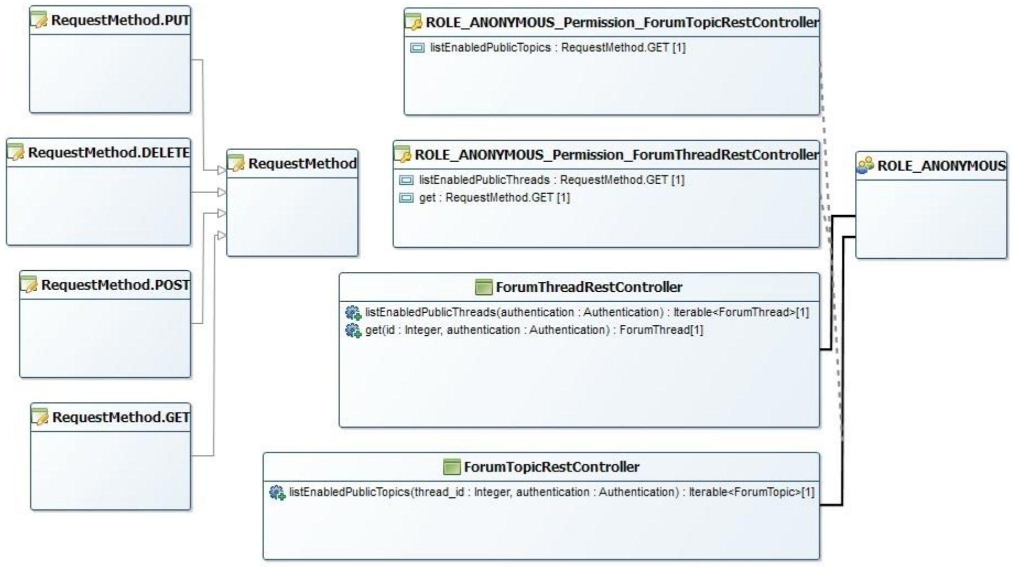

RequestMethod.PUT
RequestMethod.DELETE
RequestMethod.POST
RequestMethod.GET
RequestMethod
ROLE_ANONYMOUS_Permission_ForumTopicRestController
listEnabledPublicTopics : RequestMethod.GET [1]
ROLE_ANONYMOUS_Permission_ForumThreadRestController
listEnabledPublicThreads : RequestMethod.GET [1]
get : RequestMethod.GET [1]
ForumThreadRestController
listEnabledPublicThreads(authentication : Authentication) : Iterable<ForumThread>[1]
get(id : Integer, authentication : Authentication) : ForumThread[1]
ForumTopicRestController
listEnabledPublicTopics(thread_id : Integer, authentication : Authentication) : Iterable<ForumTopic>[1]
ROLE_ANONYMOUS

Figure 37. Generated SecureUML model of RBAC for student web application with applied filter - show only unsecured resources.

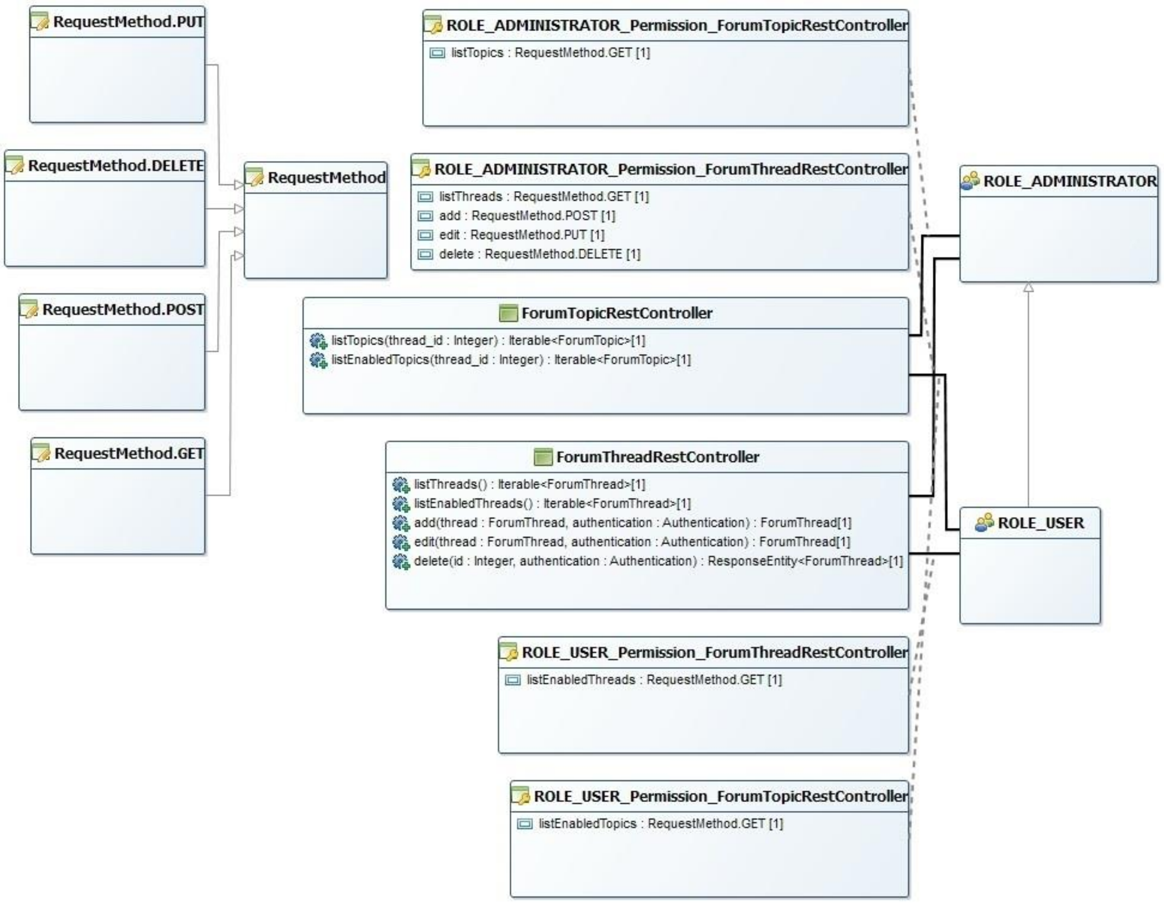

RequestMethod.PUT
RequestMethod.DELETE
RequestMethod.POST
RequestMethod.GET
RequestMethod
ROLE_ADMINISTRATOR_Permission_ForumTopicRestController
listTopics : RequestMethod.GET [1]
ROLE_ADMINISTRATOR_Permission_ForumThreadRestController
listThreads : RequestMethod.GET [1]
add : RequestMethod.POST [1]
edit : RequestMethod.PUT [1]
delete : RequestMethod.DELETE [1]
ForumTopicRestController
listTopics(thread_id : Integer) : Iterable<ForumTopic>[1]
listEnabledTopics(thread_id : Integer) : Iterable<ForumTopic>[1]
ForumThreadRestController
listThreads() : Iterable<ForumThread>[1]
listEnabledThreads() : Iterable<ForumThread>[1]
add(thread : ForumThread, authentication : Authentication) : ForumThread[1]
edit(thread : ForumThread, authentication : Authentication) : ForumThread[1]
delete(id : Integer, authentication : Authentication) : ResponseEntity<ForumThread>[1]
ROLE_USER_Permission_ForumThreadRestController
listEnabledThreads : RequestMethod.GET [1]
ROLE_USER_Permission_ForumTopicRestController
listEnabledTopics : RequestMethod.GET [1]
ROLE_ADMINISTRATOR
ROLE_USER

Figure 38. Generated SecureUML model of RBAC for student web application with applied filter - show only secured resources.

4.2.3 Discussion and Reporting

At this point of our case study we use our analysis results and formulate the completion of our goals for each Stage.

4.2.3.1 Stage A

For Stage A of the case study we check our analysis results and the goal "We validate the quality of SecureUML diagram that is created manually by using running application and description to reflect the RBAC model of this application", and make the following decisions:

<u>The discovery of a model is a complicated task and requires experience and attention:</u>

- None of 23 models captured all the model features;
- The best result was in detection of 16 (88%) out of 18 model features;
- The average level of features detection was 12 (66%) out of 18 model features.

<u>A small detail may have a great influence on the model; however, it is possible to miss it:</u>

- The detection of access limiting flags M12, M13 and M14 caused the problem in the submissions 19 (82%) out of 23, 22 (95%) out of 23, 12 (52%) out of 23.

<u>The mismatch of description and the actual state of an application may bring the confusion of a model:</u>

- The number of wrong assumptions is 8 (34%) out of 23 submissions.

<u>The manual creation of a model may cause the problem of resources disposition:</u>

- The detection of children objects caused the problem in 12 (52%) out of 23 submissions;
- The detection of object lists caused the problem in 15 (65%) out of 23 submissions;
- The detection of object's attributes M3 and M6 caused the problem in 13 (56%) out of 23 submissions, and 10 (43%) out of 23 submissions.

4.2.3.2 Stage B

For Stage B of the case study we check our analysis results and the goal "We compare the consistency of manual and generated SecureUML diagrams and check limitations behind the process of diagrams creation", and make the following decisions:

<u>Manual and generated models are different</u>:

- Due to difference in creation of a model, whether manually or from source code the naming strategy and interpretation of terms may vary (for example, action types are `RequestMethod.POST` and `Insert`);
- Due to difference in creation of a model, whether manually or from source code the key resource may vary (for example, resources are `Forum Thread` and `ForumThreadRestController`);
- A generated model contains optimizations by default; however, the manual model may require additional efforts for this (for example, generalization of roles);
- A manual model may contain specific details and notes, which do not exist on the generated model (for example, complex constraints like `{AC#1}` and `{AC#2}`);
- Despite the differences in presentation, both models are similar – they have analogue equalities (for example, operations are `selectThread()` and `get(id : Integer)`).

Every process, whether a manual process or an automated one, has own limitations and benefits:

- The manual process requires much more time and efforts.
- Manual process has a fixed state, it is fixed to the moment of creation, and thus, after a while the manual model is outdated.
- Automated process allows the interaction with model, the manual process not.
- Manual process may activate the process of re-thinking and optimization of the software, while automated process is more about consumption of results.
- In case of process automation, members of the software development team may concentrate on their prime goals, rather than spending time on building a model of software.

To summarize, this case study confirmed the need for creation and adaption of modeling tools, as the manual creation of models is complex and time consuming task. In addition, this research confirmed the need for better adaptation of the generated model to minimize the possible differences of manually created and generated models.

4.2.4 Threats to Validity

This section gives an overview about possible threats to validity and reliability of a given case study [35] [37]:

- Low motivation of the students had significant influence on the results of model discovery.
- The involvement of students assumes they are capable to accomplish the delegated task.
- The discovery model of software developer contains mistakes, thus can have negative impact on the interpretation of student results.
- A student's answer was inaccurate or the selection of answer was completed with mistake.
- Experimental validity of this case study – possible repetition of this research or different selection of students may change the results of survey.
- Constructs validity of this survey – We assume the proper measurement of the right things; however, it is possible to commit mistake during validation of students' submission.
- The conclusion validity of this survey – the possibility to make a mistake in conclusion and hypothesis regarding the relationship between observation and the outcome of this case study.

4.3 Summary

In this chapter we provided the validation of thesis software contribution. We described and analyzed the survey of evaluation of SecureUML plugin. Finally, we discussed the differences in manual and automatic methods of RBAC model creation through case study and comparison. In the next chapter we concentrate on the conclusion of this thesis.

5 Conclusion

In this study we analyzed the Role Based Access Control architecture and SecureUML modelling language. We confirmed the compatibility and conformity of RBAC and SecureUML. Next, we selected the stack of technologies and tools for the development of SecureUML plugin and implemented its concepts. Finally, we arranged the empirical research to evaluate and confirm future perspectives of the SecureUML plugin.

5.1 Related Work

Research which is related to this work addresses the different methods and can be categorized into two main groups: forward engineering approach and reverse engineering approach.

The forward engineering approach is represented by work of [41] [42] [43]. Jin X. [41] proposes a framework to provide support for modeling the RBAC system in XACML architecture (RBAC XACML Modeler) and automatic generation of policy specification in XACML format. Her study uses UML Profile mechanism to integrate security to system development cycle but the profile contains both RBAC elements and XACML elements like Rule and Policy, and only contains static separation of duty RBAC constraint. Another research [42] oriented to improve the understanding and articulate the security model and associated polices at the software analysis and design stage via Assurance Management Framework (AMF). They use standard UML to represent access control features of the security model and support policy validation using OCL and a role-based constraint language (RCL2000) and then translate the security model to enforcement code. Thirdly, the study [43] proposed a UML Profile for RBAC, which provides early integration of access control specifications to entire development process. RBAC constraints embedded into UML Profile in order to get use of the strengths of RBAC, such as separation of duties and cardinality constraints. The models to that this profile is applied, can be validated against inconsistency and security constraint violations. A formal language; OCL is used for validation. The proposed UML Profile is lightweight and has a wide-range of CASE-tools support.

The reverse engineering approach is represented by [44]. This research [44] introduces the methodology and PHP2SecureUML tool recovering the RBAC security model from automatically recovered structural and behavioral models of web applications (implemented in php[19] language). They receive the RBAC model in SecureUML of the web application, but the transformation is divided into preparation and model creation phases. This work is close to this research; however, significant differences are in the source code language - Java, and the target nature of their tool, as it requires the input of prepared models to make the initial transformation.

5.2 Answer to Research Questions

Using the accumulated experience and results of this thesis we provide and discuss answers to the research questions:

[19] http://php.net

RQ1: How is it possible to support a Spring web application and improve the understanding of its usage? It is possible to support a Spring web application and improve the understanding of it within proper modeling and visualization of the Spring web application components. This concept was confirmed by representatives of software development community via survey results. Majority of respondents 55% were optimistic about this approach, furthermore, 16% of respondents agreed to this term.

RQ2: How to validate an existing Spring web application for the roles configuration? The validation is achievable in the way of graphical allocation of the role and its related resources, this mapping allows to software developer in the simplified manner to understand and check the correspondence of a role and its actions. Graphical allocation, in fact, is the model of Role Based Access Control policy of a web application. This idea found the proof in results of our survey, 55% of respondents confirmed it; and 38% of respondents almost agreed to this concept.

RQ3: How is it possible to support the software developer and help him/her to understand the security of a Spring web application? It is possible through creation of integrated tools that can be embedded into development environment; and this tool allows building a comprehensive model of a web application on the base of its source code. The resulting model shows the relation between a role and its system usage allowing a software developer to decide whether such construct is allowed or restricted. In case of mistake or a needed for update, the software developer modifies the source code and updates the synchronization of the model that have to reflect the change and support the developer in decision making about validity of the modification. This hypothesis was confirmed by majority of software developers (50%), moreover, 38% of respondents were optimistic about it.

5.3 Future work

Our approach to validate this thesis through empirical research was effective in the scope of validation; moreover, it gave us a good retrospective about existing things and allowed to collect suggestions from software developers. On the top of this input we may plan and continue our study in order to improve and evolve SecureUML plugin and its features. We are looking forward to implement:

- full support for the Spring Security stack of configurations;
- improvements of the constraints support;
- improvements of the diagram supporting the navigation to source code;
- improvements of the plugin configuration and its customization.

6 References

[1] “Secure Software Engineering - ENISA,” European Union Agency for Network and, [Online]. Available: https://www.enisa.europa.eu/topics/critical-information-infrastructures-and-services/internet-infrastructure/secure-software-engineering. [Accessed 25 January 2016].

[2] “OWASP Top Ten Cheat Sheet - OWASP,” OWASP, 28 Septemeber 2015. [Online]. Available: https://www.owasp.org/index.php/OWASP_Top_Ten_Cheat_Sheet. [Accessed 25 January 2016].

[3] M. Woschek, OWASP Cheat Sheets, OWASP, 2015.

[4] “Java - Top 10 Java-based Web Frameworks for 2014-2015 - Bytes Cravings,” Bytes Cravings, [Online]. Available: http://vitalflux.com/java-top-10-java-based-web-development-frameworks-2014-2015/. [Accessed 25 January 2016].

[5] “Web framework rankings | HotFrameworks,” HotFrameworks, [Online]. Available: http://hotframeworks.com/languages/java. [Accessed 25 January 2016].

[6] “Top 4 Java Web Frameworks Revealed: Real Life Usage Data of Spring MVC, Vaadin, GWT and JSF | zeroturnaround.com,” ZeroTurnaround, [Online]. Available: http://zeroturnaround.com/rebellabs/top-4-java-web-frameworks-revealed-real-life-usage-data-of-spring-mvc-vaadin-gwt-and-jsf/. [Accessed 25 January 2016].

[7] “Spring Framework,” Pivotal Software, [Online]. Available: http://projects.spring.io/spring-framework/. [Accessed 25 January 2016].

[8] L. Liu and M. T. Özsu, Eds., Encyclopedia of Database Systems, Springer, 2009.

[9] R. S. Sandhu, E. J. Coyne, H. L. Feinstein and C. E. Youman, “Role-Based Access Control Models,” *Computer,* vol. 29, no. 2, pp. 38-47, February 1996.

[10] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn and R. Chandramouli, “Proposed NIST standard for role-based access control,” *ACM Transactions on Information and System Security,* vol. 4, no. 3, pp. 224-274, August 2001.

[11] I. Ray, N. Li, R. France and D.-K. Kim, “Using UML To Visualize RoleBased Access Control Constraints,” in *SACMAT '04 Proceedings of the ninth ACM symposium on Access control models and technologies,* Yorktown Heights, 2004.

[12] R. Araujo and S. Gupta, Design Authorization Systems Using SecureUML, Foundstone Professional Services, 2005.

[13] T. Lodderstedt, D. Basin and J. 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,* Dresden, 2002.

[14] C. Bontemps, Towards a Model Driven Architecture Approach within an Identity and Access Management Software, Stockholm: KTH Computer Science and Communication, 2012.

[15] C. Hochreiner, Z. Ma, P. Kieseberg, S. Schrittwieser and E. Weippl, “Using Model Driven Security Approaches in Web Application Development,” in *Second IFIP TC5/8 International Conference, ICT-EurAsia 2014, Bali, Indonesia, April 14-17, 2014. Proceedings,* Bali, 2014.

[16] L. Lúcio, Q. Zhang, P. H. Nguyen, M. Amrani, J. Klein, H. Vangheluwe and Y. Le Traon, “Advances in Computers,” *Advances in Computers,* vol. 93, pp. 103-152, 2014.

[17] R. Johnson, Expert One-on-One J2EE Design and Development, Wrox, 2002.

[18] S. Haines, “Mastering Spring MVC,” JavaWorld, 28 April 2009. [Online]. Available: http://www.javaworld.com/article/2078034/spring-framework/mastering-spring-mvc.html. [Accessed 25 January 2016].

[19] C. Yates, S. Ladd, M. Deinum, E. Vervaet, K. Serneels and C. Vanfleteren, Pro Spring MVC: With Web Flow, New York: Apress, 2012.

[20] “Introduction to the Spring Framework,” Pivotal Software, [Online]. Available: http://docs.spring.io/spring-framework/docs/current/spring-framework-reference/html/overview.html. [Accessed 17 March 2016].

[21] “Spring Security,” Pivotal Software, [Online]. Available: http://projects.spring.io/spring-security/. [Accessed 25 January 2016].

[22] C. Scarioni, Pro Spring Security, New York: Apress, 2013.

[23] A. Dikanski, R. Steinegger and S. Abeck, “Identification and Implementation of Authentication and Authorization Patterns in the Spring Security Framework,” in *The Sixth International Conference on Emerging Security Information, Systems and Technologies (SECURWARE 2012),* Rome, 2012.

[24] “About the Eclipse Foundation,” Eclipse Foundation, [Online]. Available: http://www.eclipse.org/org/. [Accessed 25 January 2016].

[25] D. Steinberg, F. Budinsky, M. Paternostro and E. Merks, EMF: Eclipse Modeling Framework, Second Edition, Addison Wesley, 2008.

[26] “JDT Core Component,” Eclipse Foundation, [Online]. Available: http://www.eclipse.org/jdt/core/. [Accessed 25 January 2016].

[27] A. El Kouhen, C. Dumoulin, S. Gérard and P. Boulet, Evaluation of Modeling Tools Adaptation, HAL, 2012.

[28] “Eclipse Modeling Project,” Eclipse Foundation, [Online]. Available: https://eclipse.org/modeling/. [Accessed 25 January 2016].

[29] “Eclipse Modeling Framework (EMF),” Eclipse Foundation, [Online]. Available: https://eclipse.org/modeling/emf/. [Accessed 25 January 2016].

[30] “Graphical Modeling (GMF),” Eclipse Foundation, [Online]. Available: https://eclipse.org/modeling/graphical.php. [Accessed 25 January 2016].

[31] “Graphical Editing Framework (GEF),” Eclipse Foundation, [Online]. Available: https://eclipse.org/gef/. [Accessed 25 January 2016].

[32] V. Vujović, M. Maksimović and B. Perišić, “Sirius: A Rapid Development of DSM Graphical Editor,” in *IEEE 18th International Conference on Intelligent Engineering Systems INES 2014,* Tihany, 2014.

[33] “Sirius,” Eclipse Foundation, [Online]. Available: https://eclipse.org/sirius/overview.html. [Accessed 25 January 2016].

[34] L. K. Klauske, C. D. Schulze, M. Spönemann and R. v. Hanxleden, “Improved Layout for Data Flow Diagrams with Port Constraints,” in *Diagrammatic Representation and Inference, 7th International Conference, Diagrams 2012,* Canterbury, 2012.

[35] M. Ciolkowski, O. Laitenberger, S. Vegas and S. Biffl, “Practical Experiences in the Design and Conduct of Surveys in Empirical Software Engineering,” in *Empirical Methods and Studies in Software Engineering*, Berlin, Springer Berlin Heidelberg, 2003, pp. 104-128.

[36] A. J. Ko, T. D. LaToza and M. M. Burnett, “A practical guide to controlled experiments of software engineering tools with human participants,” *Empirical Software Engineering*, vol. 20, no. 1, pp. 110-141, 2015.

[37] C. Wohlin, M. Höst and K. Henningsson, “Empirical Research Methods in Software Engineering,” in *Empirical Methods and Studies in Software Engineering*, Berlin, Springer Berlin Heidelberg, 2003, pp. 7-23.

[38] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell and A. Wesslén, “Case Studies,” in *Experimentation in Software Engineering*, Berlin, Springer Berlin Heidelberg, 2012, pp. 55-72.

[39] O. A. L. Lemos , F. C. Zanichelli, R. Rigatto, F. Ferrari and S. Ghosh, “Visualization, Analysis, and Testing of Java and AspectJ Programs with Multi-level System Graphs,” in *Software Engineering (SBES), 2013 27th Brazilian Symposium on*, Brasilia, 2013.

[40] A. M. P. Grilo, A. C. R. Paiva and J. P. Faria, “Reverse engineering of GUI models for testing,” in *5th Iberian Conference on Information Systems and Technologies*, Santiago de Compostela, 2010.

[41] X. Jin, Applying Model Driven Architecture approach to Model Role Based Access Control System, Ottawa: University of Ottawa, 2006.

[42] G.-J. Ahn and H. Hu, “Towards realizing a formal RBAC model in real systems,” in *SACMAT '07 Proceedings of the 12th ACM symposium on Access control models and technologies*, Monterey, 2007.

[43] Ç. Cirit and F. Buzluca, “A UML Profile for Role-Based Access Control,” in *SIN '09 Proceedings of the 2nd international conference on Security of information and networks*, Gazimagusa, 2009.

[44] M. H. Alalfi, J. R. Cordy and T. R. Dean, “Recovering Role-Based Access Control Security Models from Dynamic Web Applications,” in *12th International Conference, ICWE 2012*, Berlin, 2012.

Appendix

I. List of Acronyms

GUI	Graphical User Interface
HTTP	The Hypertext Transfer Protocol
NIST	National Institute of Standards and Technology
OCL	Object Constraint Language
OWASP	The Open Web Application Security Project
PIM	Platform Independent Model
PSM	Platform Specific Models
RBAC	Role Based Access Control
URL	Uniform Resource Locator
XACML	eXtensible Access Control Markup Language
XMI	XML Metadata Interchange

II. Survey: Questionnaire

Please fill the questionnaire form.

General:	
1. What is your development experience (years)?	
2. What is your development experience with Spring (years)?	
3. What is your development experience with Spring Security (years)?	
4. Have you had Spring Security Roles misconfiguration (yes/no)?	
5. Do you have Eclipse IDE experience (yes/no)?	
6. Do you have UML experience (yes/no)?	

Share your opinion on modeling, rate the statement (0-1-2-3-4):	**Disagree**		**Neutral**		**Agree**
7. The modeling is important in software development.	0	1	2	3	4
8. SecureUML is easy to understand.	0	1	2	3	4
9. SecureUML is easy to learn.	0	1	2	3	4
10. Generation of models from source code is useful for me.	0	1	2	3	4
11. Modeling of software development can help me in everyday work.	0	1	2	3	4

Share your opinion on SecureUML Plugin, rate the statement (0-1-2-3-4):	**Disagree**		**Neutral**		**Agree**
12. SecureUML Plugin is easy to use.	0	1	2	3	4
13. SecureUML Plugin is easy to learn.	0	1	2	3	4
14. SecureUML Plugin helps to understand the RBAC model.	0	1	2	3	4
15. SecureUML Plugin helps to understand the usage of resources.	0	1	2	3	4
16. SecureUML Plugin improves the understanding of a web application.	0	1	2	3	4
17. SecureUML Plugin helps to improve the security of a web application.	0	1	2	3	4
18. I would like to try the SecureUML Plugin.	0	1	2	3	4

Please give a feedback.

Did you like presentation?

Do you have ideas/suggestions/feature requests?

Do you have a name suggestion for SecureUML Plugin?

Thank you!

III. Survey: Results

The raw results of questionnaire responses are shown in the Table 4.

Table 4. Raw results of questionnaire responses.

#	Question answers 1 – 18																	
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Plugin presentation at Helmes[20]																		
1	10	3	3	N	Y	Y	4	3	3	1	1	1	3	2	1	2	2	2
2	7	3	3	Y	Y	Y	3	3	3	4	3	4	4	4	4	4	4	4
3	15	0	0	N	Y	Y	4	3	3	3	4	4	3	4	4	4	4	0
4	10	3	1	Y	Y	Y	4	4	4	4	4	4	4	4	3	3	4	4
5	10	3	1	N	Y	Y	2	2	2	3	3	4	3	4	3	3	4	4
6	12	5	5	Y	N	Y	4	4	4	4	4	4	4	4	3	2	2	4
7	10	4	4	N	Y	Y	4	2	3	3	2	2	1	3	3	3	3	3
8	1,5	1	0,5	Y	Y	Y	4	4	4	3	3	4	4	3	4	2	3	3
9	1	1	1	N	N	Y	3	3	3	3	3	4	4	3	4	3	4	4
10	1	1	1	N	Y	Y	4	3	3	4	3	4	3	4	3	4	4	3
11	12	4	3	Y	Y	Y	4	3	4	3	3	4	3	4	3	3	4	4
Video[21] presentation of plugin																		
12	6	3	2	Y	Y	Y	3	3	4	3	3	4	4	4	3	3	3	4
13	9	5	3	Y	Y	Y	3	4	3	2	2	3	3	4	2	3	3	4
14	3	1	1	Y	N	N	3	3	3	4	3	4	3	4	3	2	3	3
15	3	1	1	N	Y	Y	4	4	4	4	4	3	3	3	4	3	3	3
16	8	5	3	Y	Y	Y	4	4	4	3	4	4	3	3	4	3	3	3
17	6	2	2	N	Y	Y	3	3	3	2	2	3	4	3	3	3	4	3
18	2	1	1	N	Y	N	3	2	3	3	2	4	4	3	2	2	4	4

[20] http://www.helmes.ee
[21] https://youtu.be/G4qp5wRmxHI

The analysis of survey results allows building the quantitative model represented by charts for every question, showing the number of responses and respondent's attitude to the question subject.

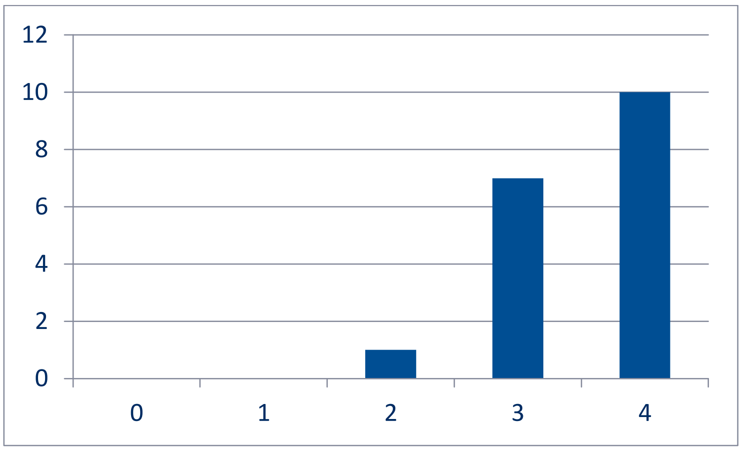


Figure 39. The score of results for question 7 (x axis – number defining the attitude of a respondent; y axis defines the number of responses).

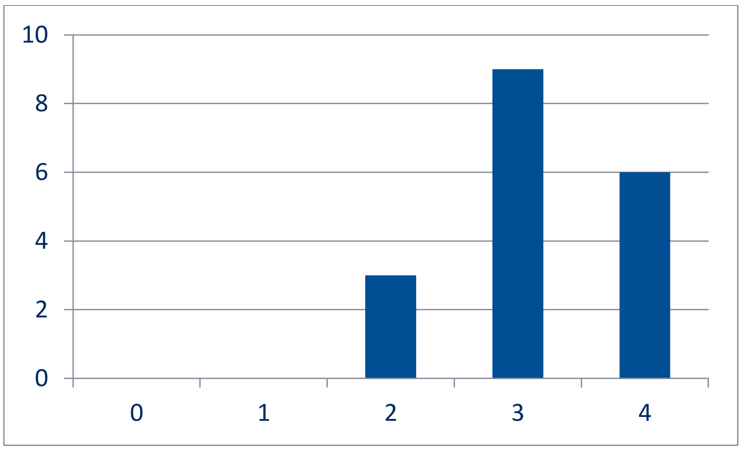


Figure 40. The score of results for question 8 (x axis – number defining the attitude of a respondent; y axis defines the number of responses).

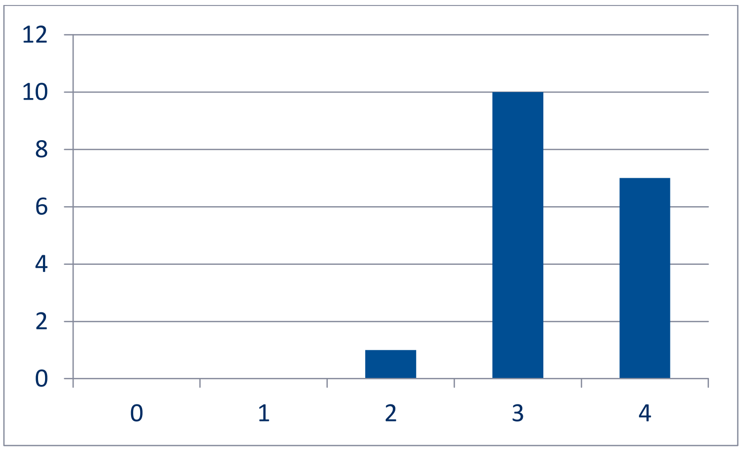


Figure 41. The score of results for question 9 (x axis – number defining the attitude of a respondent; y axis defines the number of responses).

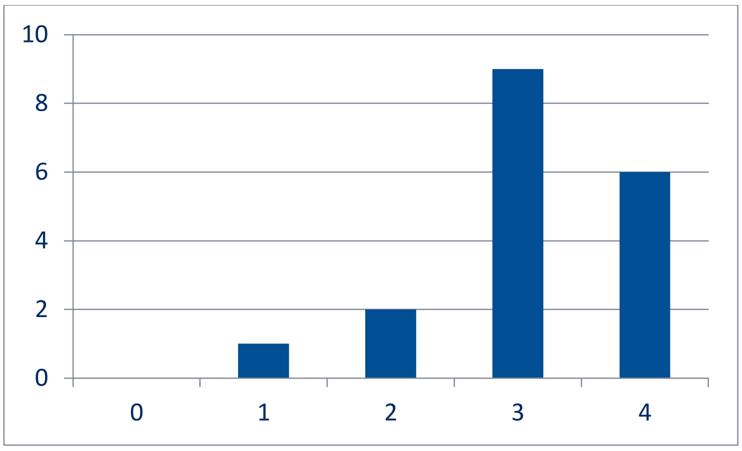


Figure 42. The score of results for question 10 (x axis – number defining the attitude of a respondent; y axis defines the number of responses).

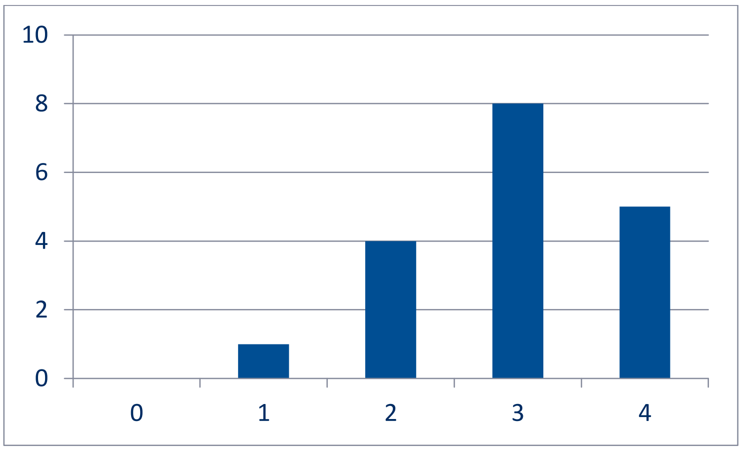


Figure 43. The score of results for question 11 (x axis – number defining the attitude of a respondent; y axis defines the number of responses).

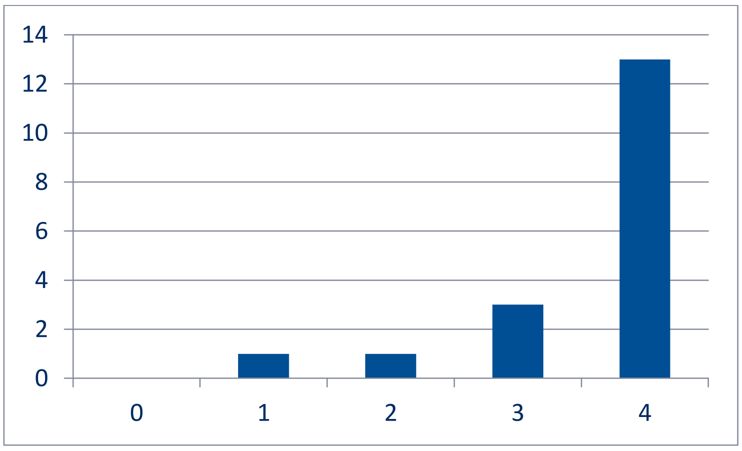


Figure 44. The score of results for question 12 (x axis – number defining the attitude of a respondent; y axis defines the number of responses).

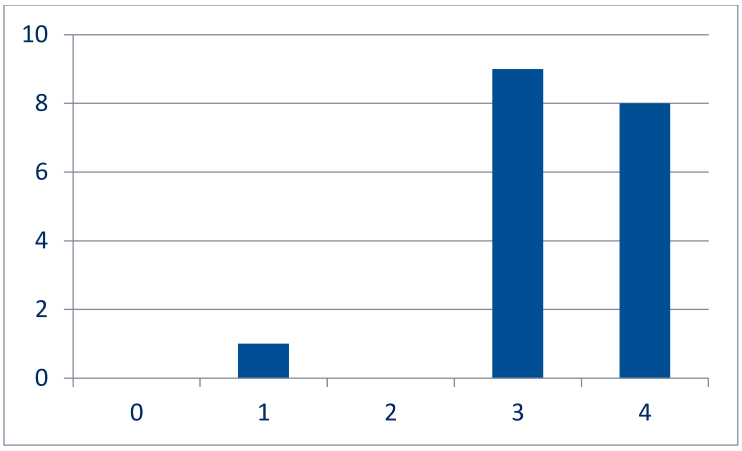


Figure 45. The score of results for question 13 (x axis – number defining the attitude of a respondent; y axis defines the number of responses).

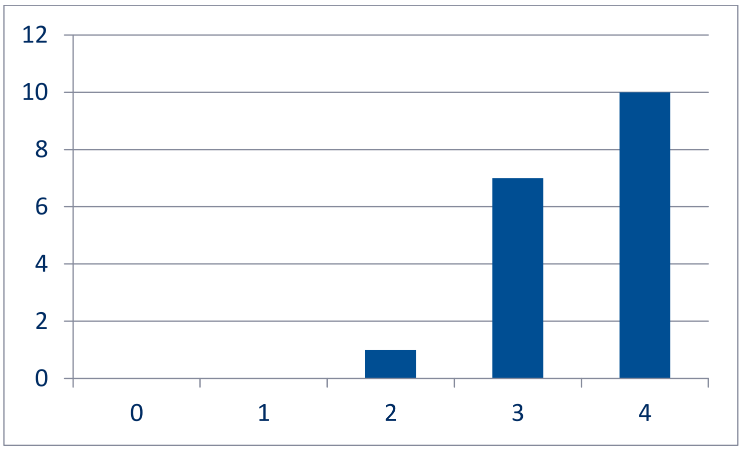


Figure 46. The score of results for question 14 (x axis – number defining the attitude of a respondent; y axis defines the number of responses).

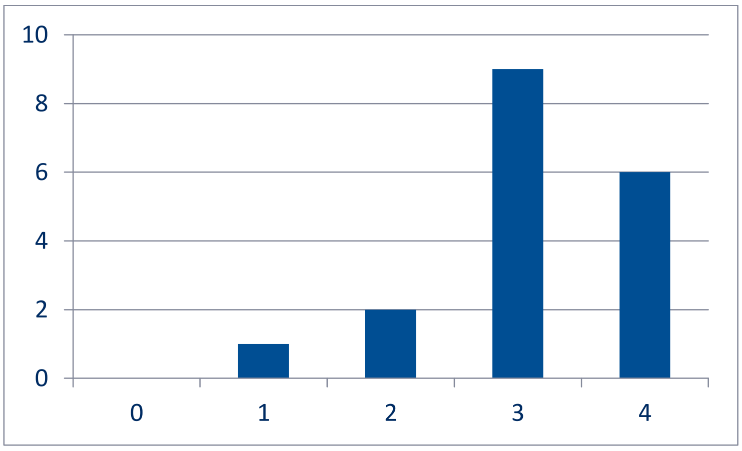


Figure 47. The score of results for question 15 (x axis – number defining the attitude of a respondent; y axis defines the number of responses).

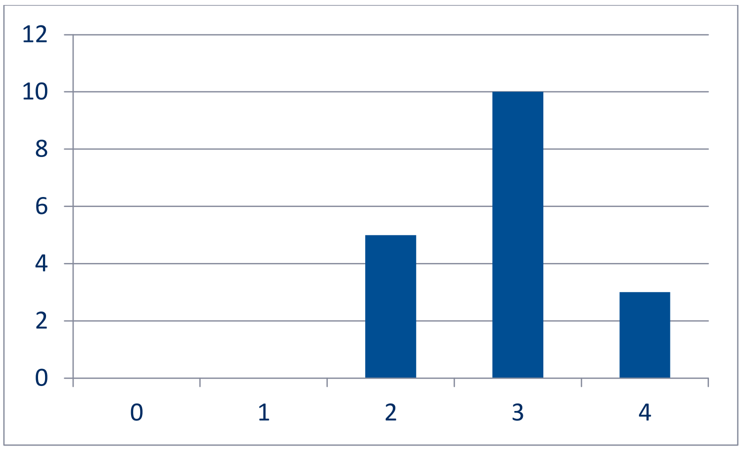


Figure 48. The score of results for question 16 (x axis – number defining the attitude of a respondent; y axis defines the number of responses).

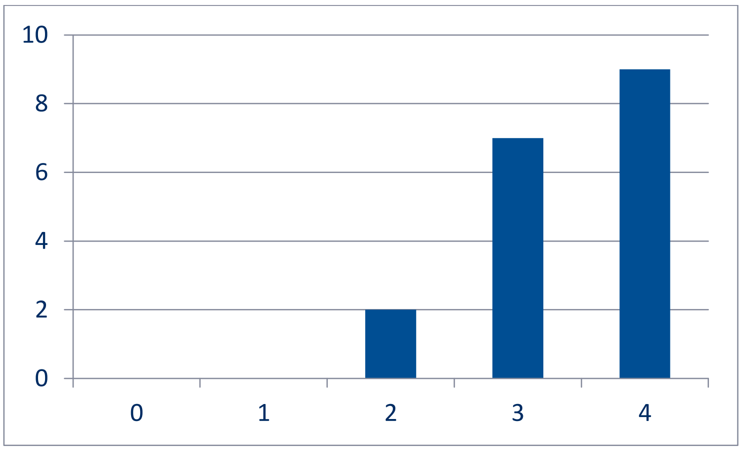


Figure 49. The score of results for question 17 (x axis – number defining the attitude of a respondent; y axis defines the number of responses).

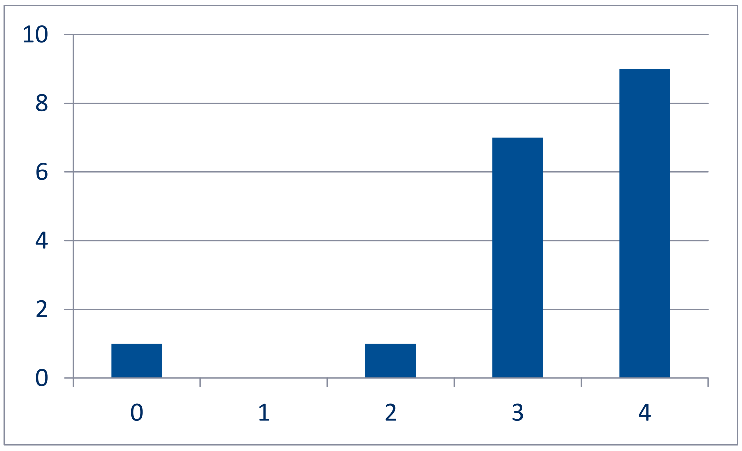


Figure 50. The score of results for question 18 (x axis – number defining the attitude of a respondent; y axis defines the number of responses).

IV. Case Study: Student Task

Let's analyze the Forum Web Application provided at http://c37-76.uvn.zone.eu:8080/XXXXXXXX/.

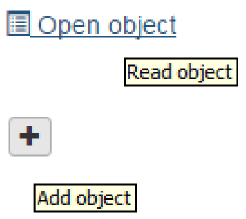
This web page represents a part of forum software allowing users to share and exchange information by creating threads, topics and posts. Usually forum has users with various responsibilities and privileges. For example, the hierarchy of forum users might be the following:

- Administrator is in charge for forum’s sections (threads), their definition and management;
- Moderator is for forum's sub-sections (topics), their management, content formation and validation;
- Users are typically the biggest target group of a forum application, they are consumers and actors of forum’s knowledge base with abilities to propose the information and to evaluate its quality;
- Guests are all unregistered users with limited privileges; usually they have a read only access to forum's public resources.

Your task is to define the SecureUML model representing role-based access control policy regarding the data gathered form this website.

Task notes:

- Consider links and buttons as actions (action is described within tooltip):



An example of link and button with tooltips.

- Ignore any login/logout functionality and concentrate on thread(s).
- Consider removal of objects as your last step as you cannot create/restore all objects.

Analyzing, consider the following questions:

- What are object and its concerned attributes?
- What operations do change the values of the attributes?
- What are roles?
- What are the security actions?
- What are the permissions of roles towards the object?
- What are the users?

Credits for website usage:

- Guest
- jack-XXXXXXXX@c37-76.uvn.zone.eu // 123456
- john-XXXXXXXX@c37-76.uvn.zone.eu // 654321

V. Case Study: Results

Table 5. The resulting matrix of validation of student submissions.

#	M 1	M 2	M 3	M 4	M 5	M 6	M 7	M 8	M 9	M 10	M 11	M 12	M 13	M 14	M 15	M 16	M 17	M 18	M 19
1	0	1	0	1	1	0	1	1	1	1	1	0	0	0	1	1	1	0	1
2	0	1	1	0	1	1	1	1	1	1	1	0	0	1	1	1	1	1	0
3	0	1	0	0	1	1	1	1	1	1	1	0	0	0	1	1	1	1	1
4	0	1	0	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
5	1	1	0	0	0	0	1	1	1	1	1	0	0	0	1	1	1	0	0
6	0	1	0	0	1	1	1	1	1	1	1	0	0	0	1	1	1	1	0
7	1	1	1	1	0	1	1	1	1	1	1	1	0	1	1	1	1	1	0
8	0	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0
9	1	1	1	1	0	1	1	1	1	1	1	0	0	0	1	1	1	1	0
10	0	1	1	0	0	1	1	1	1	1	1	0	0	0	1	1	1	0	1
11	0	1	1	0	0	0	0	1	1	1	1	0	0	1	1	1	1	1	0
12	0	1	0	0	0	0	1	1	1	1	1	0	0	1	1	1	1	0	0
13	0	1	0	0	1	1	0	1	1	1	1	0	0	1	1	1	1	0	0
14	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0
15	0	1	0	0	1	0	0	1	0	0	0	0	0	0	1	1	1	0	1
16	1	1	0	1	1	1	0	1	1	1	1	0	0	1	1	1	1	1	1
17	0	1	0	0	1	1	0	1	1	1	1	0	0	1	1	1	1	1	0
18	0	1	1	0	0	0	1	1	1	1	1	0	0	0	1	1	1	0	0
19	1	0	0	1	0	0	0	1	0	1	1	0	0	1	1	1	1	1	1
20	0	1	1	0	0	1	1	0	1	1	1	0	0	0	1	1	1	1	0
21	1	0	0	1	0	0	1	1	1	1	1	0	0	1	1	1	1	1	0
22	0	1	1	0	1	1	1	1	1	1	1	0	0	0	1	1	1	1	1
23	1	1	1	0	0	0	0	1	1	1	1	0	0	0	1	1	1	0	0

The stacked column chart representation of detected and undetected model features (M1 – M18) is presented in the Figure 51.

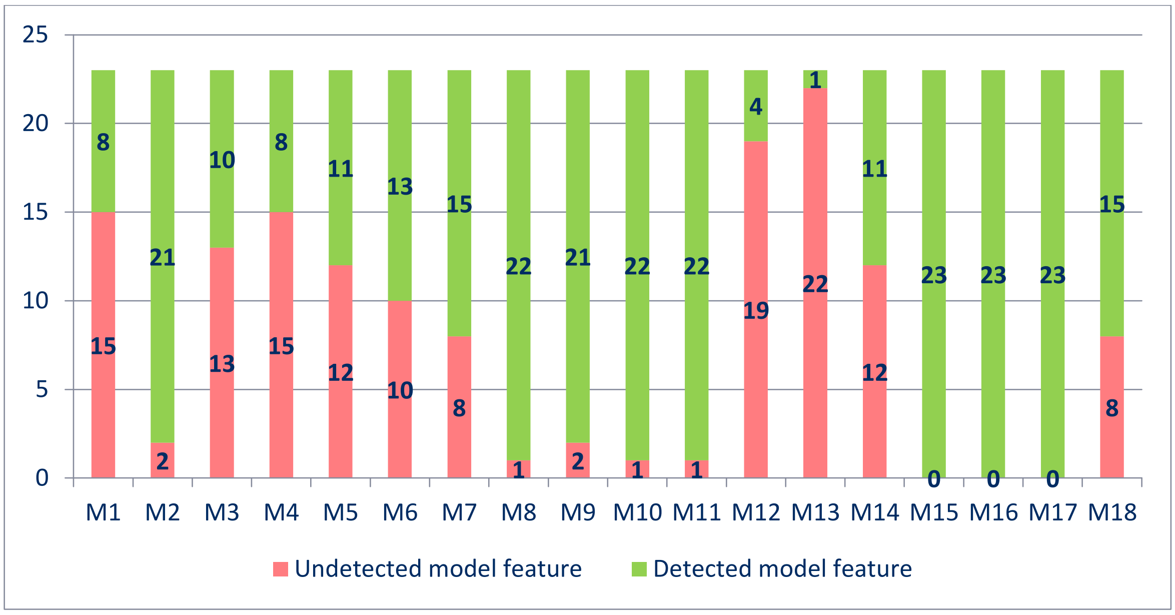


Figure 51. The stacked column chart representation of detected and undetected model features (M1 – M18) (x axis – model feature identifiers; y axis defines the number of student submissions).

VI. Source Code

The source code of software contribution is located in public repository of Bitbucket[22].

Eclipse plugin contribution:

https://bitbucket.org/andrey_secureuml/secureuml_plugin

Test web application:

https://bitbucket.org/andrey_secureuml/secureuml_webapp

Student web application:

https://bitbucket.org/andrey_secureuml/secureuml_webapp_student

[22] https://bitbucket.org/

VII. License

Non-exclusive license to reproduce thesis and make thesis public