Incorporating Security Requirements Engineering into Standard Lifecycle Processes

Nancy R. Mead
Carnegie Mellon University
nrm@sei.cmu.edu

Venkatesh Viswanathan
Carnegie Mellon University
vviswan1@andrew.cmu.edu

Justin Zhan
Carnegie Mellon University
justinzh@andrew.cmu.edu

Abstract

This paper gives an overview of various standard lifecycle development processes. It then provides a roadmap for developing security-critical projects using Rational Unified Process as a framework for development. The Security Quality Requirements Engineering (SQUARE) methodology provides a way to address security issues early in the development lifecycle. SQUARE can be more effective when it fits into an organization's existing development process. Hence this paper describes a way to fit the SQUARE methodology into the Rational Unified Process.

1. Introduction

It is well recognized in the software industry that requirements engineering is critical to the success of any major development project. Security requirements are often identified during the system lifecycle. However, the requirements tend to be general mechanisms such as password protection, firewalls, and virus detection tools. Often the security requirements are developed independently of the rest of the requirements engineering activity, and hence are not integrated into the mainstream of the requirements activities. As a result, security requirements that are specific to the system and that provide for protection of essential services and assets are often neglected. The requirements elicitation and analysis that is needed to get a better set of security requirements seldom takes place.

The Software Engineering Institute's CERT Program at Carnegie Mellon University has developed a methodology to help organizations build security into the early stages of the production lifecycle. The Security Quality Requirements Engineering (SQUARE) Methodology [1] consists of nine steps that generate a final deliverable of categorized and prioritized security requirements. Although the SQUARE Methodology could likely be generalized to any large-scale design project, it was designed for use with information technology systems.

The SQUARE process involves the interaction of a team of requirements engineers and the stakeholders of an IT project. The requirements engineering team can be thought of as external consultants, though often the team is composed of one or more internal developers of the project. When SQUARE is applied, the user should expect to have identified, documented, and inspected relevant security requirements for the system or software that is being developed. SQUARE may be more suited to a system under development or one undergoing major modification than one that has already been fielded, although it has been used both ways. Software life-cycle models describe phases of the software cycle and the order of execution of those phases. Many models are being adopted by software companies, but most of them have similar patterns. Typically each phase produces deliverables required

by the next phase in the life cycle. Requirements are translated into design. Code is produced during the implementation phase and is driven by the design. Code is finally tested against requirements to ensure quality.

2. Lifecycle processes

This paper focuses on incorporating SQUARE with standard life-cycle models, as SQUARE can be more effective when it fits into an organization's existing development process. The most commonly adopted life-cycle models and process methodologies, namely, the waterfall model, Rational Unified Process (iterative and incremental model), spiral model, and Dynamic Systems Development Method (agile methodology—iterative and incremental model) are briefly described and then SQUARE with RUP is explained in detail.

2.1. Waterfall model

The first process model to be considered is the waterfall approach. It was proposed by Royce in 1970 and is still widely followed in software engineering. The waterfall model is a sequential software development model that divides the process of software development into these phases:

1. Requirements Analysis and Specification
   In this phase, the problem is specified along with the goals and constraints. The output of this phase is the requirements specification document containing the necessary requirements and their goals and constraints.
2. Software Design
   The system specifications are translated into a software representation. The system architecture is defined, along with the detailed design of the product to be developed. The hardware requirements are also determined at this stage. By the end of this stage, there should be a clear relationship between the hardware, software, and the associated interfaces. The output of this phase is the software design document.
3. Implementation and Unit Testing
   The software design is translated into the software domain. The module is unit tested.
4. Integration and System Testing
   All the program units are integrated and tested to ensure that the system completely satisfies the software requirements. After this stage the software is delivered.
5. Operation and Maintenance
   In this phase the software is updated to meet the changing customer needs. The overall efficiency of the product is enhanced by correcting errors that were not detected in the testing phase.

2.2. Spiral model

The spiral model combines the iterative nature of prototyping (a rework scheduling strategy in which time is set aside to revise and improve parts of the system [2]) with the controlled and systematic aspects of the waterfall model [3]. Software is therefore developed in incremental releases, which could range from paper prototypes in the initial phases to more complex versions of the product towards the end of the project. The spiral model [4] helps to identify and achieve quality objectives during development.

The spiral model consists of four phases of development: planning, risk analysis, engineering, and evaluation. Each cycle of the spiral typically begins with identification of objectives, alternatives, and constraints. This is followed by identifying areas of uncertainty that are sources of project risks and evaluating and resolving them. Prototypes are generated as a means of reducing risks. Next, the appropriate model is chosen for the next phase of development. Finally, the project is reviewed and plans are drawn for the next round of the spiral.

2.3. Agile development

This section introduces agile development methodology and its best practices in terms of requirements handling. Some of the agile methodologies are introduced here, and those that best fit with SQUARE have been identified and elaborated.

Agile methodologies have gained a lot of importance in recent years, and many organizations now follow agile principles. Agile methodologies minimize risk through development of software in a short amount of time.

Working software is the primary measure of progress in agile. The Agile Alliance declares the following as the core principles of agile development [5]:

> *Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.*
>
> *Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.*
>
> *Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter time scale.*
>
> *Business people and developers must work together daily throughout the project.*
>
> *Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.*
>
> *The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.*
>
> *Working software is the primary measure of progress.*
>
> *Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.*
>
> *Continuous attention to technical excellence and good design enhances agility.*
>
> *Simplicity—the art of maximizing the amount of work not done—is essential.*

The best architectures, requirements, and designs emerge from self-organizing teams.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Requirements are handled very differently in agile as compared to standard life-cycle methodologies such as waterfall, spiral, and RUP. Agile requirements modeling is built on the fact that requirements keep changing regardless of the time spent to elicit the specifications. Developers clarify the requirements directly from the customer rather than from the requirements specification document. Hence agile requirements modeling focuses on collecting barely enough requirements to start with and then undergoing a change management process as new requirements emerge. Some of the important principles of agile requirements are

- continuous customer interaction
- modeling requirements in iterations
- prioritizing requirements
- converting requirements to test cases
- validating with prototypes and review meetings
- managing change in requirements

2.4 Rational Unified Process

Nowadays many companies have adopted Rational Unified Process (RUP) as a reference framework for their development process. RUP is not a single concrete prescriptive process but rather an adaptable process framework, intended to be tailored by the development organizations and software project teams selecting the elements of the process that are appropriate for their needs. The Rational Unified Process divides the lifecycle into four consecutive phases, namely, Inception, Elaboration, Construction, and Transition. Every phase has an associated milestone, which when achieved completes a critical set of activities. We will expand on this discussion in section 4.

3. SQUARE in a nutshell

The SQUARE Methodology begins with the requirements engineering team and project stakeholders agreeing on technical definitions that serve as a baseline for all future communication. Next, business and security goals are outlined. Third, artifacts and documentation are created, which are necessary for a full understanding of the relevant system. A structured risk assessment determines the likelihood and impact of possible threats to the system. Following this work, the requirements engineering team determines the best method for eliciting initial security requirements from stakeholders, which is dependent on several factors, including the stakeholders involved, the expertise of the requirements engineering team, and the size and complexity of the project. Once a method has been established, the participants rely on artifacts and risk assessment results to elicit an initial set of security requirements. Two subsequent stages are spent categorizing and prioritizing these

requirements for management's use in making tradeoff decisions. Finally, an inspection stage is included to ensure the consistency and accuracy of the security requirements that have been generated.

Table 1 [6] details the steps involved in the SQUARE process.

Table 1. SQUARE steps

Step 1: Agree on definitions
Input: Candidate definitions from IEEE and other standards **Technique:** Structured interviews, focus group **Participant:** Stakeholders, requirements team **Output:** Agreed-to definitions
Step 2: Identify assets and security goals
Input: Definitions, candidate goals, business drivers, policies and procedures, examples **Technique:** Facilitated work session, surveys, interviews **Participant:** Stakeholders, requirements engineer **Output:** Assets and goals
Step 3: Develop artifacts to support security requirements definition
Input: Potential artifacts (e.g., scenarios, misuse cases, templates, forms) **Technique:** Work session **Participant:** Requirements engineer **Output:** Needed artifacts: scenarios, misuse cases, models, templates, forms
Step 4: Perform risk assessment
Input: Misuse cases, scenarios, security **Technique:** Risk assessment method, analysis of anticipated risk against organizational risk tolerance, including threat analysis **Participant:** Requirements engineer, risk expert, stakeholders **Output:** Risk assessment results
Step 5: Select elicitation techniques
Input: Goals, definitions, candidate techniques, expertise of stakeholders, organizational style, culture, level of security needed, cost/benefit analysis, etc. **Technique:** Work session **Participant:** Requirements engineer **Output:** Selected elicitation techniques
Step 6: Elicit security requirements
Input: Artifacts, risk assessment results, selected techniques **Technique:** Joint Application Development (JAD), interviews, surveys, model-based analysis, checklists, lists of reusable requirements types, document reviews **Participant:** Stakeholders facilitated by requirements engineer **Output:** Initial cut at security requirements
Step 7: Categorize requirements as to level (system, software, etc.) and whether they are requirements or other kinds of constraints
Input: Initial requirements, architecture **Technique:** Work session using a standard set of categories **Participant:** Requirements engineer, other specialists as needed **Output:** Categorized requirements

Step 8: Prioritize requirements
Input: Categorized requirements and risk assessment results **Technique:** Prioritization methods such as Analytical Hierarchy Process (AHP), Triage, Win-Win **Participant:** Stakeholders facilitated by requirements engineer **Output:** Prioritized requirements
Step 9: Inspect requirements
Input: Prioritized requirements, candidate formal inspection technique **Technique:** Inspection method such as Fagan, peer reviews **Participant:** Inspection team **Output:** Initial selected requirements, documentation of decision making process and rationale

4. RUP in a nutshell

RUP is a prescriptive process [7]. It is a generic process framework that adopts sound principles of software engineering and provides well-defined guidelines for software development. It implies the following best practices [8].

1. Develop software iteratively.
2. Manage requirements.
3. Use component-based architectures.
4. Visually model software.
5. Verify software quality.
6. Control changes to software.

The Rational Unified Process supports risk driven development [9]. It advocates developing the software iteratively to mitigate high risks in each stage and provides for frequent releases and continuous involvement of the end user. RUP prescribe ways to elicit, analyze, and manage requirements. It supports the use of the Unified Modeling Language (UML) as a means for communicating requirements, architecture, and design.

Quality assurance is integrated into the process. Guidelines are provided to completely assist in planning, analysis, design, and execution.

The Rational Unified Process divides the development cycle in four consecutive *phases* [8]:

- Inception phase
- Elaboration phase
- Construction phase
- Transition phase

Each phase is associated with a *milestone*, which when achieved completes a critical set of activities. Each phase serves a specific purpose. The Inception phase primarily involves identification of business case. Basic use case model, project plan, initial risk assessment, and project description in terms of core project requirements, constraints, and key features are also generated. In the Elaboration phase, the architecture of the project is identified. The risks and requirements identified in the Inception phase are revised and prioritized in this phase. The Construction phase involves development of the designed components. Hence this is where the bulk of the coding occurs. In the Transition phase, the product has moved to the end user. This phase includes end user training and beta testing of the system to validate it against the end users' expectations. Discussion of the Construction and Transition phases is beyond the scope of this paper, as our focus is on requirements engineering. Hence discussion is restricted to outcomes [8] of the Inception and Elaboration phases.

The outcome of the *Inception phase* is:

- A vision document: a general vision of the core project's requirements, key features, and main constraints
- An initial use-case model (10%-20% complete)
- An initial project glossary (may optionally be partially expressed as a domain model)
- An initial business case, which includes business context, success criteria (revenue projection, market recognition, and so on), and financial forecast
- An initial risk assessment
- A project plan, showing phases and iterations
- A business model, if necessary
- One or several prototypes
- The outcome of the *Elaboration phase* is:
- A use-case model (at least 80% complete) — all use cases and actors have been identified, and most use-case descriptions have been developed
- Supplementary requirements capturing the non-functional requirements and any requirements that are not associated with a specific use case
- A Software Architecture Description
- An executable architectural prototype
- A revised risk list and a revised business case
- A development plan for the overall project, including the coarse-grained project plan, showing iterations and evaluation criteria for each iteration
- An updated development case specifying the process to be used
- A preliminary user manual (optional)

5. RUP with SQUARE

The Rational Unified Process captures non-functional requirements as supplementary specification during the Elaboration phase. A project for which security is critical requires more details to be captured early in the lifecycle to successfully address security concerns. SQUARE provides a way to capture security requirements early in the development lifecycle. Assuming RUP as the process framework adopted, it is important that the steps of SQUARE go hand–in-hand with the steps in RUP. SQUARE fits into the Inception and Elaboration phases, as it purely deals with requirements engineering.

5.1. Inception phase

The business case of the system and the scope of the project are decided during the Inception phase. The external interfaces, nature of interaction, success criteria, risk assessment, and phase plan should be identified to accomplish the business case. Prior to capturing the significant interactions, the security definitions best suited for the organization should be identified and agreed upon. After agreeing upon the definitions, security goals will be identified, along with the high-level project goals to be achieved for the successful completion of the project. Without overall security goals for the project, it is impossible to identify the priority and relevance of any security requirements that are generated.

Artifacts are developed, which are necessary for a full understanding of the relevant system. These are some of the potential artifacts that can be developed:

- system architecture diagrams
- use case scenarios/diagrams
- misuse case scenarios/diagrams
- attack trees
- standardized templates and forms

One of the main aspects of RUP is that it is risk driven and risk is identified in every phase. SQUARE also emphasizes identification of risks pertaining to security. The purpose of this step in the SQUARE process is to identify the vulnerabilities and threats to the system, the likelihood that the threats will materialize as real attacks, and any potential consequences of such an attack. After the threats have been identified by the risk assessment method, they must be classified according to likelihoods. This will aid in prioritizing the security requirements that are generated at a later stage. A traceability matrix can also be developed indicating a clear mapping between the risk and security requirements. This mapping might result in identifying additional security requirements. The next step would be to elicit requirements. The interaction of the entities with the system will be captured and modeled by the requirements team. One of the best practices suggested by RUP is to identify requirements using use cases. It is obvious from the fact that RUP recommends using UML for communicating requirements. Identify the external interfaces interacting with the system and establish the functional use case. Also, choose an appropriate elicitation technique and gather security requirements in terms of misuse cases that support the goals and security definitions already collected. The details that can be captured in misuse cases are shown in Table 2.

Table 2. Misuse case template

Misuse case ID	
Name	
Priority	Low or Medium or High
Scope	
Development Environment	
Mis-actors:	
Access Right Levels:	Low-Level System Users or/and Medium-Level System Users or/and High-Level System Users or/and Sys Admin or/and Other Network User
Point of Entry:	
Security Attributes Affected:	Confidentiality or/and Integrity or/and Availability
Description:	
Sophistication:	Low or Medium or High
Pre-conditions:	
Assumptions:	
Post-conditions:	
Potential Mis-actor Profiles:	
Stakeholders and Threats:	
Related Use Cases:	
Related Threats:	
Architectural Recommendation:	

5.2. Outcomes

Along with the outcomes suggested by the Inception phase of RUP, the following outcomes will be observed.

- Agree on definitions
- Security goals
- Initial risk assessment includes security risk
- Initial use case model including security misuse cases (20–30% complete)

The first milestone (Lifecycle objective) review process in RUP should include the above mentioned outcomes.

5.3. Elaboration phase

Most of the security requirements should be incorporated into the misuse case model and described in this phase. The security requirements are refined and categorized based on level (software, system, etc.). The purpose of this step is to allow the requirements engineer and stakeholders to classify the requirements as essential, non-essential, system level, software level, or as architectural constraints.

At a minimum, the requirements engineering team can provide to the stakeholders a matrix [6] as shown in Table 3.

Table 3. Categorization of requirements

	System level	Software level	Architectural constraint
Essential			
Non-essential			

The risk list is revisited and revised and assessed again. The security requirements are then prioritized and validated with the stakeholders along with the other requirements. These requirements act as an important input when making architecture decisions, and hence the security requirements have to be categorized, prioritized, and validated before deciding on the architecture. Other non-functional requirements are captured as usual in a supplementary specification document. The elaboration phase activities ensure that the architecture, requirements, and project plans are stable enough and the risks are sufficiently mitigated, which will ultimately help to determine the cost and schedule for the completion of development.

5.4. Outcomes

Along with the outcomes suggested by the Elaboration phase of RUP, the following outcomes will be observed.

- Use case model (functional requirements + security requirements described in misuse cases, 80% complete)
- Revised risk list
- Categorized and prioritized security requirements

The second milestone (Lifecycle Architecture) review process in RUP should include the above mentioned outcomes.

Table 4 summarizes integration of the SQUARE methodology into RUP.

Table 4. Summary of RUP with SQUARE

Inception
Agree on definitions (Step 1) The requirements engineering team and project stakeholders agree on technical definitions that serve as a baseline for all future communication. **Identify security goals (Step 2)** Security goals for the project are clearly outlined along with business goals. **Develop artifacts to support security requirements definitions (Step 3)** Create artifacts necessary for full understanding of the system. **Perform risk assessment (Step 4)** The initial risk assessment of RUP should also include security risk. The identified security risk should then be prioritized. **Select elicitation technique (Step 5)** Select elicitation technique best suited for the project. **Elicit security requirements (Step 6)** Gather initial security requirements in terms of misuse cases that support the goals and security definitions already collected. **Inspect requirements (Step 9)** The first milestone review process of RUP should include requirements inspection on gathered security requirements.
Elaboration
Perform risk assessment (Step 4) This step is revisited to develop a revised risk list. Additional security risks are captured and risks are prioritized. **Elicit security requirements (Step 6)** This step is revisited to capture additional security requirements.
Elaboration
Categorize requirements as to level (system, software) and whether they are requirements or other kind of constraints (Step 7) Categorize the elicited security requirements. **Prioritize security requirements (Step 8)** The categorized security requirements are prioritized. Prioritizing techniques such as Analytical Hierarchical Process (AHP) can be used. **Inspect requirements (Step 9)** The second milestone review process of RUP should include inspection of the security requirements.
Construction
Not Applicable
Transition
Not Applicable

6. Conclusion

Security requirements are of paramount importance, as they directly reflect the quality of the product. Current processes support capturing requirements like security, but it is often an afterthought relative to functional (end user) requirements. For instance, financial projects have critical security issues that need to be elaborated in security requirements. SQUARE provides an efficient way to capture such security requirements. Since RUP is a reference framework, it is usually customized by the organization to suit project needs. Likewise, RUP with SQUARE can be considered as a framework for security-critical projects.
We have shown how SQUARE can also be combined with other processes, such as the Waterfall model, Spiral model, and Agile processes [10]. Considering RUP with SQUARE as an example framework, any organization can fit SQUARE into their process methodology.

6. References

[1] http://www.cert.org/sse/square.html
[2] “Iterative and incremental development,” Wikipedia. http://en.wikipedia.org/wiki/Iterative_development
[3] “Waterfall model,” Wikipedia. http://en.wikipedia.org/wiki/Waterfall_Model
[4] B. Boehm, “A Spiral Model of Software Development and Enhancement,” *IEEE Computer*, vol. 21, no. 5, pp. 61-72, May 1988.
[5] “Principles behind the Agile Manifesto,” 2001. http://agilemanifesto.org/principles.html
[6] Mead, N. R., E. Hough, and T. Stehney. *Security Quality Requirements Engineering (SQUARE) Methodology* (CMU/SEI-2005-TR-009). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2005. http://www.sei.cmu.edu/publications/documents/05.reports/05tr009.html
[7] Kruchten, P. *The Rational Unified Process: An Introduction*, 3rd ed. Boston, MA: Addison-Wesley, 2003.
[8] *Rational Unified Process: Best Practices for Software Development Teams*. Rational Software White Paper TP026B, Rev 11/01, 2001. http://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf
[9] Ambler, S. W. *A Manager’s Introduction to Rational Unified Process*, 2005. http://www.ambysoft.com/downloads/managersIntroToRUP.pdf
[10] Mead, N.R., Viswanathan, V., Padmanabhan, D., and Raveendran, A., Incorporating Security Quality Requirements Engineering (SQUARE) into Standard Life-Cycle Models, (CMU/SEI-2008-TR-006). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2008. http://www.sei.cmu.edu/publications/documents/08.reports/08tn006.html

7. Acknowledgment

NO WARRANTY
ANY CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL CONTAINED HEREIN IS FURNISHED ON AN “AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL.
CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

Authors

Nancy R. Mead is a senior member of the technical staff in the Survivable Systems Engineering Group, which is part of the CERT Program at the Software Engineering Institute (SEI). Mead is also a faculty member in the Master of Software Engineering and Master of Information Systems Management programs at Carnegie Mellon University. Her research interests are in the areas of information security, software requirements engineering, and software architectures.

Mead has more than 100 publications and invited presentations. She is a Fellow of the Institute of Electrical and Electronic Engineers, Inc. (IEEE) and the IEEE Computer Society and is also a member of the Association for Computing Machinery (ACM). Dr. Mead received her PhD in mathematics from the Polytechnic Institute of New York, and received a BA and an MS in mathematics from New York University.

Justin Zhan is a faculty member at Carnegie Mellon University. His research interests include privacy and security aspect of data mining, privacy and security issues in bioinformatics, privacy-preserving scientific computing, privacy-preserving electronic business, artificial intelligence applied in the information security domain, data mining approaches for privacy management, and security technologies associated with compliance and security intelligence. He has served as an editor/advisory/editorial board member for a number of international journals and a committee chair/member for many international conferences.