Model-based Security Engineering for Compliance with Regulatory and Business Requirements

Jan Jürjens

Department of Computing, The Open University, GB
`http://www.jurjens.de/jan`

Abstract. We give an overview over a soundly based secure software engineering methodology and associated tool-support developed over the last few years under the name of Model-based Security Engineering (MBSE). The aim of this approach is to provide a methodology to develop secure software in a way that is compliant with regulatory and business requirements. We focus in particular on applications in industry.

Problem Statement

Compliance frameworks, laws and regulations such as Sarbanes Oxley, Basel II, Solvency II, HIPAA, REACH, RoHS/WEEE and 21 CFR Part 11 demand from companies in a more and more rigorous way to demonstrate that their organisation, processes and supporting IT landscape implement and follow a set of guidelines at differing levels of abstraction. This immediately impacts the software vendors. They find themselves in a position where they need to demonstrate that their software has been designed and implemented in a way that allows for compliant configuration and operation.

There is a strong relationship between compliance, security, and risk management. Many of the regulations have been introduced in response to fraudulent behaviour of all different kinds being detected (cf. the Antifraud Resource Center of AICPA the American Institute of Certified Public Auditors at http://antifraud.aicpa.org) or other kinds of misuse of an organisations business processes. Since business processes are run through IT systems, preventive measures and controls required by the regulations can be implemented through IT security mechanisms, including identity management, authentication, role-based access control, auditing, trust mechanisms, etc. Preventing insider and outsider attacks by these means directly contributes to compliance.

However, the contribution of the introduction of a security mechanism or other design decisions for the enterprise IT system with respect to achieving compliance has to be evaluated. In essence, this means to specify the properties achieved by the particular decision, to evaluate their potential to implement a control or to mitigate a business risk identified by a regulation, to assess alternate solutions, and to perform a cost / benefit analysis. Altogether, we need a process guided by risk considerations implied by compliance, and the management of the risks throughout the system lifecycle.

It is the aim of to work summarized in this presentation to develop a software engineering process, driven by security, risk and compliance management considerations. It aims to be equipped with tool-support that will enable seamless tracing and evaluation of security requirements, risks and safeguards along the software lifecycle. Moreover, a focus will be put on the transition between risk, security and compliance requirements and the resulting service oriented software and architectures. The aim is to identify and develop security architectures and artifacts (e.g., services, methods, solutions, and patterns) enabling the realization and enforcement of high-level security requirements.

Because of space restrictions, we can only give a very general overview over the work presented here. For details, the reader is referred to the cited publications.

Proposed Solution

Model-based Security Engineering In MBSE, recurring security requirements (such as secrecy, integrity, authenticity and others) and security assumptions on the system environment, can be specified either within a UML specification, or within the source code (Java or C) as annotations (see [Jür02,Jür04,Jür05a,Jür05b,Jür06,Jür07] for details). The associated tools [UML04] (Fig. 1b) generate logical formulas formalizing the execution semantics and the annotated security requirements. Automated theorem provers and model checkers automatically establish whether the security requirements hold. If not, a Prolog-based tool automatically generates an attack sequence violating the security requirement, which can be examined to determine and remove the weakness. This way we encapsulate knowledge on prudent security engineering as annotations in models or code and make it available to developers who may not be security experts. Since the analysis that is performed is too sophisticated to be done manually, it is also valuable to security experts.

One can use MBSE within model-based development (Fig. 1a). Here one first constructs a model of the system. Then, the implementation is derived from the model: either automatically using code generation tools as far as available on the market, or manually, in which case one can generate test sequences from the model to establish conformance of the code regarding the model. The goal is to increase the quality of the software while keeping the implementation cost and the time-to-market bounded. For security-critical systems, this approach allows one to consider security requirements from early on in the development process, within the development context, and in a seamless way through the development cycle: One can first check that the system fulfills the relevant security requirements on the design level by analyzing the model and secondly that the code is in fact secure by generating test sequences from the model. However, one can also use our analysis techniques and tools within a traditional software engineering context, or where one has to incorporate legacy systems that were not developed in a model-based way. Here, one starts out with the source code. One can use

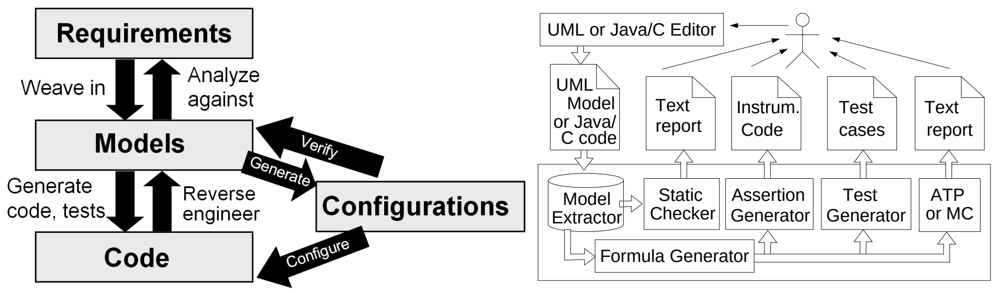

Fig. 1. a) Model-based Security Engineering; b) Model-based Security Tool Suite

our tools to construct models based on the source code, which can then again be analyzed against the security requirements. Using MBSE, one can incorporate the configuration data (such as user permissions) in the analysis, which is very important for security but often neglected (see [Jür04] for more details).

Security Design Analysis using UMLsec [Jür04] The UMLsec extension is given in form of a UML profile using the standard UML extension mechanisms. *Stereotypes* are used together with *tags* to formulate the security requirements and assumptions. *Constraints* give criteria that determine whether the requirements are met by the system design, by referring to a precise semantics of the used fragment of UML. The security-relevant information added using stereotypes includes security assumptions on the physical level of the system, security requirements related to the secure handling and communication of data, and security policies that system parts are supposed to obey. The UMLsec tool-support in Fig. 1b) can be used to check the constraints associated with UMLsec stereotypes mechanically, based on XMI output of the diagrams from the UML drawing tool in use [UML04,Jür05b]. There is also a framework for implementing verification routines for the constraints associated with the UMLsec stereotypes. Thus advanced users of the UMLsec approach can use this framework to implement verification routines for the constraints of self-defined stereotypes. The semantics for the fragment of UML used for UMLsec is defined in [Jür04] using so-called *UML Machines*, which is a kind of state machine with input/output interfaces similar to Broy's Focus model, whose behavior can be specified in a notation similar to that of Abstract State Machines (ASMs), and which is equipped with UML-type communication mechanisms. On this basis, important security requirements such as secrecy, integrity, authenticity, and secure information flow are defined. To support stepwise development, we show secrecy, integrity, authenticity, and secure information flow to be *preserved* under refinement and the composition of system components. We have also developed an approach that supports the secure development of layered security services (such as layered security protocols). UMLsec can be used to specify and implement security patterns, and is supported by dedicated secure systems development processes, in particular an Aspect-Oriented Modeling approach which separates complex security mechanisms (which implement the security aspect model) from the core functionality of the system (the primary model) in order to allow a security verification of the particularly security-critical parts, and also of the composed model.

Code Security Assurance [Jür05a,Jür06] Even if specifications exist for the implemented system, and even if these are formally analyzed, there is usually no guarantee that the implementation actually conforms to the specification. To deal with this problem, we use the following approach: After specifying the system in UMLsec and verifying the model against the given security goals as explained above, we make sure that the implementation correctly implements the specification with techniques explained below. In particular, this approach is applicable to legacy systems. In ongoing work, we are automating this approach to free one of the need to manually construct the UMLsec model.

Run-time Security Monitoring using Assertions A simple and effective alternative is to insert security checks generated from the UMLsec specification that remain in the code while in use, for example using the assertion statement that is part of the Java language. These assertions then throw security exceptions when violated at run-time. In a similar way, this can also be done for C code.

Model-based Test Generation For performance-intensive applications, it may be preferable not to leave the assertions active in the code. This can be done by making sure by extensive testing that the assertions are always satisfied. We can generate the test sequences automatically from the UMLsec specifications. More generally, this way we can ensure that the code actually conforms to the UMLsec specification. Since complete test coverage is often infeasible, our approach automatically selects those test cases that are particularly sensitive to the specified security requirements (see e.g. [JW01] for more details).

Automated Code Verification against Interface Specifications For highly nondeterministic systems such as those using cryptography, testing can only provide assurance up to a certain degree. For higher levels of trustworthiness, it may therefore be desirable to establish that the code does enforce the annotations by a formal verification of the source code against the UMLsec interface specifications. We are currently developing an approach that does this automatically and efficiently by proving locally that the security checks in the specification are actually enforced in the source code.

Automated Code Security Analysis We developed an approach to use automated theorem provers for first-order logic to directly formally verify crypto-based Java implementations based on control flow graphs that are automatically generated (and without first manually constructing an interface specification). It supports an abstract and modular security analysis by using assertions in the source code. Thus large software systems can be divided into small parts for which a formal security analysis can be performed more easily and the results composed. Currently, this approach works especially well with nicely structured code (such as created using the MBSE development process). See [Jür06] for more details.

Secure Software-Hardware Interfaces We have tailored the code security analysis approach to software close to the hardware level. More concretely, we considered the industrial Cryptographic Token Interface Standard PKCS 11 which defines how software on untrustworthy hardware can make use of tamper-proof hardware such as smart-cards to perform cryptographic operations on sensitive data. We developed an approach for automated security analysis with first-order logic theorem provers of crypto protocols making use of this standard (see [Jür05c] for more details).

Analyzing Security Configurations We have also performed research on linking the UMLsec approach with the automated analysis of security-critical configuration data. For example, our tools automatically checks SAP R/3 user permissions for security policy rules formulated as UML specifications [Jür04]. Because of its modular architecture and its standardized interfaces, the tool can be adapted to check security constraints in other kinds of application software, such as firewalls or other access control configurations.

Evidence the Solution Works

Industrial Applications of MBSE include:

Biometric Authentication For a project with an industrial partner, MBSE was chosen to support the development of a biometric authentication system at the specification level, where three significant security flaws were found [Jür05b]. We also applied it to the source-code level for a prototypical implementation constructed from the specification [Jür05a].

Common Electronic Purse Specifications MBSE was applied to a security analysis of the Common Electronic Purse Specifications (CEPS), a candidate for a globally interoperable electronic purse standard supported by organizations representing 90 % of the world's electronic purse cards (including Visa International). We found three significant security weaknesses in the purchase and load transaction protocols [Jür04], proposed improvements to the specifications and showed that these are secure [Jür04]. We also performed a security analysis of a prototypical Java Card implementation of CEPS.

Web-based Banking Application In a project with a German bank, MBSE was applied to a web-based banking application to be used by customers to fill out and sign digital order forms [Jür04]. The personal data in the forms must be kept confidential, and orders securely authenticated. The system uses a proprietary client authentication protocol layered over an SSL connection supposed to provide confidentiality and server authentication. Using the MBSE approach, the system architecture and the protocol were specified and verified with regard to the relevant security requirements.

In other applications [Jür04], MBSE was used ...

- to uncover a flaw in a variant of the Internet protocol TLS proposed at IEEE Infocom 1999, and suggest and verify a correction of the protocol.
- to perform a security verification of the Java implementation Jessie of SSL.
- to correctly employ advanced Java 2 or CORBA security concepts in a way that allows an automated security analysis of the resulting systems.
- for an analysis of the security policies of a German mobile phone operator.
- for a security analysis of the specifications for the German Electronic Health Card in development by the German Ministry of Health.
- for the security analysis of an electronic purse system developed for the Oktoberfest in Munich.
- for a security analysis of an electronic signature pad based contract signing architecture under consideration by a German insurance company.
- in a project with a German car manufacturer for the security analysis of an intranet-based web information system [BJN07].
- with a German chip manufacturer and a German reinsurance company for security risk assessment, also regarding Return on Security Investment.
- in applications specifically targeted to service-based, health telematics, and automotive systems.

Related Work

After the research on model-based security engineering using UML presented in this book started in [Jür01], and after some earlier work on role-based access control such as [FH97], there exist now several lines of research toward using UML for security systems development. They seem to differ from the one presented here that they usually aim to cover a less comprehensive set of security requirements, mostly focussing on role-based access control requirements. Also, most of them do not attempt to perform an analysis of the security requirements based on a formal semantics of a simplified fragment of UML. A model-driven security approach for inter-organizational workflow is presented in [HB05,HAB06]. [RFLG03] proposes to use aspect-oriented modeling for addressing access control concerns. Functionality that addresses a pervasive access control concern is defined in an aspect. [DRR$^+$02] uses UML for the risk assessment of an e-commerce system within the CORAS framework for model-based risk assessment. This framework is characterized by an integration of aspects from partly complementary risk assessment methods. [FMMMP02] uses UML for the design of secure databases. It proposes an extension of the use case and class models of UML using their standard extension mechanisms designing secure databases. [KPP02,BKL02] demonstrate how to deal with access control policies in UML. The specification of access control policies is integrated into UML. [LBD02] shows how UML can be used to specify access control in an application and how one can then generate access control mechanisms from the specifications. The approach is based on role-based access control and gives additional support for specifying authorization constraints.

Current Status and Next Steps

Given the current insatisfactory state of computer security in practice, MBSE seems a promising approach, since it enables developers who are not experts in security to make use of security engineering knowledge encapsulated in a widely used design notation. Since there are many highly subtle security requirements which can hardly be verified with the “naked eye”, even security experts may profit from this approach. Thus one can avoid mistakes that are difficult to find by testing alone, such as breaches of subtle security requirements, as well as the disadvantages of the “penetrate-and-patch” approach. Since preventing security flaws early in the system life-cycle can significantly reduce costs, this gives a potential for developing securer systems in a cost-efficient way. MBSE has been successfully applied in industrial projects involving German government agencies and major banks, insurance companies, smart card and car manufacturers, and other companies. The approach has been generalized to other application domains such as real-time and dependability. Experiences show that the approach is adequate for use in practice, after relatively little training. On the basis of the book [Jür04] and the associated tutorial material and tools [UML04], usage of UMLsec can in fact be rather easily taught to industrial developers.

References

[BJN07] B. Best, J. Jürjens, and B. Nuseibeh. Model-based security engineering of distributed information systems using UMLsec. In *29th International Conference on Software Engineering (ICSE 2007)*, page TBD (10 pages). ACM, 2007.

[BKL02] G. Brose, M. Koch, and K.-P. Löhr. Integrating access control design into the software development process. In *Integrated Design and Process Technology (IDPT)*, 2002.

[DRR+02] T. Dimitrakos, B. Ritchie, D. Raptis, J. Ø. Aagedal, F. den Braber, K. Stølen, and S. H. Houmb. Integrating model-based security risk management into ebusiness systems development: The CORAS approach. In J. Monteiro, P. Swatman, and L. Tavares, editors, *Second IFIP Conference on E-Commerce, E-Business, E-Government (I3E 2002)*, volume 233 of *IFIP Conference Proceedings*, pages 159–175. Kluwer, 2002.

[FH97] E. B. Fernandez and J. C. Hawkins. Determining role rights from use cases. In *Workshop on Role-Based Access Control*, pages 121–125. ACM, 1997.

[FMMMP02] E. Fernández-Medina, A. Martínez, C. Medina, and M. Piattini. UML for the design of secure databases: Integrating security levels, user roles, and constraints in the database design process. In Jürjens et al. [JCF+02], pages 93–106.

[HAB06] Michael Hafner, Muhammad Alam, and Ruth Breu. Towards a mof/qvt-based domain architecture for model driven security. In Oscar Nierstrasz, Jon Whittle, David Harel, and Gianna Reggio, editors, *MoDELS*, volume 4199 of *Lecture Notes in Computer Science*, pages 275–290. Springer, 2006.

[HB05] M. Hafner and R. Breu. Realizing model driven security for inter-organizational workflows with ws-cdl and uml 2.0. In Lionel C. Briand and Clay Williams, editors, *MoDELS*, volume 3713 of *Lecture Notes in Computer Science*, pages 39–53. Springer, 2005.

[JCF$^+$02] J. Jürjens, V. Cengarle, E. B. Fernandez, B. Rumpe, and R. Sandner, editors. *Critical Systems Development with UML (CSDUML 2002)*, TU München Technical Report TUM-I0208, 2002. UML 2002 satellite workshop proceedings.

[Jür01] J. Jürjens. Towards development of secure systems using UMLsec. In H. Hußmann, editor, *4th International Conference on Fundamental Approaches to Software Engineering (FASE)*, volume 2029 of *LNCS*, pages 187–200. Springer, 2001. Also Oxford University Computing Laboratory TR-9-00 (November 2000), http://web.comlab.ox.ac.uk/oucl/publications/tr/tr-9-00.html.

[Jür02] J. Jürjens. UMLsec: Extending UML for secure systems development. In *5th Int. Conf. on the Unified Modeling Language (UML)*, LNCS. Springer, 2002.

[Jür04] J. Jürjens. *Secure Systems Development with UML.* Springer, 2004.

[Jür05a] J. Jürjens. Code security analysis of a biometric authentication system using automated theorem provers. In *ACSAC'05.* IEEE, 2005.

[Jür05b] J. Jürjens. Sound methods and effective tools for model-based security engineering with UML. In *27th Int. Conf. on Softw. Engineering.* IEEE, 2005.

[Jür05c] J. Jürjens. Verification of low-level crypto-protocol implementations using automated theorem proving. In *3rd ACM & IEEE International Conference on Formal Methods and Models for Co-Design (MEMOCODE 2005).* IEEE, 2005.

[Jür06] J. Jürjens. Security analysis of crypto-based Java programs using automated theorem provers. In *21st IEEE/ACM Int. Conf. Autom. Softw. Eng.*, 2006.

[Jür07] J. Jürjens. *IT-Security.* Springer, 2007. In preparation.

[JW01] J. Jürjens and G. Wimmel. Formally testing fail-safety of electronic purse protocols. In *16th International Conference on Automated Software Engineering (ASE 2001)*, pages 408–411. IEEE, 2001.

[KPP02] M. Koch and F. Parisi-Presicce. Access control policy specification in UML. In Jürjens et al. [JCF$^+$02], pages 63–78.

[LBD02] T. Lodderstedt, D. Basin, and J. Doser. SecureUML: A UML-based modeling language for model-driven security. In J.-M. Jézéquel, H. Hußmann, and S. Cook, editors, *5th International Conference on the Unified Modeling Language (UML 2002)*, volume 2460 of *LNCS*, pages 426–441. Springer, 2002.

[RFLG03] I. Ray, R. B. France, N. Li, and G. Georg. An aspect-based approach to modeling access control concerns. *Information & Software Technology*, 2003. To be published.

[UML04] UMLsec group. Security analysis tool, 2004. http://www.umlsec.org.