Secure Web Engineering Supported by an Evaluation Framework[*]
Preliminary Report on a Web Engineering Approach for Secure Applications Supported by a Conceptual Evaluation Framework for Secure Systems Engineering

Marianne Busch
Ludwig-Maximilians-Universität München, Oettingenstraße 67, 80538 München, Germany

1 STAGE OF THE RESEARCH

This paper reports on the progress of the author's PhD in the area of security engineering for web applications. Initially, the work was located at the beginning of the Software Development Life Cycle (SDLC) with a focus on design. However, designing a perfectly secure application is worth nothing, if it is not possible for security engineers to choose appropriate methods, notations and tools (so called *mechanisms*) to work with in each phase of the SDLC. Therefore, we[2] additionally develop a conceptual framework for the evaluation of these Mechanisms, which is not limited to the web, and also has a focus on security.

At the moment, almost two-thirds of the work for the PhD is done, which means that most underlying ideas are written down but further case studies and evaluations will follow.

2 OUTLINE OF OBJECTIVES

Daily news tells that web applications are often not secure enough, which is a threat to the user's privacy as well as to the image of companies. Our *first aim* is to make web applications more secure by taking security features into account at the very beginning of the SDLC. During the requirements and design phases, graphical or textual models can help to get an overview of a web application and its security features. Besides, models can be used for documentation purposes, and security-related properties in models can be validated and transformed to artifacts for the implementation phase. We extend an existing modeling approach in a way that general security features as secure connections, authentication and access control on data structures can be represented. Additionally, we focus on specific security features, as access control on the navigation structure of a web application or automated reactions to denial-of-service attacks.

Collecting and analyzing data of existing security engineering methods, notations and tools (*mechanisms*) is of major importance for security and software engineers, as it helps them to take decisions about solutions for upcoming tasks. These tasks can be related to the design of web applications, but can as well go beyond web engineering.

To ease the tasks of recording own results and of getting an overview of existing methods, notations and tools the Common Body of Knowledge[3] was implemented as a semantic Wiki within the scope of the EU project NESSoS. As we are members of this project, we gained experience working with the CBK and its underlying ontology and reflected on requirements for a conceptual evaluation framework. Our *second aim* is to provide an approach for the evaluation of methods, notations and tools for the engineering of secure software systems. Evaluation should also be possible for vulnerabilities, threats and security properties (e.g., integrity). The term "evaluation" covers the collection, analysis and finer-grained representation of (security-specific) knowledge. Another requirement is that the core framework is not overly detailed, but easy to extend.

3 RESEARCH PROBLEM

In this section, research questions are prompted, which – to the best of our knowledge – were not fully answered before the author's work started in 2011.

Generally, secure web applications can be mod-

[*]I like to thank my PhD supervisor Martin Wirsing and my colleague Nora Koch for their support and for fruitful discussions. This work has been supported by the EU-NoE project NESSoS, GA 256980.

[2]The term "we", as used in this paper, refers to a team effort, however, with substantial contributions by the author.

[3]CBK. http://nessos-project.eu/cbk

eled on a high-level of abstraction, as UML-based Web Engineering (UWE) (Koch et al., 2008; Busch et al., 2011) or on a level where all information for an implementation is given, as usually done with ActionGUI (Basin et al., 2011). Advantages of a higher level of abstraction are that modeling of modern web applications does not take too much time, quickly provides an overview and the engineer can go into details whenever needed. Advantages of lower level of abstraction are that, ideally, the code of a web application can be fully generated from the model, although generated applications are severely limited by the power of the modeling language.

A large majority of companies strives for attracting customers by using leading-edge technology in their web pages. Limitations are unwanted, despite the prospect of web applications that are more secure, if modeled thoroughly. This is why the author started to extend UWE for security features in her Master's thesis (Busch, 2011). Beyond the Master's thesis, interesting questions are:

- How can a high-level modeling language as UWE be extended so that the result is useful, security-specific and more closely related to the web?
- Although not the whole source code can be obtained, which artifacts can be generated?
- Which benefits can be achieved by model-checking and testing of artifacts?
- Is UWE's graphical notation enough or might it be useful to create a textual version?
- If engineers start modeling with UWE, but later decide that complete model generation is important in a certain situation, how can UWE models be transformed to ActionGUI?
- Can UWE's security extensions be added to other web modeling languages, as to (IFML, 2013)?

A challenge is to understand the domain of security engineering, not only regarding security modeling, but also regarding the creation of an evaluation framework. For the latter, issues to be examined are:

- How could security-related features, as security properties, vulnerabilities and threats, become first-class citizens and how could the dependences between *mechanisms* (method, notations and tools) and security features be modeled best?
- How would an ontology look like, which can be used for finding answers to research questions beyond comparing general *mechanisms*?
- How is the process of data collection and data analysis specified, to make sure that emerging research results are comprehensible and valid?

4 STATE OF THE ART

This chapter introduces approaches for secure web engineering as well as evaluation approaches.

4.1 Secure Web Engineering

Many web engineering approaches exist, which do not allow to model security features, as e.g., OOHRIA (Meliá et al., 2008), OOWS (Valverde and Pastor, 2008), WebML (Brambilla and Fraternali, 2013) and (IFML, 2013), which is based on WebML.

Security-aware modeling approaches are briefly introduced in the following [description adapted from (Busch et al., 2011; Busch et al., 2013)].

Our approach is based on the UWE LANGUAGE (Koch et al., 2008; Busch et al., 2011). One of the cornerstones is the "separation of concerns" principle using separate models for different views. However, we can observe that security features are cross-cutting concerns which cannot be separated completely. Since the extension of the author's Master thesis (Busch, 2011), main UWE models have been:

The Requirements Model. defines (security) requirements for a project.

The Content Model. contains the data structure used by the application.

The UWE Role Model. describes a hierarchy of user groups to be used for authorization and access control issues. It is usually part of a *User Model*, which specifies basic structures, as e.g., that a user can take on certain roles simultaneously.

The Basic Rights Model. describes access control policies. It constrains elements from the *Content Model* and from the *Role Model*.

The Presentation Model. sketches the web application's user interface.

The Navigation State Model. defines the navigation flow of the application and navigational access control policies. The former shows which possibilities of navigation exist in a certain context. The latter specifies which roles are allowed to navigate to a specific state and the action taken in case access cannot be granted. In a web application such actions can be, e.g., to logout the user and to redirect to the login form or just to display an error message. Furthermore, secure connections between server and browser are modeled.

For each view, an appropriate type of UML diagram is used, e.g., a state machine for the navigation model. In addition, the UWE Profile adds a set of stereotypes, tag definitions and constraints, which can be downloaded from the UWE website (LMU, 2013). Stereotypes can then be applied to UML model elements and values can be assigned to tags, which are associated to at least one stereotype.

ACTIONGUI (Basin et al., 2010) is an approach for generating complete, but simplified, data-centric web applications from models. It provides an OCL specification of all functionalities, so that navigation is only modeled implicitly by OCL constraints. As described in Section 3, ActionGUI abstracts less from an implementation than UWE does.

UMLSEC (Jürjens, 2004) is an extension of UML with emphasis on secure protocols. It is defined in form of a UML profile including stereotypes for concepts like authenticity, freshness, secrecy and integrity, role-based access control, guarded access, fair exchange, and secure information flow. In particular, the use of constraints gives criteria to evaluate the security aspects of a system design, by referring to a formal semantics of a simplified fragment of UML. UMLsec models, compared to UWE models, are extremely detailed and therefore quickly become very complex. Tool support is only partly adopted from UML1.4 to UML2. However, the new tools[4] have not been updated for almost two years.

SECUREUML (Lodderstedt et al., 2002) is a UML-based modeling language for secure systems. It provides modeling elements for role-based access control and the specification of authorization constraints. A SecureUML dialect has to be defined in order to connect a system design modeling language as, e.g., ComponentUML to the SecureUML metamodel, which is needed for the specification of all possible actions on the predefined resources. In our approach, we specify role-based execution rights to methods in a basic rights model using dependencies instead of the SecureUML association classes, which avoids the use of method names with an access related return type. However, UWE's basic rights models can easily be transformed into a SecureUML representation.

A similar approach is UACML (Slimani et al., 2011) which also comes with a UML-based meta-metamodel for access control, which can be specialized into various meta-models for, e.g., role-based access control (RBAC) or mandatory access control (MAC). Conversely to UWE, the resulting diagrams of SecureUML and UACML are overloaded, as SecureUML uses association classes instead of dependencies and UACML does not introduce a separate model to specify user-role hierarchies.

There is a set of approaches that address modeling of security aspects of service-oriented architectures (SOAs), such as the SECTET framework (Hafner and Breu, 2008), UML4SOA (Gilmore et al., 2011), and SecureSOA (Menzel and Meinel, 2009). The first one proposes the use of sequence diagrams for the representation of a set of security patterns, in UML4SOA security features are modeled as non-functional properties using class diagrams, and the latter relies on FMC block diagrams and BPMN notation.

[4]UMLsec tools. http://carisma.umlsec.de

4.2 Evaluation Approaches

Evaluation approaches are often tailored to the needs of a specific area. We start by introducing general approaches and continue with security-specific ones. This section is adapted from (Busch and Koch, 2013).

General Evaluation Approaches. KITCHENHAM et al. (Kitchenham and Charters, 2007) specify so called "Systematic Literature Reviews" in software engineering. The aim is to answer research questions by systematically searching and extracting knowledge of existing literature. Our evaluation framework, called SECEVAL, is based on their work, however SECEVAL is not restricted to literature reviews. We focus on the use of arbitrary resources, as source code or experiments which are carried out to answer a security-related research question. In contrast to Kitchenham's approach, the process we define is generic and thus allows us to refine the way of searching after first results indicate a worthwhile direction for further research.

SIQINU (Becker et al., 2013) is a framework for evaluating the quality of a product version which can then be improved. It uses the conceptual framework C-INCAMI, which specifies concepts and relationships for measurement and evaluation. SIQinU defines a strategy using UML activity diagrams whereas C-INCAMI is specified by a UML class diagram.

MOODY (Moody, 2003) proposes an evaluation approach which is based on experiments. Practitioners use methods and afterwards answer questions about perceived ease of use, perceived usefulness and intention to use. A figure how Moody's approach can be integrated can be found online (Busch, 2013).

The CBK (Common Body of Knowledge) (Beckers et al., 2012) defines an ontology for software engineers to describe Knowledge Objects (KOs), which are methods, techniques, notations, tools or standards. Techniques are methods which do not specify activities for applying the method. The CBK is implemented as a semantic Wiki and serves as a knowledge base containing all relevant information about existing KOs. Unlike the CBK, SECEVAL is not implemented yet. In contrast to the CBK, SECEVAL focuses on security-related features and provides a fine-grained ontology, which explicitly considers the phases of the SDLC. Additionally, it defines a process for the evaluation of KOs.

Security-specific Evaluation Approaches. Security-related frameworks often consider concrete software systems for their evaluation. An example is the OWASP RISK RATING METHODOLOGY[5], where the risk for a concrete application or system is estimated. For SECEVAL, we added vulnerability-dependent features of the OWASP model, as e.g., the difficulty of detecting or exploiting a vulnerability. Features that are related to a concrete system and the rating of a possible attack are introduced as an extension of SECEVAL, which can be found online (Busch, 2013).

The i* metamodel[6] is the basis of a vulnerability-centric requirements engineering framework introduced in (Elahi et al., 2010). The extended, VULNERABILITY-CENTRIC I* METAMODEL aims at analyzing security attacks, countermeasures, and requirements based on vulnerabilities. The metamodel is represented using UML class models.

Another approach which is focused on vulnerabilities is described by Wang et al. (Wang and Guo, 2009) Their concept model is less detailed than the i* metamodel. Their aim is not to describe reality by using graphical models, but to create a knowledge base which can then be queried using a language for the semantic web, called SWRL.

5 METHODOLOGY

To answer the research questions, listed in Section 3, the following methodological steps are used.

To conduct a LITERATURE REVIEW, similar to Kitchenham et al. (Kitchenham and Charters, 2007), helps to get into the topic at the beginning and makes it easy to stay up-to-date during the work on the PhD. To prove the applicability of SECEVAL, we plan to graphically model the knowledge gathered in the literature review regarding (security) modeling for web applications.

To fully understand the research questions that we want to answer, it is necessary to ELICIT REQUIREMENTS for the UWE extension as well as for SECEVAL. Additionally, small CASE STUDIES can show, if new ideas are reasonable and consistent to the whole approach. Towards the end of the PhD, big case studies are going to make sure that our approach scales and that common, as well as exceptional situations can be modeled.

Asking for FEEDBACK is crucial, be it by submitting papers or by conducting interviews. To get feedback from international senior researchers, who are experts in various areas of security engineering, we combined a questionnaire with live-explanations and discussions, which was very useful for the development of SECEVAL. Regarding UWE, it is interesting to compare the amount of errors in bigger web applications, which are modeled with UWE, with those which are implemented traditionally. Unfortunately, such a big empirical study is out of the scope of the author's PhD[7], but feedback from students, which are working with UWE in their bachelor or master's theses partly fills this gap, along with the validation by using case studies.

6 EXPECTED OUTCOME

In this section, an overview of preliminary and expected results for secure web engineering and for our security framework SECEVAL is given.

6.1 Secure Web Engineering

We briefly introduce a small case study from (Busch et al., 2013), named SmartGrid Bonus Application, before (expected) results are presented, using the same order as in Section 3. Another case study about patient monitoring can be found in (LMU, 2013) and additionally, we are working on a big case study about the customer web interface in a smart grid home. Both case studies are built upon requirements from SIEMENS, which is a partner in the NESSoS project. Most tools and prototypes, which are mentioned in this section can also be found at the UWE web page (LMU, 2013).

Of all UWE models, introduced in Section 4, the basic rights model and the navigation state model are most important for modeling security features. Our case study is a prototype of an energy offer management system including optional bonus handling. It provides two different user roles namely provider and customer: providers manage and sell energy packages including optional bonus programs for customers, as depicted in Figure 1. Customers have the possibility to buy offered energy packages. Therefore, our application lists all available energy offers and the customer selects a specific offer which includes a bonus code. After buying an energy package, the application

[5] Risk Rating Methodology. https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology

[6] i* notation. http://istar.rwth-aachen.de/

[7] Advice how to do this in a small way are welcome.

shows the corresponding bonus code which contains a gift voucher, e.g., for online shops. Finally, the customer gets a confirmation for the ordered energy.

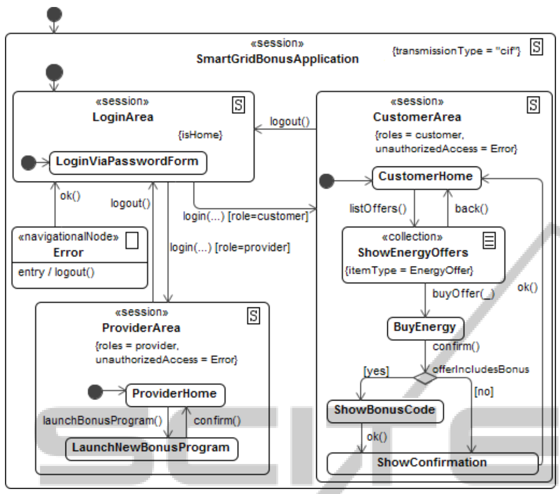

Figure 1: UWE: Navigation States Example.

In order to model the data structure managed by our case study, we use UWE's CONTENT MODEL. Basically, it comprises two domain classes, `EnergyOffer` and `BonusProgram`, which are also used in Figure 2. An instance of the class `EnergyOffer` represents a specific energy offer launched by an energy provider including start and end date. Each object of `EnergyOffer` can include an arbitrary number of `BonusProgram` instances. A `BonusProgram` instance stands for an additional bonus customers get, after they have bought the corresponding `EnergyOffer`.

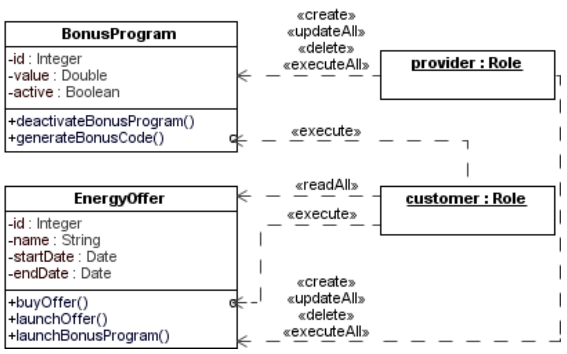

Figure 2: UWE: Basic Rights Example.

In order to model RBAC constraints we use UWE's BASIC RIGHTS MODEL, depicted in Figure 2. Basically, it uses classes of the CONTENT MODEL on the left-hand side in combination with user roles on the right-hand side. Access permissions were defined by stereotyped dependencies: for our application, a provider has no restricting constraints. By contrast, there is only a limited set of permissions for users taking on the role of a customer: they are only allowed to read instances of the class `EnergyOffer` and to call the methods `buyOffer()` and `generateBonusCode()`.

At the moment, we are working on more web-security specific content, which is not shown in the example above. It will e.g. be possible to use UWE for modeling application behavior in case of panic- or under-fire mode. The panic mode is useful for tumultuous regions around the world, where users might be physically forced to sign in an online service to perform certain tasks. This can e.g., be mitigated by a second password, which changes the web application in a predefined way. The under-fire modeling elements refer to the application's behavior under denial-of-service attacks.

Artifacts which can be extracted from UWE models are e.g. RBAC rules. We use the model to text transformation language XPand to transform UWE models to XACML[8] and to code snippets. For the latter, we implemented a prototypic transformation of the data structure, roles and RBAC rules to Apache Wicket with Apache Shiro and Hibernate (Wolf, 2012).

Exporting XACML policies, which can include RBAC policies from the Basic Rights model as well as from the navigational states model, is implemented in a tool called UWE2XACML. In (Busch et al., 2012), it is also explained how XACML can be transformed in FACPL, a formal policy language with the advantage of fully specified semantics.

Additionally, XACML can be used to generate tests by using a tool chain, including UWE2XACML, X-CREATE [9] for generating XACML requests, a PDP (Policy Decision Point which responds to XACML requests), as suggested in (Bertolino et al., 2013). The advantage is that policies are modeled at a high level of abstraction so they are easy to understand and to maintain, whereas policies written in XACML tend to become lengthy and error-prone.

Constraints like "a customer can only get access to a bonus code after he bought an energy package" can be inferred from navigational state models (cf. transitions in Figure 1). To extract so called Secure Navigation Paths (SNPs) and to use them both for automatic testing and for the generation of a monitor, which shields the web application from illicit access sequences, is described in (Busch et al., 2013). The associated tool MagicSNP is an extension of a plugin for MagicDraw 17.0[10], called MagicUWE, which we

[8]XACML 2.0. http://docs.oasis-open.org/xacml/2.0/
[9]X-CREATE. http://labsedc.isti.cnr.it/tools/xcreate
[10]MagicDraw. http://magicdraw.com/

implemented to ease the creation of UWE models.

Another transformation we considered is the model-to-model transformation from UWE to ActionGUI, described in (Busch and García de Díos, 2012).

Some engineers prefer textual models and as this is a matter of taste and both types have advantages and disadvantages, we aim at providing a Domain Specific Language (DSL) for UWE, called TEXTUALUWE. The focus is on creating an intelligible language, which is not only easy to read but also exploitable by algorithms. As we want to open the way to expressive algorithms, we decided to use Scala[11], a multi-paradigm programming language. For our work we use the functional style, as it allows writing short and precise algorithms to support TextualUWE. We are working on algorithms to check security features of TextualUWE models, as e.g., which part of the web application can be reached by a user which is associated to a certain role. Further verifiable features are to find inconsistencies in the model or to check what happens when parts of pages (so-called navigation nodes) are illegally accessed.

A transformation from graphical UWE models to TextualUWE is currently under construction (Rzehaczek, 2013) – it works for the small case study shown above. For implementing the transformation we use Acceleo, as it is mightier than XPand. It transforms MagicDraw projects, exported using the XML Metadata Interchange (XMI) format, to our DSL.

As soon as we added all web specific security features we plan to integrate, we are going to use the concepts for IFML and report on needed extensions, due to the fact that the navigation structure cannot be modeled in IFML.

6.2 Conceptual Evaluation Framework

Our evaluation framework SECEVAL provides a structure for data as well as a structure for performing a data analysis on the collected data (i.e., methods, notations and tools in the context of security properties, vulnerabilities and threats). We grouped these concepts in three packages: Security Context, Data Collection and Data Analysis. Figure 3 shows SECEVAL's ontology represented as a UML class diagram. We selected UML, as we think it fits our needs best. Deliverable D2.4 (Busch and Koch, 2013) of the NESSoS project includes a detailed description of SECEVAL. This section is adapted from D2.4 and focuses exemplarily on the security context.

The SECURITY CONTEXT package is used to specify the object-oriented data structure we use for

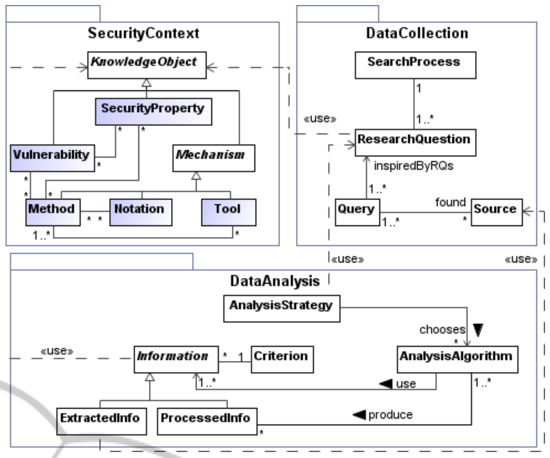

Figure 3: SECEVAL: Model Overview.

describing methods and related security features, notations and tools. The classes `Method`, `Notation` and `Tool` inherit general attributes, as e.g., a problem description and whether or not it is based on standards, from the abstract class `Mechanism`. Additionally, a tool can support methods and a notation can be used for several methods.

Having a well-defined data structure which is extensible is one thing, collecting concrete data the other. The package DATA COLLECTION contains the search process as well as research questions which should be investigated.

Research questions define what is inside and outside the scope of research. Before specifying the research questions, a basic understanding of the context is needed, therefore a dependency, stereotyped by "use", points from `ResearchQuestion` to the package `SecurityContext`. Queries can be defined accordingly. They are used to find matching sources containing data which might help to answer the research questions. Please note that for us the term "research question" does not have to refer to scientific research questions.

After collecting data, the source data may consist of, e.g., some papers, several websites or code. The process of extracting information from the data is called DATA ANALYSIS. An analysis strategy defines criteria which are important for answering the research question, like costs or preconditions. These criteria are used to classify information which is extracted from the sources. Information can also be processed, for example one could calculate annual costs for tools. The instructions used for those calculations are referred to as `AnalysisAlgorithm` in Figure 3. Each piece of information (processed or not) can relate to classes of the security context model, e.g., a tool can be described using information from the website of the tool. However, information does not have

[11] Scala. http://scala-lang.org

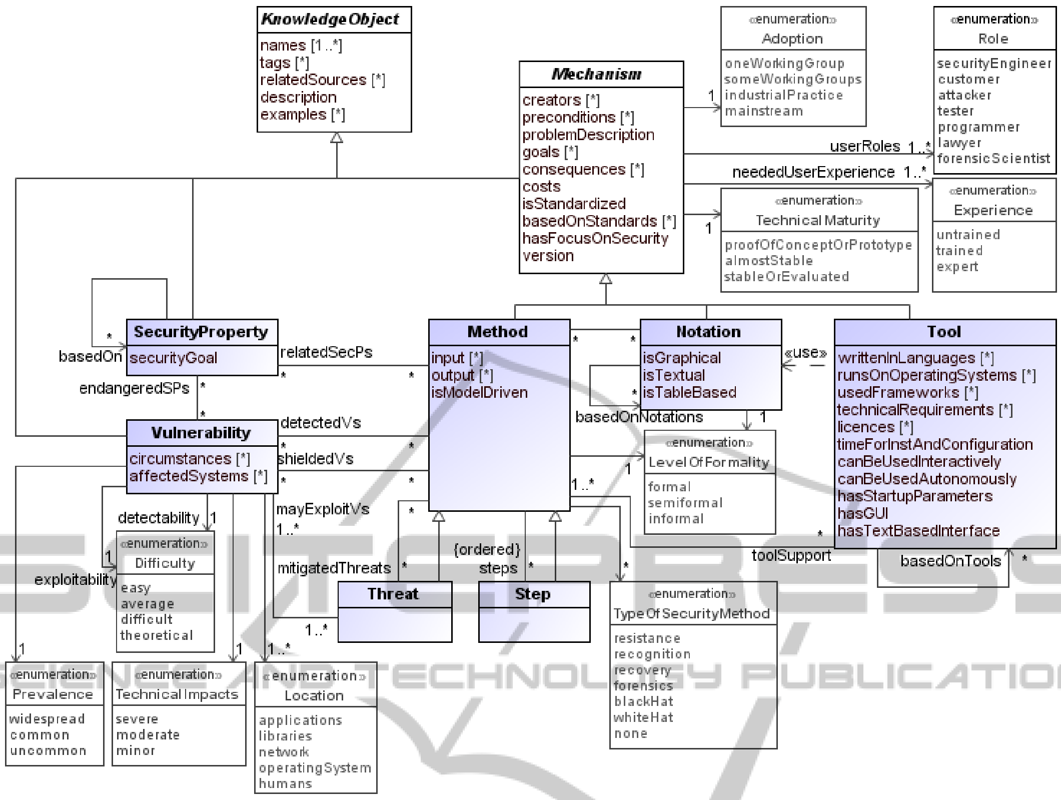

Figure 4: SECEVAL: Security Context.

to be related to a context model's class.

Figure 4 shows a more detailed data structure of the Security Context package, as attributes and enumerations are shown.

During our experience with the CBK, we noticed that tools as well as methods would be better described according to their usage in the SDLC, because attributes which are used to describe a method or tool are related to the SDLC phases it covers. As far as we know, no phase-related attributes are needed to describe features of notations. Figure 5 also depicts our `Method` class. The abstract class `MAreasOfDev` is a wildcard for detailed information about the method. A method can support several development phases. The phases of the SDLC are the same we have chosen to classify tools and methods in the NESSoS project (Busch and Koch, 2011): requirements, design, implementation, testing, assurance, risk & cost, service composition and deployment.

The full MagicDraw 17.0 model of SECEVAL and all diagrams (including the overall process and full details of method and tool description according to phases within the SDLC) can be downloaded from the web (Busch, 2013). There, the OWASP and Moody extension of SECEVAL, which was mentioned in Section 4 can also be found.

In (Busch and Koch, 2013), we use SECEVAL for a case study in which we evaluate vulnerability scanners for testing security features of web applications. Additionally, we plan to update our knowledge about secure web engineering approaches by conducting an evaluation based on SECEVAL. A long-term objective is to implement SECEVAL as a knowledge base, similarly to the CBK.

In summary, it can be stated that our approach for engineering secure web applications and our conceptual evaluation framework SECEVAL, tackle the problem of securing applications during the SDLC. Consequently, a long-term impact could be the reduction of security flaws and of necessary security patches.

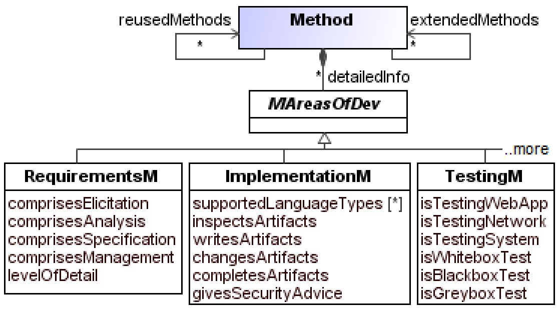

Figure 5: SECEVAL: Details of Methods (excerpt).

REFERENCES

Basin, D., Clavel, M., and Egea, M. (2010). Automatic Generation of Smart, Security-Aware GUI Models. In *Engineering Secure Software and Systems*, volume 5965 of *Lecture Notes in Computer Science*, pages 201–217. Springer.

Basin, D., Clavel, M., Egea, M., García de Dios, M. A., Dania, C., Ortiz, G., and Valdazo, J. (2011). Model-Driven Development of Security-Aware GUIs for Data-Centric Applications. In Aldini, A. and Gorrieri, R., editors, *Foundations of Security Analysis and Design VI*, volume 6858 of *Lecture Notes in Computer Science*, pages 101–124. Springer Berlin Heidelberg.

Becker, P., Papa, F., and Olsina, L. (2013). Enhancing the Conceptual Framework Capability for a Measurement and Evaluation Strategy. *4th International Workshop on Quality in Web Engineering* , 6360:1–12.

Beckers, K., Eicker, S., Heisel, M., and (UDE), W. S. (2012). NESSoS Deliverable D5.2 – Identification of Research Gaps in the Common Body of Knowledge. http://www.nessos-project.eu/media/deliverables/y2/NESSoS-D5.2.pdf.

Bertolino, A., Busch, M., Daoudagh, S., Koch, N., Lonetti, F., and Marchetti, E. (2013). A Toolchain for Designing and Testing XACML Policies. In *Proceedings of ICST 2013*.

Brambilla, M. and Fraternali, P. (2013). Large-scale Model-Driven Engineering of web user interaction: The WebML and WebRatio experience. *Science of Computer Programming*.

Busch, M. (2011). Integration of Security Aspects in Web Engineering. Master's thesis, Ludwig-Maximilians-Universität München. http://uwe.pst.ifi.lmu.de/publications/BuschDA.pdf.

Busch, M. (2013). SecEval – Information and Figures. http://www.pst.ifi.lmu.de/ busch/SecEval/.

Busch, M. and García de Díos, M. A. (2012). ActionUWE: Transformation of UWE to ActionGUI Models. Technical report, Ludwig-Maximilians-Universität München. Number of Report: 1203.

Busch, M., Knapp, A., and Koch, N. (2011). Modeling Secure Navigation in Web Information Systems. In Grabis, J. and Kirikova, M., editors, *10th International Conference on Business Perspectives in Informatics Research*, LNBIP, pages 239–253. Springer Verlag.

Busch, M. and Koch, N. (2011). NESSoS Deliverable D2.1 – First release of Method and Tool Evaluation. http://www.nessos-project.eu/media/deliverables/y1/NESSoS-D2.1.pdf.

Busch, M. and Koch, N. (2013). NESSoS Deliverable D2.4 – Second Release of the Method and Tool Evaluation. to appear.

Busch, M., Koch, N., Masi, M., Pugliese, R., and Tiezzi, F. (2012). Towards model-driven development of access control policies for web applications. In *Model-Driven Security Workshop in conjunction with MoDELS 2012*. ACM Digital Library.

Busch, M., Koch, N., and Wirsing, M. (2014). SecEval: An Evaluation Framework for Engineering Secure Systems. submitted.

Busch, M., Ochoa, M., and Schwienbacher, R. (2013). Modeling, Enforcing and Testing Secure Navigation Paths for Web Applications. Technical Report 1301, Ludwig-Maximilians-Universität München.

Elahi, G., Yu, E., and Zannone, N. (2010). A vulnerability-centric requirements engineering framework: analyzing security attacks, countermeasures, and requirements based on vulnerabilities. *Requirements Engineering*, 15(1):41–62.

Gilmore, S., Gönczy, L., Koch, N., Mayer, P., Tribastone, M., and Varró, D. (2011). Non-functional Properties in the Model-Driven Development of Service-Oriented Systems. *SOSYM*, 10(3):287–311.

Hafner, M. and Breu, R. (2008). *Security Engineering for Service-Oriented Architectures*. Springer.

IFML (2013). Interaction Flow Modeling Language (IFML), FTF – Beta 1. OMG standard. http://www.omg.org/spec/IFML/.

Jürjens, J. (2004). *Secure Systems Development with UML*. Springer. Tools and further information: http://www.umlsec.de/.

Kitchenham, B. and Charters, S. (2007). Guidelines for performing Systematic Literature Reviews in Software Engineering. Technical Report EBSE 2007-001, Keele University and Durham University Joint Report.

Koch, N., Knapp, A., Zhang, G., and Baumeister, H. (2008). UML-based Web Engineering: An Approach based on Standards. In *Web Engineering: Modelling and Implementing Web Applications*, Human-Computer Interaction Series, pages 157–191. Springer.

LMU (2013). UWE – UML-based Web Engineering Homepage. http://uwe.pst.ifi.lmu.de/.

Lodderstedt, T., Basin, D., and Doser, J. (2002). SecureUML: A UML-Based Modeling Language for Model-Driven Security. In *Proc. 5th Int. Conf. Unified Modeling Language (UML'02)*, volume 2460 of *Lecture Notes in Computer Science*, pages 426–441. Springer.

Meliá, S., Gómez, J., Pérez, S., and Díaz, O. (2008). A Model-Driven Development for GWT-Based Rich Internet Applications with OOH4RIA. In *ICWE'08*, pages 13–23. IEEE.

Menzel, M. and Meinel, C. (2009). A Security Meta-model for Service-Oriented Architectures. In *Proc. 2009 IEEE Int. Conf. Services Computing (SCC'09)*, pages 251–259. IEEE.

Moody, D. L. (2003). The Method Evaluation Model: a Theoretical Model for Validating Information Systems Design Methods. In Ciborra, C. U., Mercurio, R., de Marco, M., Martinez, M., and Carignani, A., editors, *ECIS*, pages 1327–1336.

Rzehaczek, K. (2013). Transformation of graphical UWE models to a textual DSL. Bachelor Thesis.

Slimani, N., Khambhammettu, H., Adi, K., and Logrippo, L. (2011). UACML: Unified Access Control Modeling Language. In *NTMS 2011*, pages 1–8.

Valverde, F. and Pastor, O. (2008). Applying Interaction Patterns: Towards a Model-Driven Approach for

Rich Internet Applications Development. In *Proc. 7th Int. Wsh. Web-Oriented Software Technologies (IW-WOST'08)*.

Wang, J. A. and Guo, M. (2009). Security Data Mining in an Ontology for Vulnerability Management. In *Bioinformatics, Systems Biology and Intelligent Computing, 2009. IJCBS '09. International Joint Conference on*, pages 597–603.

Wolf, K. (2012). Sicherheitsbezogene Model-to-Code Transformation für Webanwendungen (German). Bachelor Thesis.