Towards a Systematic Development of Secure Systems

Ruth Breu[1], Klaus Burger[1], Michael Hafner[1], Gerhard Popp[2]

[1] Research Group "Quality Engineering"
Universität Innsbruck, Institut für Informatik
Technikerstraße 13
A - 6020 Innsbruck

[2] Software & Systems Engineering
Technische Universität München , Institut für Informatik
Bolzmannstraße 3
D-85748 Garching

Abstract. In this paper we outline a new process model for security engineering. This process model extends object oriented, use case driven software development by the systematic treatment of security related issues. We introduce the notion of security aspects describing security relevant requirements and measures at a certain level of abstraction. We define a micro-process for security analysis supporting the systematic development of secure components within iterative systems development.

1 Introduction

Due to the increasing number of distributed applications security plays a more and more important role within systems development. In particular, evolving new web technologies supporting the dynamic interconnection between software components and novel mobile devices require a high level of security.

Today's process models like the Unified Process ([1, 2]) or Catalysis ([3]) treat security aspects as non-functional requirements among others. Our claim is that security is a requirement which has to be considered in all stages of development and which needs particular modelling techniques to be captured.

Moreover, the development of secure systems poses particular challenges to the development process. This comprises the separation of requirements and measures, the traceability of security requirements, the correctness of the measures taken and the completeness of requirements and measures.

Observing that security relevant issues are often merely considered at the technical level (by using encryption techniques, security protocols, logging etc.) our main goal is to separate abstraction levels and to specify requirements and measures at the appropriate level. This ranges from eliciting security requirements in the business model, taking into account security specific aspects at the level of work processes to the technical level of the software architecture. A first step towards this aim has been

Breu R., Burger K., Hafner M. and Popp G. (2004).
Towards a Systematic Development of Secure Systems.
In *Proceedings of the 2nd International Workshop on Security in Information Systems*, pages 1-12
DOI: 10.5220/0002654300010012

achieved in [4, 5, 6, 7].

We describe both security requirements and measures in the artefact of the core process and call these requirements and measures *security aspects*. Our core process is based on a general approach which can be easily mapped to any of the established process models [8]. The core artefacts are the Business Model (describing work processes), the System Requirements (describing the system's use cases), the Application Architecture (describing the system's logical components and the core message flows) and the Software Architecture (describing the technical structure).

A special challenge to the development process of security critical systems is imposed by the concept of iterative software construction. For instance, the introduction of classes in a new increment requires an elaboration of the access rights which in turn may lead to new measures and use cases (e.g. logging the access to the new objects and surveying this logging information).

We meet this challenge by introducing a micro-process for security analysis. The micro-process comprises the five steps of security requirements elicitation, threats and risk analysis, taking measures and the correctness check relating measures and requirements. These five steps are repeatedly performed at each level of abstraction during the incremental development.

The two steps of threats and risk analysis support the transition from requirements to measures by gathering the potential threats related to the security requirements and by estimating the occurrence of each threat and its potential harm. Separating the application and the technical level we define two new artefacts - the Application Risks and the Technical Risks containing the description of threats and risks at the respective level of abstraction.

The structuring of this paper is as follows. In section 2 we outline the principles of the design process our approach is based on. Section 3 presents the activities and artefacts of our process model and in section 4 a conclusion is drawn.

We illustrate this process model with a case study based on "TimeTool", a software project which was realized at the University of Innsbruck. TimeTool is a software package supporting project controlling and administration. Based on a three tier architecture it was implemented on top of the J2EE platform. The application is accessed through a web front-end. Team workers' working time is calculated in real time through a log in / log out timestamp, with the option of performing the entries manually under special circumstances. The system performs specific checks (on date and booked time) automatically and offers administrative and controlling features to the project manager (team worker management, statistical reports generation).

2 Basic Concepts of an Object Oriented Software Process

In this section we present the key concepts of the core object oriented process.

The **Business Model** captures the organizational environment of the IT-system. It describes the *actors*, the *activities* and the *objects*. In TimeTool the actors are the *project manager*, the *team worker* and the *administrator*. Example activities are *Book Worked Hours* and *Post Adjustment*. Example objects in the application domain are the *project*, the *booking* and the *team worker*.

Actors, activities and objects are modelled in activity diagrams and class diagrams. In its system view the business model focuses on the work processes and is independent of the IT-system.

The **System Requirements** consist of the use case diagram and the class model and give a black box view of the system. A sample use case is *Book Worked Hours*.

Commonly, the class model of the System Requirements is a refined version of the class model of the Business Model. The textual description of a use case comprises sections for pre- and post-conditions of the use case, for the main steps and interactions when performing the use case and sections for exceptions and variants.

The **Application Architecture** refines the level of description. The system is divided into a set of *logical components*. Each component is responsible for a portion of the system structure and behaviour. It consists of component diagrams, a set of sequence diagrams and state diagrams. Interfaces enable the independent development of the system components. Textual descriptions of the use cases are refined into *scenarios*, describing the use cases as message flows between objects.

Besides the artefacts themselves, their sequence and interdependencies form the main characteristics of a process. In this respect, iterative development [2] is one of the most important concepts in modern process models. For instance, for the System Requirements this means, that not all use cases are specified in detail in a first step but only the kernel ones, and other use cases are specified in later stages of design.

3 Core Concepts of a Process Model for Security Engineering

In this section we present the concepts of our process model. The core idea is the introduction of a micro-process which we call *Security Analysis*. In section 3.1 we clarify the basic idea of the micro-process and its integration in the core process. Section 3.2 is devoted to the security enhanced artefact.

3.1 The Security Analysis Process

Security related aspects in the software lifecycle are tackled in a five step approach which we call *Security Analysis* (Table 1).

We illustrate these five steps by the scenario in Table 2. In our process model we treat the Security Analysis as a micro-process which is *performed* at *each level of abstraction* and *for each increment*. This has the following advantages:

- Requirements and measures are each explored and described at the appropriate level of detail. Each security requirement can be traced along the levels of abstraction. More precisely, each requirement is transformed into one or several requirements or into some measure at the abstraction level beneath.

Table 1. The Security Analysis Process

1. *Security Requirements Elicitation* – Specify security requirements in the context of the core artefact. 2. *Threats Modelling* – Gather potential threats related with the security requirements. 3. *Risk Analysis* – Estimate the occurrence of every threat and its potential harm either quantitatively or qualitatively. This provides the basis for the decision whether a threat has to be countered or not. 4. *Measures Design* – Design appropriate measures taking into account the result of the risk analysis and integrate the description of these measures into the core artefacts. 5. *Correctness Check* – Check the chosen measures (formally or informally) against the specified requirements and decide what requirements still wait for realisation.

Table 2. Scenario of a Security Analysis

1. The non-repudiation of the activity *Adjustment Posting* (performed either by the project manager or the team worker) is identified as a security requirement in the business model of "TimeTool". 2. This activity is associated with the following threats: -The team worker performs a positive time adjustment on her account in order to increase her billable time. - The project manager performs negative time adjustments on several accounts in order to hide budget overruns. 3. The probability of occurrence is estimated as high, the possible damage is estimated as substantial. 4. The measures to counter the threats involve both the business level and the IT System. On the one hand side the business process is reorganised and improved, e.g. adjustment postings require additional information and involved parties are automatically notified. On the other hand a functional requirement, namely that all *Adjustment Postings* have to be logged by the system is added to the System Requirements. 5. The proposed measures are checked against the requirement of non-repudiation. The result of this new requirement is that the involved parties must not have access to logging information.

Security Aspects. Security measures at one level of abstraction may be seen as security requirements at a lower level of abstraction. This is why we generalise requirements and measures to the concept of *security aspects*.

Security aspects are security relevant parts described in the core artefact. For expressing some of the security aspects we introduce extended notation techniques, e.g. in the Business Model. Other security aspects can be described within the UML notation (e.g. sequence or state diagrams can be used to describe security protocols).

A core idea in our approach is that we relate security aspects by a *realisation relation*. Each security aspect is realised by zero, one or many other security aspects. For instance, the security aspect of non-repudiation in the Business Model of the sample scenario of Table 2 is realised by a business process enhancement and a logging mechanism. Aspects realised by no other aspect are measures at the lowest level of abstraction. In general, the realisation relation may relate aspects at different levels of abstraction (in different core artefacts) or at the same level of abstraction. The relation is many-to-many. Our approach has the advantage that security requirements and measures can be traced through several artefact. This supports a systematic check for correctness and completeness.

Artefacts and their Integration into the Process. The enhanced process involves the following artefacts:

- All core artefacts are also part of the enhanced process. We extend these core artefacts by techniques and methods to express security requirements at the given level of abstraction.
- We define two additional artefacts – the *Application Level Risks* and the *Technical Risks* – documenting threats and risks at the application and platform-dependent level.

Table 3. Iterations of a Given Core Artefact

Step 0:	Develop the core artefact as described in the core process.
Step 1:	Enhance this artefact by security aspects modelling requirements according to given methodological guidelines.
Step 2, 3:	Analyse related threats and risks in the respective Risks artefact.
Step 4:	Integrate security aspects in the core artefact modelling measures. These security measures may refer to security aspects of the abstraction level above or of the same abstraction level. Document the realisation relation between the new aspect and the given aspects.
Step 5:	Perform the correctness check and eventually add refined security aspects to be fulfilled by the abstraction level beneath.

The Risks artefacts complement the specification of requirements by supporting the choice of appropriate measures and the definition of test cases. In the Risks artefacts both external and internal risks are analysed. From the viewpoint of a given core artefact the Security Analysis results in the series of steps (for each increment) depicted in Table 3.

A crucial aspect in this method is the comprehensive specification of security requirements. In our approach we provide systematic checks of the base models with respect to the following settled set of objectives:

- Confidentiality – keeping content from all but those authorised to have it
- Authenticity – establishing the validity of transmission, message or originator
- Data Integrity – prevention of unauthorised modification of data

- Non-Repudiation - guarantee that an entity cannot deny previous commitments of actions
- Availability – ensuring that unauthorised subjects cannot prevent authorised ones from the execution of their functions

Figure 1 summarises the activities and artefact of the enhanced process. In the subsequent section we will demonstrate the application of our process in more detail focusing on the security enhancements of the core artefact.

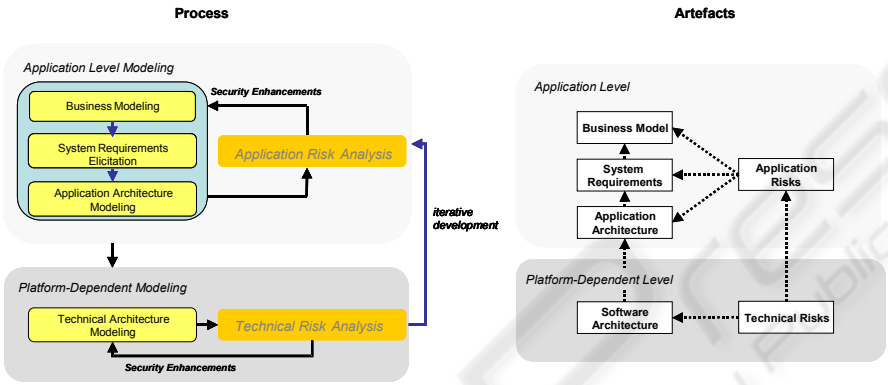

Figure 1. The Security Process Model

3.2 Security Enhanced Core Artefacts

In the following we briefly describe for each artefact how security requirements are specified, what kind of threats are captured in the correlated Risks document and what kind of measures the core artefact may comprise. In our approach we follow a schematic pattern-based elicitation of security requirements. This may be complemented by textual or formal specifications. For a more detailed presentation we refer to related publications ([9, 10]).

Business Model. The systematic security check of the given business model comprises the following aspects.

Confidentiality is specified at the business level at a rough level of detail in the class diagram. Each class (or attribute) is provided with one of the keywords public, confidential or secret. Another aspect is related with the object flow between activities. Object flows between different actors require some data exchange. For every single object flow in the business process model we have to check if confidentiality of the object involved is critical. Data Integrity concerns the access to objects and is captured by the security levels in the class diagram described above. Moreover, analogously to above, each object flow in the process model is checked for the aspect of data integrity.

As an example, think of the workflow of an adjustment posting (Figure 2). Since the adjustment posting and its confirmation as well as the generated notification contain the data about billable time both, their confidentiality and integrity is critical.

Authenticity is a requirement which refers to activities and actors in the business model. Each activity has to be checked if authenticity of the executing actor is a critical requirement. In our example the project manager has to be authenticated when executing the activities *confirm adjustment* and, like any team worker, *perform booking* (of his billable time spent on project).

Non-repudiation is again a requirement involving activities and actors. For each activity it has to be checked if it is important that the executing actor cannot repudiate the execution of this activity. In our case study the activity *Post Adjustment* is associated with the requirement of non-repudiation.

Each of the security requirements related threats, like the one described in Figure 2 (item 2), has to be described in the Application Level Risks document and estimated. Measures at the level of the business model may involve the reorganisation of work processes typically including the separation of duties and the way of handling objects (e.g. requiring the destruction of certain documents after use).

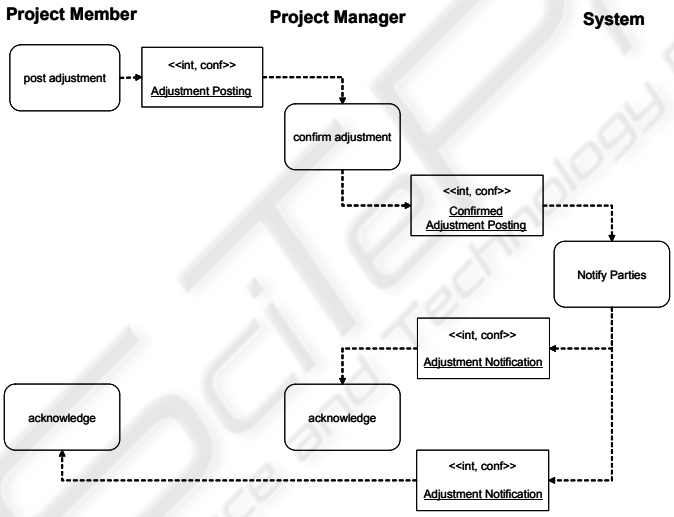

Figure 2. Sample Object Flow with Security Requirements

System Requirements. In the System Requirements document the security requirements of the business model are systematically detailed and put into the context of the use cases and the extended class diagram.

An important activity within the specification of system requirements is the development of a detailed *access policy*. In [11] we present a formally based model for specifying access rights in the context of class diagrams and use case diagrams. Using an extension of OCL predicates (or informal text) the model describes for each

actor and each class (or, on a more detailed level, for each method or attribute) the kind of permission. As examples we specify:

The team worker has read access to his own accounted working hours.
The team worker has write access on his own accounted working hours whose status I not set to "frozen".
The project manager has read access to the accounted hours of all projects members of his project.

Here reading and writing access means an indirect access through application of the use cases. The access policy is developed in cooperation with customers and/or end-users. If the access rights are specified formally they can be automatically transformed into code. For a more detailed presentation of our specification framework we refer to [11].

Other security relevant aspects are part of the textual description of use cases or are treated in new use cases (such as the use case *log in* in which the authentication takes place). The schematic textual description of use cases is now extended by a section **security,** describing the enhanced security aspects.

As an example, the textual description of the use case *Adjustment Posting* is enhanced by the security aspects A1 and A2 as shown in Table 4. Both requirements are refined versions of the security requirements in the business model.

Table 4. Security Section of the Use Case Adjustment Posting

use case Adjustment Posting
… (previous textual description of the use case)
security
A1 The adjustment posting is logged by the system.
A2 The team worker has to be authenticated before starting the use case.
A3 Web browser and TimeTool have to authenticate each other before the transaction starts.
A4 The system must guarantee the confidentiality and integrity of the input data.
A5 The use case must be available during extended working hours (6a.m. to 22a.m.) with a maximum of 2 continuous working days breakdown per month.

Further security requirements contained in the use case description analyse the communication with external systems. In our example the use case *Adjustment Posting* involves the communication with the web browser of the client. Since confidential data is sent across the network, requirements A3 and A4 were added to the security section on the present use case.

Finally, another requirement which comes into play at this level of abstraction is availability (A5) which guarantees a minimum availability of the system during working hours and days.

The security analysis of the use cases in general leads to new threats (like the threat that a team member tries to manipulate the logging information). These threats have to be integrated in the Application Level Risks document and should be linked with the relevant parts in the System Requirements.

Application Architecture. Based on the security requirements stated in the System Requirements, the security enhancement of the Application Architecture mainly deals with the design of appropriate measures. This comprises the following steps:

- Definition of logical security components and their interfaces
- Extension of the sequence diagrams describing the message flow of use case execution by security specific messages

Concerning the security requirements of the use case model the development of the (Security) Application Architecture involves the following kind of measures:

- Design of authentication procedures
- Access control and access rights management at the application level
- Error tracing measures (e.g. for authentication and access control)
- Introduction of security protocols for data integrity, confidentiality and non-repudiation

As an example, the use of a challenge-response protocol for authentication or of blind signatures for data integrity is chosen at this level of abstraction. The check of correctness of these measures against the requirements may require complex mathematical proofs.

For this reason and for the systematic transition from requirements to measures the use of *security patterns* is of great importance in this stage of design. Catalogs and classifications of security patterns currently are developed by a number of groups [12, 13, 14, 15]. However, the flexible integration of security patterns into concrete models still poses a number of unsolved problems, e.g. concerning the dynamic behaviour of the resulting system and the combination and interference of patterns.

Threats recovered at the level of the Application Architecture relate to the chosen measures. An example for this are the threats created when applying the still very popular perimeter security model of the “mainframe era” to a distributed server environment [16].

Software Architecture. This phase starts with the design of the basic architecture (comprising the network structure, database structure, choice of programming languages, frameworks and so on) and the logical components which are distributed across network nodes.

The security enhancement to the Software Architecture (Security Software Architecture for short) is then developed in five consecutive steps as listed in Table 2.

After the mapping of the security requirements stated in the System Requirements and the Application Architecture, the technical threats of this basic architecture are analysed. The technical threats can be gathered independently of the application domain e.g. based on checklists [17, 18, 19]. Examples for technical threats in a concrete architecture are wiretaps, insider abuse of net access or system penetration. After being identified each technical threat is related with the application-level threats cross-checking the technical and the application-dependent level. The third step leads

to an estimation of associated risks (taking into account the technical opportunities of the attacker and the possible damage).

In the fourth step the security architecture is designed. Typically this comprises the following aspects:

- realisation of the security measures described in the Application Architecture on the chosen platform
- selection of special hardware devices (such as smart cards, key generators)
- access control of databases
- choice of predefined components or frameworks supporting security (like PGP, J2EE, etc.)
- introduction of measures ensuring the availability of system services and disaster recovery

Finally, the proposed measures have to be evaluated in their compliance with the requirements defined at the beginning of the phase.

For the development of the Software Architecture the security of the basic environment has to be taken into account as well. If the system runs on a platform which provides basic protection (e.g. through firewalls, intrusion detections, virus scanners) the related aspects do not have to be analysed in the context of the project. In the other case the basic protection has to be provided as part of the systems development.

Concerning the systematic transition from the Application Architecture to the Software Architecture the use of frameworks or model driven approaches is the primary choice. Their goal is the realisation of systems in a platform-independent style following the idea that the framework takes over the platform-dependent part of the work. A prominent exponent of such approaches is the J2EE-environment [20]. In SECTINO we develop a framework for developing secure workflows based on Web Services [25].

4 Conclusion

In the preceding sections we sketched a process model supporting the systematic development of security-critical systems within the framework of object oriented use case based modelling. Security analysis is integrated via a micro-process specifying a series of consecutive steps, which are repeatedly applied and refined through all the stages of the design process.

The comprehensive view of the whole design process across all layers of abstraction and its rigorous support of traceability distinguishes our approach to the development of secure systems from related work, e.g. [6, 21, 22, 23, 24, 26, 27, 28]. Our process model aims to integrate these existing approaches to security engineering.

Our goal is to develop a tool-supported process appropriate for industrial use. Positive results from pilot projects in an industrial context encourage us to move further to this direction. Currently, the process model is extended and elaborated by our groups in many directions, ranging from the formal modelling of access policies, the tool-supported management of requirements, threats and risks to the platform-independent development of security solutions to component framework (J2EE / .NET) related security issues.

References

[1] I. Jacobson, G. Booch, J. Rumbaugh: *The Unified Software Development Process*. Addison-Wesley, 1999.
[2] P. Kruchten: *The Rational Unified Process*. Addison-Wesley, 1999.
[3] D. D´Souza, A. Wills: *Components and Frameworks with UML – The Catalysis Approach*. Addison-Wesley, 1999.
[4] R. Breu, K. Burger, M. Hafner, G. Popp, J. Jürjens, G. Wimmel: *Security-Critical System Development with Extended Use Cases.* Accepted for APSEC03.
[5] D. Basin, J. Doser, T. Lodderstedt: *Model Driven Security for Process-Oriented Systems*. In *8th ACM Symposium on Access Control Models and Technologies*. ACM Press, 2003.
[6] D. Firesmith: *Security Use Cases*. In: Journal of Object Technology 2(3), 2003. http://www.jot.fm/issues/issue_2003_05/column6
[7] T. Lodderstedt, D. Basin, J. Doser: *Secureuml: A uml-based modeling language for model-driven security.* In: J.-M. Jézéquel, H. Hussmann, S. Cook (eds.): UML 2002. Lecture Notes in Computer Science, vol. 2460, Springer, 2002.
[8] www.v-modell.iabg.de
[9] R. Breu, K. Burger, M.Hafner, G. Popp: *Core Concepts of a Process Model for Security Engineering*. Accepted for Icssea 2003.
[10] G. Popp: *Vorgehensmodelle für die Entwicklung sicherer Systeme*. Dissertation, Munich University of Technology, to appear.
[11] R. Breu, G. Popp: *Actor-Centric Modeling of Access Rights.* Submitted for publication.
[12] J. Yoder, J. Barcalow: *Architectural Patterns for Enabling Application Security* . 4th Conference of Pattern Languages of Programs (PloP), 1997.
[13] E. Fernandez, R. Pan: *A Pattern Language for Security Models.* 8th Conference of Pattern Languages of Programs (PloP), 2001.
[14] B. Blakley: *Securtiy Design Patterns.* The OpenGroup. 2002. http://www.opengroup.org/security/gsp.htm
[15] M. Schumacher: *Security Engineering with Patterns.* PhD Thesis, Lecture Notes in Computer Science, LNCS 2754, Springer, 2003.
[16] M. Kis: *Information Security Antipatterns in Software requirements Engineering*. 9th Conference of Pattern Languages of Programs (PloP), 2002.
[17] J.D. Meier et al.:*, Improving Web Application Security, Threats and Countermeasures*. Microsoft Corporation, 2003.
[18] Bundesamt für Sicherheit in der Informationstechnologie: *IT Baseline Protection Manual.* Bonn, 2001. http://www.bsi.de/gshb/english/menue.htm
[19] T. R. Peltier: *Information Security Risk Analysis.* Auerbach, 2001.
[20] http://java.sun.com/j2ee/
[21] R. Anderson: *Security Engineering*. John Wiley, 2001.
[22] J. Jürjens: *Secure Systems Development with UML*. Springer, to appear.

[23] G. Sindre, A. Opdahl: *Templates for misuse case description*.In: Proc. Seventh International Workshop on Requirements Engineering: Foundation of Software Quality (REFSQ'2001), 2001.
[24] E.B. Fernandez, J.C. Hawkins: *Determining Role Rights from Use Cases*. Proc. ACM Workshop on Role-Based Access Control. Proceedings of the second ACM workshop on Role-based access control, United States, 1997.
[25] R. Breu, M. Hafner, B. Weber: *Modeling and Realizing Security-Critical Inter-Organizational Workflow.* Submitted for Publication.
[26] G. Sindre, G. G. Firesmith, A. L. Opdahl: A Reuse-Based Approach to Determining Security Requirements. In: Proc. 9th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'03), Klagenfurt/Velden, Austria, 2003.
[27]A. Toval, A. Olmos, M. Piattini: *Legal Requirements Reuse: A Critical Success Factor for Requirements Quality and Personal Data Protection*. In: Proc. IEEE Joint International Conference on Requirements Engineering (RE'02) Essen, Germany, 2002.
[28]J. A. Toval, J. Nicolás, B. Moros, F. Garcia: *Requirements Reuse for Improving Information Systems Security: A Practitioner's Approach.* Requirements Engineering (6), London, 2002.