International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:1, No:4, 2007

# RUPSec: An Extension on RUP for Developing Secure Systems - Requirements Discipline

Mohammad Reza Ayatollahzadeh Shirazi, Pooya Jaferian, Golnaz Elahi,
Hamid Baghi, Babak Sadeghian
{ashirazi, jaferian, elahi, baghi, basadegh}@ce.aut.ac.ir
Computer Engineering and IT department
Amirkabir University of Technology (Tehran Polytechnic)

***Abstract*—** The world is moving rapidly toward the deployment of information and communication systems. Nowadays, computing systems with their fast growth are found everywhere and one of the main challenges for these systems is increasing attacks and security threats against them. Thus, capturing, analyzing and verifying security requirements becomes a very important activity in development process of computing systems, specially in developing systems such as banking, military and e-business systems. For developing every system, a process model which includes a process, methods and tools is chosen. The Rational Unified Process (RUP) is one of the most popular and complete process models which is used by developers in recent years. This process model should be extended to be used in developing secure software systems. In this paper, the Requirement Discipline of RUP is extended to improve RUP for developing secure software systems. These proposed extensions are adding and integrating a number of Activities, Roles, and Artifacts to RUP in order to capture, document and model threats and security requirements of system. These extensions introduce a group of clear and stepwise activities to developers. By following these activities, developers assure that security requirements are captured and modeled. These models are used in design, implementation and test activities.

## I. Introduction

Security is an attribute of system that prevents the system from revealing, changing and denying of resource services and system information in an illegal way. Generally three aspects of security are: confidentiality, integrity and availability of service of resources and information. To achieve these aspects and develop a secure system, security services and mechanisms should be considered [1].

One of main activities in developing any computing system is requirements engineering. Requirements engineering is capturing, analyzing, documenting and validating of requirements. In requirements engineering security is considered as a nonfunctional requirement [2, 3, 4, 5]. Although in some references security is classified as a functional requirement [6, 7]. In most cases, security requirements are naturally difficult to identify, evaluate, apply and achieve [2].

Enforcing and managing security requirements requires professional abilities and wide knowledge, because the systems are built to run under attacks of unknown sources and therefore the requirements of systems might be unknown. One of the challenges in requirement engineering activities is capturing and analyzing nonfunctional requirements such as security, reliability, performance and usability requirements. Our experience shows that often these quality attributes are missed and are not captured by analysts. In practice, the usual method in secure systems development is the “penetrate and patch” approach. This means that developers attempt to remove the vulnerable points after the system is developed and attacks against the system and defects occur. This is a big risk and often imposes heavy costs and defects on software projects.

Usually it is possible to find the security needs and goals of an organization or a system in its security policy document. But there is no common, clear and defined method to transfer the facts in the Security Policy Document to precise and unambiguous requirements and then including security requirements in analysis, design, implementation and test phases of software development process. Researchers are trying to solve the mentioned problems. The main focus of current works in this area is using software patterns to capture and model security requirements [8, 9. 11] and common threats [10]. Also, in [2] a new modeling language is proposed for modeling security requirements [2]. According to the above discussion, it is reasonable to integrate and coordinate secure software development activities in the software process models [2].

In this work we have chosen Rational Unified Process (RUP) as target process model for security extension. We believe that this process model is one of the most complete and flexible process models. It is easy to understand and follow and most of the guidelines and activities in this process model is based on software engineering related standards that have been proposed by ISO and IEEE. We name the extended RUP for developing security systems as RUPSec. The aim of proposing the RUPSec is to define a software process model in which security requirements are considered in all development phases of a computer-based system: business modeling, requirements, analysis and design, implementation, and testing. In this paper a part of these extensions in Requirement discipline of RUP are presented and described. This paper is a summarized and improved version of work that has been presented in [12]. In that paper we have presented our extensions in Business Modeling Discipline and Requirements discipline.

Our main contributions in Requirements discipline of RUPSec are: identifying security threats against the system and the organization, capturing, modeling and evaluating security requirements. These extensions are presented considering analysis and design, implementation and test phases.

This paper organizes as follows. Section 2, provides a description of security requirements in RUP. Section 3 presents the case study that is used to provide examples of practical usages of proposed extensions. Section 4 is devoted to the new extensions in Requirements discipline of RUP. In Section 5 the extensions in the process model are evaluated. In Section 6, the extensions are compared with related works and similar models. The paper is concluded with Section 7 which contains a brief recapitulation of the main points.

## II. Security Requirements in RUP

In RUP FURPS+ model is used for categorizing requirements [6]. In this model, security requirements are categorized in Functionality requirement category. In RUP just some steps and an approach (Software Requirements Specification guideline, section 6) is given to establish and classify the security requirements. According to this guideline, captured security requirements are documented in Software Requirement Specification document, but it is not mentioned how these requirements should be modeled, analyzed and used in the remaining phases of development process. In the following sections, we describe how this problem is solved by the extensions that we have proposed for RUP.

## III. Motivation Case Study

One of the possible ways to evaluate software development process models or methodologies is to choose some exemplar systems as case studies and employ the process model or methodology in developing case study systems. Then, the weaknesses of the process model in developing the system are analyzed, the process model is improved and the system development is repeated according to the improved version of the process model. In this paper, a Sales and Purchase system of a dealer organization has been chosen as a case study. This organization offers some services to the sellers to demonstrate and sell the stocks. Customers can select and purchase the stocks from the sellers. In this paper, the examples are based on this case study system.

## IV. Extending Requirements Discipline of RUP

The goal of Requirements Discipline in RUP is to establish and maintain agreement with the customers and other stakeholders on what the system should do. In this activity system developers gain a better understanding of the system requirements and boundaries of the system are defined. In addition to common purposes, the purpose of this discipline for developing secure systems is: to capture and model security threats against the system, to propose security solutions for the threats and to elicit security requirements of the system.

In this discipline, it is supposed that the developers had followed the Business Modeling of RUPSec presented in [12]. Therefore the outputs of Business Modeling of RUPSec are used in Requirement Discipline. We also introduce a role for performing the required activities named "Security Expert Role" which is characterized in Table.1.

Table 1: Security Expert Role

| | |
|---|---|
| **Role** | Security Expert |
| **Role Type** | Additional Role |
| **Responsibilities** | Consultation in security issues, Development of security test cases |
| **Skills** | Familiar with security concepts, threats and counter-measures, Familiar with system analysis |

### A. *Finding Misactors and Misuse Cases: A New Activity in Requirements Discipline*

Along with finding use cases and actors, "Misuse-Cases" and "Misactors" should be identified. "Misactor" is an actor who threatens the system and misuses the system through a misuse case. As the functional requirements are described via use-cases, the activities that yield a security threat can be expressed as misuse cases.

RUP is a use-case driven approach for developing software [7]. Therefore, we suggest using Misuse Cases to find threats and security requirements. In the next steps misuse cases will form a basis for eliciting security requirements and security use cases. Therefore, a new activity named "Finding Misactors and Misuse-Cases" is added to RUP as an independent activity in the RUP's "Define the System" workflow detail. This activity should be done along with "Finding Use cases and Actors" activity. The inputs to this activity are "Threat Specification" and "Security Policy" [12] and the output artifact is a "Misuse-Case" model. The System Analyst role is responsible for performing this activity.

### B. *Finding System Threats: A New Activity in Requirements Discipline*

Finding Misuse-Cases and threats is an iterative activity. In other words the threats are completed with respect to Misuse-cases and Misuse-Cases are found and refined by means of threats. The Security Expert finds the threats against the system based on the experiences and knowledge of past projects. The Security Expert also presents a general solution to counter each threat. The misuse cases will be extracted using these threats and this process goes on iteratively.

With respect to the above discussions, a new activity called "Finding System Threats" is added to “Define the System” workflow of RUP. The inputs to mentioned activity are "Misuse case Model" and "Security Policy". The threats that are identified by the Security Expert are documented in

"Threat Specification Document" as output of the activity.

The "Threat Specification Document" is used by the System Analyst to extract the Misuse-Cases. Also it is recommended to record the "Threat Specification Document" in "Threat Repository" for future uses.

In Figure 1, the use cases and Misuse-Cases of the Sales system are presented. The use cases, which are identified by <<misuse case>> stereotype, illustrate threats against the system. These threats take place during the flow of use cases flow and their occurrence is illegal. Therefore, the dependency between use case and related Misuse-Case should be specified by <<extend>> relationship between use cases.

### *C. Refine Misuse Cases and Finding Security Use Cases: A New Activity in Requirements Discipline*

In order to find security requirements, we have added a new activity to the RUP, which is called "Refine Misuse cases and finding Security use cases" as an independent activity in "Define the system" workflow detail.

In this step, the System Analyst studies "Misuse case Model" and "Threat Specification Documents" to capture security use cases. In current activity, the steps of the misuse case are defined precisely. After that the system analyst specifies the solutions to confront the threats with respect to Threat Specification Document. These solutions are called "Security Use-Cases" and they should be described and documented. In current activity, the system analyst should avoid interfering technological aspects to select and describe security use cases; because using specific technology as a security use case will restrain finding better solutions in selecting architecture and design steps.

To document a Security Use-Case, the following points should be considered:

- Related threats should be specified.
- For each Security Use-Case, the flow of events should be studied in the three following viewpoints: System, Normal Actors and Misactors activities.

In this approach the system activities to provide the required security are captured and recorded as security requirements. In the Security Use-Case Description table which is presented in [9], post condition of each Security Use-Case is considered as a general security requirement. The purpose of mentioned requirement is to counter the related misuse case.

Usually, security use cases can be classified into several categories [9]. Also each category may have different paths [9]. To obtain security requirements from security use cases, each use case path should be analyzed independently and security requirements for each path are extracted.

In the use case diagram of Figure 2, for each Misuse-Case, we can present a number of solutions. These solutions are modeled by security use cases. Security Use-Cases are denoted by <<security>> stereotype. The dependency between Security Use-Case and Misuse-case is defined by <<prevent>> stereotype. This type of dependency specifies that security use case prevents the occurrence of Misuse-Case.

### *D. Refine Security Requirements: An Improvement on "Detail the Software Requirements"*

The system requirements will be obtained by describing each Security Use-Case. These requirements usually can be found in "System Actions" column (in security use case description table [9]) and Security use case post-condition.

The mentioned system requirements should be categorized and documented. Therefore, in "Detail the software requirements" activity of the RUP, the Requirement Specifier should document the security requirements in "Software Requirement Specification" document according to the requirement's type. Security requirements will form a basis for "Analysis and Design" discipline along with security use case model.

## V. Evaluation of presented extensions

There are various methods to evaluate a software development process model. Two common ways are evaluation based on case studies and Feature Based Evaluation. To evaluate proposed extensions, we use the criteria introduced in [13] for evaluating software engineering methodologies and evaluate our extension based on a subset of these criteria.

**Expressiveness**: a process model should be introduced in such a way to cover various aspects of a system. Whereas threats and security aspects like confidentiality, integrity and availability are not mentioned in RUP, in this paper some solutions are presented for modeling and documenting these aspects. In Table 2, these solutions are presented and compared to RUP.

TABLE 2: COMPARING RUP WITH EXTENDED PROCESS MODEL

| Studied aspect | Extended process model | RUP |
|---|---|---|
| Organization Security Policy | is documented in Security Policy | Is not documented |
| Threats against the system | Is modeled and documented in Threat Specification and Misuse Case document | Is not documented |
| Security aspects of system | Is modeled and documented using Security Use Cases | Is not documented |

**Preciseness:** A process model should be unambiguous, that is, it should be possible to use it in a correct way. In presented extensions inputs, outputs, time to do, related discipline, and role who dose the activities are declared and integrated with RUP. This integration prevents an ambiguity in extensions.

**Accessibility :** A process model should be practical for various groups of developers with different skills and level of knowledge. As RUP is used widely and UML is a universal modeling language among developers, the presented extensions are based on RUP and modeling language is UML. Therefore, using the extensions is easy for developers who are familiar with RUP and UML.

**Portability:** A process model should not depend on implementation language, special technology or architecture. In presented extensions in this paper, it is emphasized on using standard modeling languages such as UML and common process models such as RUP. Therefore, they can be

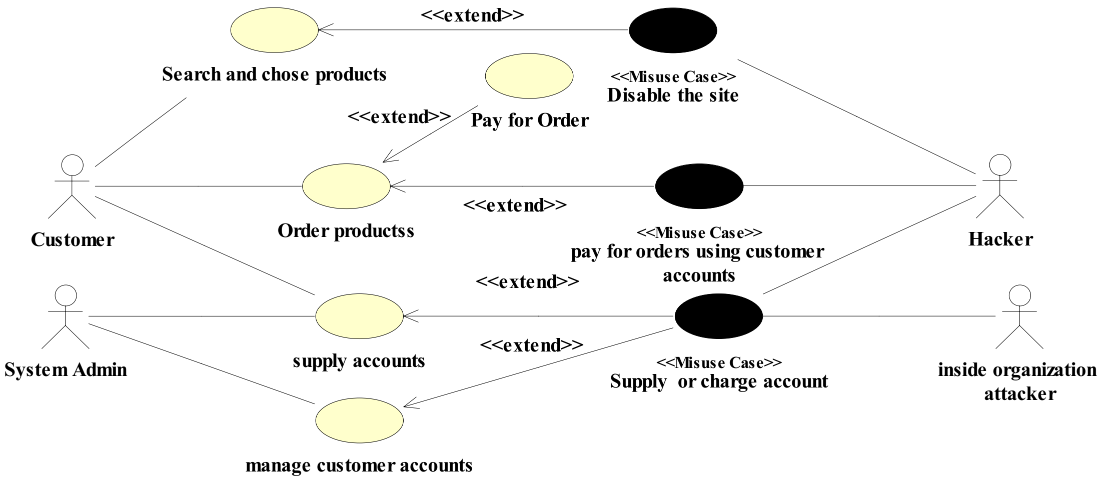

**Figure 1: Misuse Case of Sale System**

used in any type of projects.

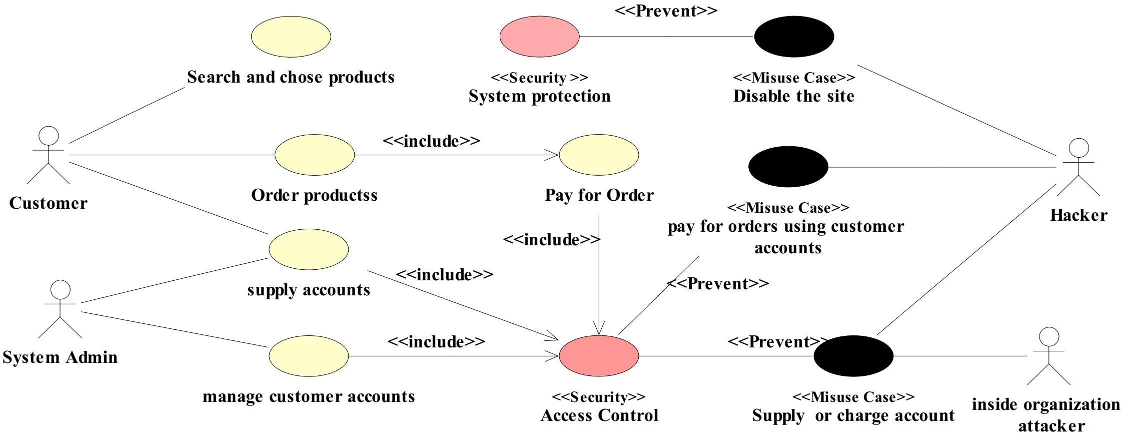

**Figure 2: Security Use Case Diagram For Sales System**

## I. COMPARISON WITH RELATED WORKS

In previous sections the extensions and improvements on RUP were introduced. A number of the discussed aspects, phases and activities are mentioned in [2,8,9,14,15,16,17,18] too, but in this paper we have investigated to integrate existing attempts and works in the framework of RUP.

In [15], the presented process model is not based on a reference and specified process model. As in [15] activities, artifacts, and roles are not specified in detail, developers have to integrate the mentioned process models with their own process model. In this paper, all extended activities and artifacts are incorporated with RUP and are provided for developers.

In [14], use cases are extended to cover security requirements, but it is not discussed for what threats the system should be protected. In [17] the aspect of Abuse Case is introduced but no way for including these threats in analysis and design model is suggested. In [9] in addition to modeling Misuse Cases, a number of Security Use Cases are assigned to normal Use Cases to protect the system against the threats. In the our work we have tried to eliminate the weaknesses of the works reported in [9, 14, 17] by using Misuse Cases and Security Use Cases and Threat Specification.

## II. Conclusion and suggestions for further works

In this paper some extensions on Requirement discipline of RUP were reported as a part of research work on defining a process model for developing secure systems, RUPSec. These extensions include adding activities, artifacts, and roles to RUP or improving them. In current stage, these extensions do not cover all RUP disciplines and in further steps of research we will work on other RUP disciplines. In the next phases of work, captured Misuse Cases will be realized in analysis and design discipline. System attacks are identified whereas they are realized. Security Use Cases will be realized and used to specify Analysis Mechanism and design against threats. Also Misuse Cases will be used in test phase to generate Test Cases.

## References

[1] Matt Bishop, Computer Security, Art & Science, Addison-Wesley, First Edition, 2002

[2] J. J¨urjens. Secure Systems Development with UML. Springer, To be published. 2004.

[3] Shreyas Doshi, Software Engineering and Security: Towards Architecting Secure Software, a graduate term paper for ICS 221-Seminar in Software Engineering, University of California, Irvine, 2001.

[4] Lawrence Chung Brian A. Nixon, Dealing with Non-Functional Requirements: Three Experimental Studies of a Process-Oriented Approach ,International Conference on Software Engineering 1995.

[5] Barbara Paech, Allen H. Dutoit, Daniel Kerkow, Antje von Knethen ,Functional requirements, non-functional requirements, and architecture should not be separated, 8th International Workshop on Requirements Engineering: Foundation for Software Quality, Essen, Germany, 2002

[6] Robert Grady, Practical Software Metrics for Project Management and Process Improvement, Prentice Hall, 1992

[7] Philippe Kruchten, The Rational Unified Process: An Introduction, Third Edition, Addison-Wesley Pub Co, 2003.

[8] Donald G. Firesmith, Engineering Security Requirements, Journal of Object Technology, Vol. 2, No. 1, January-February 2003.

[9] Donald G. Firesmith, Security Use Cases, Journal Of Object Technology, Vol. 2, No. 3, May-June 2003.

[10] Robert J. Ellison Richard C. Linger, Andrew P. Moore Attack Modeling for Information Security and Survivability CMU/SEI-2001-TN-001, 2001

[11] Jeffrey Barcalow, Joseph Yoder Architectural Patterns for Enabling Application Security, The 4th Pattern Languages of Programming Conference 1997.

[12] H. Baghi, P. Jaferian, G. Elahi, M.R. Shirazi, B. Sadeghian, An Extension on RUP for Developing Secure Systems, *Proceedings of the 10th Annual International CSI Computer Conference*, 2005

[13] Onn Shehory, Arnon Sturm, Evaluation of modeling techniques for agent-based systems, Proceedings of the fifth international conference on Autonomous agents,2001

[14] G. Popp and J. J¨urjens and G. WimmelR. Breu. Security-Critical System Development with Extended Use Cases, Tenth Asia-Pacific Software Engineering Conference, 2003

[15] Ruth Breu, Klaus Burger, Michael Hafner, Jan Jürjens, Gerhard Popp, Guido Wimmel, Volkmar Lotz , Key Issues of a Formally Based Process Model for Security Engineering, 16th International Conference "Software & Systems Engineering & their Applications" (ICSSEA), 2003.

[16] Premkumar T. Devanbu, Stuart Stubblebine. Software engineering for security: a roadmap, ICSE - Future of SE Track ,2000.

[17] John McDermott and Chris Fox Using Abuse Case Models for Security Requirements Analysis, Proceedings of the 15th Annual Computer Security Applications Conference ,1999.

[18] Gunnar Petterson, Collaboration in a Secure Development Process – Part I, Information Security Bulletin, June 2004.

[19] Ruth Breu, Klaus Burger, Michael Hafner, Gerhard Popp, Towards a Systematic Development of Secure Systems, WOSIS, 2004.

[20] J¨urgen Doser , Torsten Lodderstedt Model Driven Security For Process oriented Systems David Basin, Proceedings of the eighth ACM symposium on Access control models and technologies, 2003.

[21] Ross Anderson ,Security Engineering , a guide to building dependable system. Wiley, 2001.