SCIENTIFIC PAPERS, UNIVERSITY OF LATVIA, 2010. Vol. 757
COMPUTER SCIENCE AND INFORMATION TECHNOLOGIES 43–52 P.

Model-driven secure system development framework

Viesturs Kaugers, Uldis Sukovskis
Riga Technical University, Faculty of Computer Science and Information Technology, Riga, Latvia
{viesturs.kaugers, uldis.sukovskis}@rtu.lv

Abstract. Security is definitely one of the most important aspects in business information systems. This aspect is strongly related to costs, risks and reputation of organization. That's why authors want to propose "secure-by-design" principle by using innovative and proven development approaches in whole system lifecycle to maintain high security level. There are already some existing techniques to solve this problem but they are mainly linked with specific technologies and most frequently focus only on production phase. This paper presents new secure system development methodology based on three general aspects – model-driven secure code development, model-driven policy development and usage of run-time security management system to maintain necessary security level. All these aspects are integrated into one framework.

Keywords:, security, MDA, model, development, framework, UML.

1 Introduction

Model is a representation of system or real-world entity. Metamodel is model for models – it describes model notation. Model-driven Architecture (MDA) is a software development method which uses models to separate the specification of the information system from the details of the implementation in specific platform. Unified Modeling Language (UML) is the basis of model description. MDA was developed by Object Management Group (OMG) in 2001 [1]. OMG is international, open membership, non-profit computer industry consortium since 1989 [2]. This organization has defined four model types – Computation Independent Model (CIM), Platform Independent Model (PIM), Platform Specific Model (PSM) described by a Platform Model (PM), and an Implementation Specific Model (ISM) [3].

MDA tends to be solution for successful overall system development methodology. It gives possibility to introduce much more formal approach in early phases of design and development helping to avoid problems in later phases. This could be done because of detailed system description in high abstraction models. Practice shows that the main source of problems in system development project and later usage is caused by bad design. Time spent on system design is balanced with automatic code generation possibilities. Although currently there is no complete and universal code generation tool situation may change in a few years and developers already can use half-assisted code

generation tools. The code generation process is based on transformation rules described in Query/View/Transformation (QVT) standard. There are also other standards defined for various activities like XML Metadata Interchange (XMI) for metadata exchange and Meta Object Facility (MOF) for model-driven engineering [4].

2 Analysis and design of security aspects

Security nowadays is a critical factor in software systems. Problems with information system security affect both developers and customers [5].

Secure system maintains four main qualities – integrity, confidentiality, authentication and availability. If system does not have the qualities mentioned above, it becomes under risks like identity theft, data alteration, unauthorized access, service interruption and others.

There is a clear need in the IT market for secure software development methods to avoid problems caused by bad information system design, faulty security policies, inadequate implementation and human factor influence. It is of high importance to understand potential risks and view system in context of all organization structure, needs and resources. The most common faults include use of weak cryptography, inadequate user management, unreasoned authorization algorithm, inefficient authentication mechanisms and improper or incomplete data validation [6].

Model-driven system design approach increases system maintainability and security level and it will have significant impact on IT systems in future [7]. This is especially important for systems dealing with money, transactions, human lives and secret information. On the other hand office applications and smaller software can be successfully developed without MDA approach. So there is a domain where system development with MDA can give the best results. MDA has several advantages important also for business environment like fast adaptation to changes (changes in model not code), architecture portability among platforms (transformation from PIM to PSM, rules) and easier to trace errors and fix them.

The main goal of model-driven security is to integrate security features and requirements directly into system architecture, make it secure-by-design. So it is easier to link system design with enterprise security policy. It is also much easier to find and fix bugs in high abstraction level rather than in code. Practice demonstrates that part of bugs in code happens due to incomplete system design [6].

Models are more understandable than code and give an overall view of system being developed. Also high level models are easier to explain to top management staff rather than specific technical data and code.

Below there are described two languages for defining, designing and modeling security mechanisms. Both languages are delivered from UML and are used at PIM level.

2.1 SecureUML

One of secure development approaches is to use SecureUML. This is a modeling language or modeling extension based on UML and role-based access control (RBAC) model. So currently SecureUML is mainly focused at access control. RBAC is more

progressive authorization model than classical access-control lists because of possibility to define what kind of operation or activity user can perform on object [8]. For example, with RBAC administrator can define not only access rights to database but also what kind of data user is allowed to change in specific database table.

SecureUML offers to use some concepts like authorization constraints and state conditions to design more dynamic authorization process. By using SecureUML it is also possible to define security policies. SecureUML metamodel extends UML metamodel with types *User*, *Role*, *Permission* and *ResourceSet* and others [6]. Every UML model element can be protected resource and belong to set of resources for defining permissions and constraints. *Permission* is an object which connects a role to a *ModelElement* or to a *ResourceSet*. Element *ActionType* describes permission or allowed actions for protected resource. For example, developer can assign attribute *MaxLength: 255* to action *SetPassword* for user *John Doe*. Element *ResourceType* defines available all available action types for specific metamodel type. Element *BaseClass* defines connection to metamodel type. Element *AuthorizationConstraint* defines precondition for action to be executed on resource.

Simple example of UML/SecureUML model set is given below in Figure 1. This is basic model set for database access and management with defined users and roles.

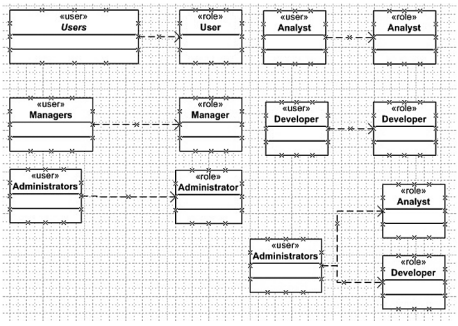

Fig. 1. Database users and roles

As it was mentioned earlier, SecureUML deals only with access control. This language would be more useful if it also could support integrity and data validation. So the authors offer to improve SecureUML with attributes and entities to support integrity, confidentiality and availability.

Integrity support can be achieved with *checksum* as attribute in *Permission* type – Figure 2.

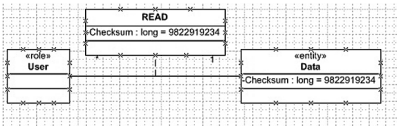

Fig. 2. Integrity support

Confidentiality requires data encryption mechanisms so we offer to add two new types *EncryptionMechanism* and D*ecryptionMechanism* which encrypts and decrypts data flow between *Users* – Figure 3.

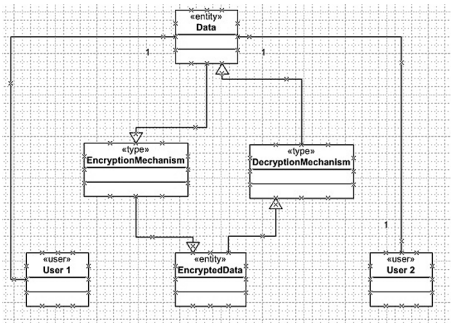

Fig. 3. Confidentiality support

Availability is a bit different characteristic and requires service to be available all time. To guarantee availability system has to be redundant, with alternative resources to perform necessary operation. In the case when main system is unavailable there should be link to alternative system as visible in Figure 4.

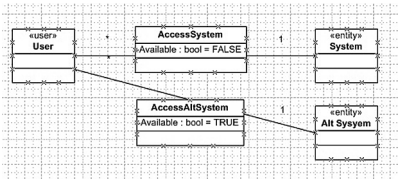

Fig. 4. Availability support

SecureUML notation is not quite suitable and needs to be enhanced for alternative link representation so we used two permissions with opposite attribute values to determine available system. On the other hand SecureUML adds additional layer of abstraction where developers can work directly on security features separated from business process diagrams. If this is not necessary they can choose UMLsec which naturally will fit into whole UML system model.

2.2 UMLsec

UML standard can be extended by using *profile*s, which consists of constraints, tagged values and stereotypes. Stereotypes allow defining new modeling elements. Tagged values are a pair of name-value. Constrains gives possibility to add additional information to model. Stereotypes attach tagged values and constraints to model elements so they define security requirements and assumptions in the system.

In UMLsec there are following stereotypes (Name (Base class, [<Tags>])) – *Internet* (Link), *Encrypted* (Link), *LAN*(Link), *secure links*(subsystem), *Secrecy*(dependency), *Integrity*(dependency), *High*(dependency), *secure dependency*(subsystem), *Critica*

(Object, subsystem, <secrecy, integrity, high>), *no down-flow*(Subsystem, <high>), *data security*(Subsystem), *fair exchange*(Subsystem, <start, stop>) and *provable*(Subsystem, <action, cert>). For example, stereotype *Internet* can be attached to the links between components so it describes how component can interoperate with it. So using mentioned stereotypes developer can design security mechanisms right inside UML model [9]. Deeper discussion of UMLsec is not the goal of this paper so we will give basic example of UMLsec usage in Figure 5.

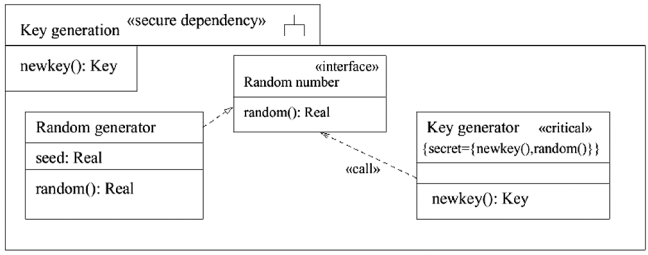

Fig. 5. UMLsec example [10]

3 Model-driven policy modeling

MDA approach is a useful method also for security policy development. It helps to create more consistent and manageable security mechanisms. By using qualification criteria which are common to all systems, designer can validate security model before and after usage and eliminate potential flaws. So security policy development with MDA consists of two phases – design and qualification phase.

In the first phase security policy is transformed into security components. One of the components is Policy Decision Point (PDP) – it is logical entity in the system that performs admission and decision control in response for access to specific resource [11]. Calls or calling points to a PDP for accessing resource are policy enforcement points (PEP). PDP and PEP graphical representation is given in Figure 6.

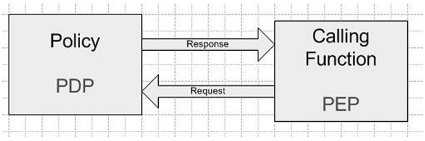

Fig. 6. PDP and PEP

Security policy is built using domain-specific language (DSL). DSL consists of all objects needed to represent policy and helps to automate the process because it provides necessary level of formalization. It is possible to generate automatically security framework, executable PDP and integrate PEP into business logic.

It is very important to ensure that PEP cannot be bypassed so there is a need for qualification phase. Qualification consists of initial security model verification before

component generation and concluding validation. Qualification environment is built on policy metamodel.

Security policy development process starts with requirement analysis then modeling, qualifications and execution come. All the process graphically is summarized in Figure 7.

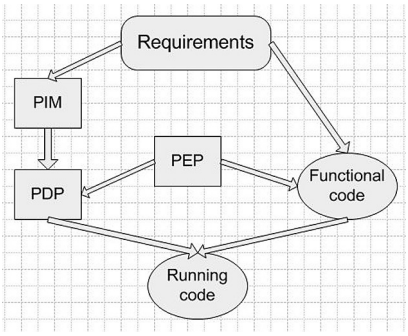

Fig. 7. Security policy design with MDA

Policy can deal with one or more security aspects like integrity, confidentiality, authentication and availability. For example, security policy can deal entirely with user access control. It means that policy model will include information about users, roles and actions. For large enterprise systems number of users can reach several thousands, every user can have some roles and many actions associated with it. So, for example, manual search for conflicting rules (one rule allows particular action, other denies) is very time-consuming job. Also at least one action needs to be associated with specific role. With MDA a developer can do these verifications automatically on platform-independent model. After PIM transformation necessary PDP are generated. PDP contains security policy implementations. These points receive requests from PEP which serves as interface between PDP and application and helps to integrate security policy software systems.

Using aspect-oriented programming business logic and security policy can be separated and security code effectively integrated into overall code of application. After integration phase comes final application testing with automated test cases [12]. Main testing goal is to ensure that security policy implementation corresponds to modeled security policy.

Security policies can be modeled using various metamodels for particular purposes. One of the metamodels is already previous mentioned RBAC. It defines all necessary concepts for access control security policy. Security policy development modeling phase is rather similar to that previously described for database access management. More detailed overview of MDA security policy development can be found in [9]. Security policies are part of model-driven secure system development framework. This is because whole system needs part where business needs and requirements are dynamically implemented and changed according to current situation.

4 Run-time security management

When software system is ready and working, it functions in dynamic and often in complex environment, linked with other systems and used by thousands of users. Software systems can be changed using patches or major upgrades. System analysts and software architects cannot predict and implement all security, usage and change scenarios at design phase, even using model-driven system development approach. On the other hand software users are interested in continuous security and reliability. There comes the challenge to maintain security for software system during all its life cycle. Run-time security management policies can be developed also in the way described in previous chapter. Such policies are understandable also for business executives and managers.

Run-time security management implementation requires defining specific security concept and requirement representation, also known as ontology, ongoing system monitoring and adequate adjustment to changes in external environment [13].

Construction of run-time security management includes finding and modeling alternative security solutions based on ontology, service discovery and adaptation mechanism. Result of construction is architecture for implementation with alternative security solutions for different situations and measurement system. Overview of run-time security management architecture is given in Figure 8.

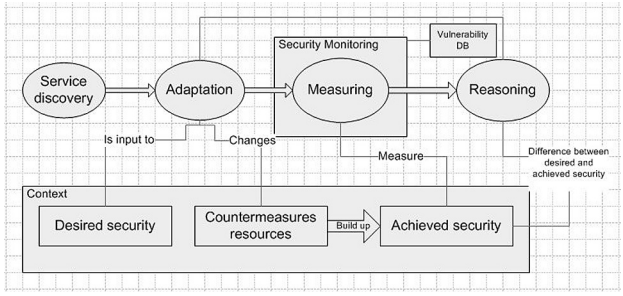

Fig. 8. Run-time security management architecture

In the beginning available services are discovered and adaptation performed on them according to their state and specifics. In adaptation process run-time security management system finds most suitable solution for security threat. After adaptation starts measurement process of base measures and system balancing according to obtained measurements. Necessary security level can change dynamically in response to situation. The whole process usually takes place without user interaction; exception is cases when system cannot maintain necessary security level [14]. In such situation it inquires assistance of the user. Run-time security management adds one more security layer for the system because it can respond to changing outer environment and automatically take preventive actions and generate easy manageable security rules. This system is a part of the model-driven secure system development framework where technical rules are enforced.

5 Model-driven secure system development framework

Model-driven secure system development framework addresses security issues and gives possibility to integrate security features in the whole system while offering innovative view on secure system development. Based on reviewed secure software development principles we are offering new approach that links all mentioned methodologies together. Overall view of the framework in high abstraction is represented in Figure 9.

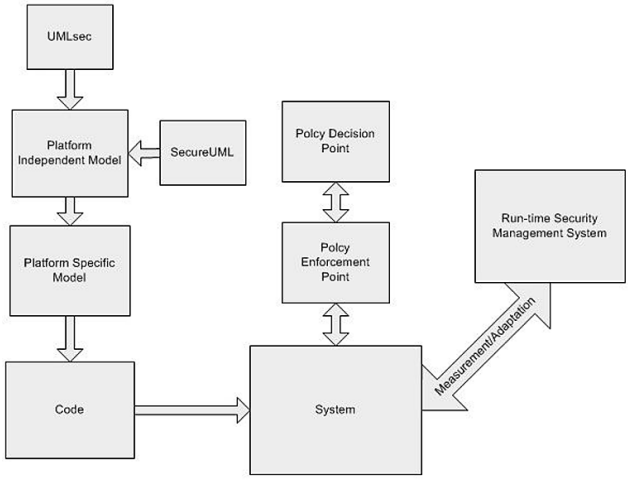

Fig. 9. Secure system development framework

This framework consists of 8 building blocks and system itself. The main goal of the framework is to integrate described security mechanisms in systematic way to provide appropriate security level in design and runtime. PIM is designed in conjunction with SecureUML or UMLsec to produce PSM and system code. System is linked with external security policy via PEP to maintain business rule changes in future. Run-time security management system protects main system from hazardous technical changes in operating environment. This approach ensures that security all considerations are taken into account and development process is model-driven.

PSM is generated from PIM and this is more technically detailed version of system architecture. Developers can mike minor adjustments here although it is recommended to work on PIM. Next level is code from which executable system is produced. In model-driven secure system development code shouldn't be changed manually/directly except if there is strong need for that. In such cases all changes must be reflected in higher level models.

System itself can be described using general system theory. This theory helps to understand working mechanisms of system without going in too much detail and is theoretical basis for the framework. It may be usable when system architects and developers communicate with business people to coordinate design process and gather necessary aspects. Basically it consists of various components and links between them (Figure 10).

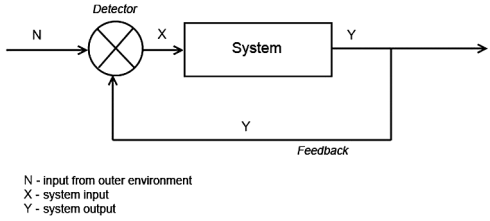

Fig. 10. Scheme of general system

In context of this framework system is controllable; it has inputs and outputs, detector and selector. The role of detector is performed by run-time security management system. Selector is implemented by policy decision point. Inputs may be any kind of business information, policy rules and control signals from run-time security management system. Outputs may be processed inputs. General system can be divided into smaller subsystems but they all apply to the same rules. In context of framework inputs are model and output is executable system.

One of the main advantages of secure system development framework is that external security policy and run-time security management system can be added and changed later during system usage. This is because framework prescribes connection points in design with cooperating systems. When main system is done, developers can continue their work on creation of mentioned systems. Of course development costs are higher in comparison with standard systems but added security requires additional expenses like in other secure system development methodologies.

The power of this framework lies in its flexibility. It is easier to integrate changing business requirements and technical measures if it is predicted in system during its development phase. Development of system can be divided into parts and every part is not limited to specific technology or approach. It is even possible, for example, to develop run-time security management system using rapid prototyping. Also PIM of the system can be developed in conjunction with security policy. Business people can easier deal with security policies and be involved in their development, because many technical things are hidden at this level. System administrators can benefit from run-time security management application which makes their job easier.

6 Results, conclusions and further research

In this paper we examined concepts of MDA applicable to system security, run-time security management, SecureUML and UMLsec languages and policy modeling. We described enhancement for SecureUML modeling language to extend its support for integrity, confidentiality and availability. Based on reviewed secure software development methods and principles we offered a new development framework that links all mentioned methodologies together. This framework aims to provide complete and integrated secure system development methodology

Future work will focus on deeper analysis and description of secure system development framework, making it detailed. Aspects of policy and run-time security

management system development require more research. Framework needs to be verified in real-life development projects. At some point it may be even possible to split system development between different developers to speed up project movement. This also needs to be researched in upcoming work.

References

1. Frank Truyen. The Fast Guide to Model Driven Architecture. Cephas Consulting Corp, 2006, 16 p.
2. About the Object Management Group. Prepared by OMG.: Object Management Group – http://www.omg.org/gettingstarted/gettingstartedindex.htm. – visited at February 2010.
3. An introduction to Model Driven Architecture. Prepared by Alan Brown.: IBM – http://www.ibm.com/developerworks/rational/library/3100.html. – visited at February 2010.
4. Joaquin Miller, Jishnu Mukerji. MDA Guide Version 1.0.1. OMG, 2003, 62 p.
5. The Importance of Information Security for Financial Institutions and Proposed Countermeasures. Prepared by Bank of Japan.: Bank of Japan – http://www.boj.or.jp/en/type/release/zuiji/kako02/fsk0004b.htm. – visited at February 2010.
6. Rudolph Araujo, Shanit Gupta. Design Authorization Systems Using SecureUML. – Foundstone, 2005, 16 p.
7. Model-Driven Security: Enabling a Real-Time, Adaptive Security Infrastructure. Prepared by MacDonald N.: Gartner – http://www.gartner.com/DisplayDocument?id=525109. – visited at February 2010.
8. David F. Ferraiolo, D. Richard Kuhn. Role-Based Access Controls. – Baltimore MD: National Institute of Standards and Technology, 1992, 11 p.
9. P. Shabalin, J. Jürjens. Towards Tool Support for UMLsec (Poster Proposal). – Dep. of Informatics, Munich University of Technology, Germany, 2003, pp. 1-3.
10. Jan Jurjens. UMLsec: Extending UML for Secure Systems Development. – Dep. of Informatics, Munich University of Technology, Germany, 2002, pp. 1-9.
11. PDP – Policy Decision Point. Prepared by Christine Martz.: Birds-Eye – http://www.birds-eye.net/definition/p/pdp-policy_decision_point.shtml. – visited at February 2010.
12. Tejeddine Mouelhi1. et. al. A model-based framework for security policies specification, deployment and testing. – Berlin: Springer, 2008, 15 p.
13. Antti Evesti, Eila Ovaska, Reijo Savola. From Security Modelling to Run-time Security Monitoring. – Finland: VTT Technical Research Centre of Finland, 2009, 11 p.
14. Ulrich Lang, Rudolf Schreiner. Managing business compliance using model-driven security management. - Vieweg&Teubner, 2009, 241 p.
15. General Systems Theory. – http://www.istheory.yorku.ca/generalsystemstheory.htm. – visited at February 2010.