Software Engineering Institute

Incorporating Security Quality Requirements Engineering (SQUARE) into Standard Life-Cycle Models

Nancy R. Mead
Venkatesh Viswanathan
Deepa Padmanabhan
Anusha Raveendran

May 2008

TECHNICAL NOTE
CMU/SEI-2008-TN-006

CERT Program

http://www.sei.cmu.edu

Carnegie Mellon

This report was prepared for the

SEI Administrative Agent
ESC/XPK
5 Eglin Street
Hanscom AFB, MA 01731-2100

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

This work is sponsored by the U.S. Department of Defense. The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

This work is sponsored by Carnegie Mellon CyLab.

For information about purchasing paper copies of SEI reports, please visit the publications portion of our Web site (http://www.sei.cmu.edu/publications/pubweb.html).

Table of Contents

List of Figures

List of Tables

Abstract

SQUARE (Security Quality Requirements Engineering) is a method for eliciting and prioritizing security requirements in software development projects. This report describes how SQUARE can be incorporated in standard life-cycle models for security-critical projects. Life-cycle models and process methods considered for the report are the waterfall model, Rational Unified Process, the spiral model, and Dynamic Systems Development Method (an agile method).

This report is for information technology managers and security professionals, management personnel with technical and information security knowledge, and any personnel who manage security-critical projects that follow standard life-cycle models.

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 life cycle. 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 CERT Program at the Carnegie Mellon® Software Engineering Institute has developed a methodology to help organizations build security into the early stages of the production life cycle. The Security Quality Requirements Engineering (SQUARE) methodology 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 stakeholders can expect it to result in the identification, documentation, and inspection of 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. This report 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 considered and explained in detail.

1.1 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 summarizes the steps in the SQUARE process. SQUARE is described in detail in *Security Quality Requirements Engineering (SQUARE) Methodology* [1].

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 security goals
Input: Definitions, candidate goals, business drivers, policies and procedures, examples Technique: Facilitated work session, surveys, interviews Participant: Stakeholders, requirements engineer Output: 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 Triage, Win-Win Participant: Stakeholders facilitated by requirements engineer Output: Prioritized requirements
Step 9: Requirements inspection
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

2 Life-Cycle Model: Waterfall

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.

These phases are cascaded as shown in Figure 1. The second phase is started only when the defined set of goals are achieved for the first phase; hence the name “waterfall model.”

From the figure, one can see that feedback loops allow for corrections to be incorporated into the model. For instance, any problem or update in the design phase requires a revisit to the specification phase. When changes are made at any phase, the relevant documentation should be updated to reflect that change.

The waterfall model, being a disciplined approach and documentation-driven method, has been criticized. There is a high probability that the client is not aware of all the requirements up front. The client might require a prototype with which to explicate exactly what is required for the project. In waterfall, however, transition to the next phase occurs only when the previous phase is completed, so the client might see the product only at the end. The notion of iterative development and prototyping is not followed in this model.

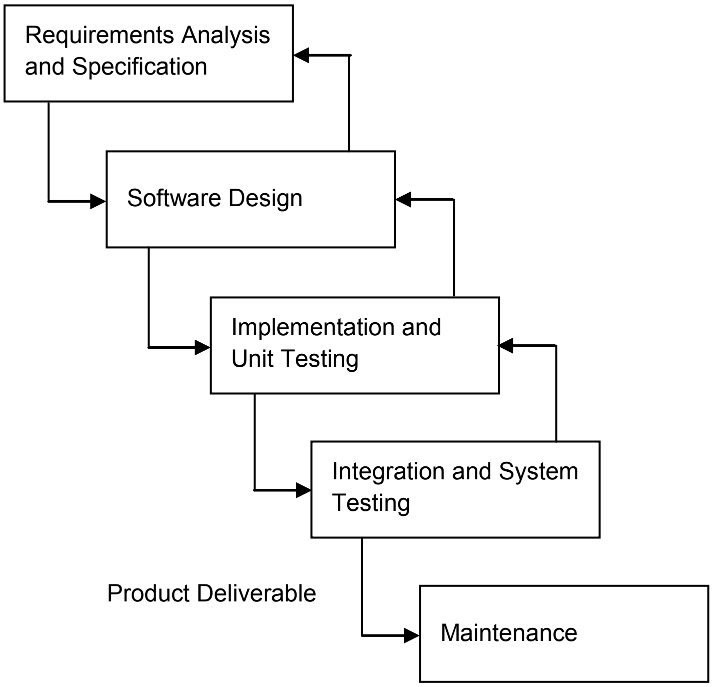
Requirements Analysis and Specification
Software Design
Implementation and Unit Testing
Integration and System Testing
Product Deliverable
Maintenance

Figure 1: Waterfall Model

2.1 WATERFALL WITH SQUARE

Since waterfall is a sequential model, incorporating SQUARE into it is straightforward. All nine steps of SQUARE fall under the requirements analysis and specification phase. The software requirements specification (SRS) should accommodate the outcome of the first eight SQUARE steps. The SRS must clearly specify the security definitions agreed on (Step 1). It is necessary to document security goals (Step 2) along with the project goals and constraints. Develop artifacts (Step 3) such as misuse cases and scenarios to support security requirements definition. Risk assessment (Step 4) is not prescribed in the waterfall model, so the fourth step of SQUARE is an addition to the model. However, risk assessment can be restricted to security-related threats. Once the elicitation technique (Step 5) is identified, elicit (Step 6) and document the security requirements along with the functional requirements. Categorize the security requirements (Step 7) and prioritize them (Step 8) along with documented functional requirements. (For clarity, it is preferable to separate security requirements from functional requirements.) Finally, requirements inspection (Step 9) is carried out to ensure completeness of all steps of SQUARE. This is an exit criterion for the phase, and only when it is satisfied is the next step carried out. Any update in the design phase may result in revisiting the requirements analysis and specification phase.

Figure 2 summarizes the use of the waterfall model with SQUARE.

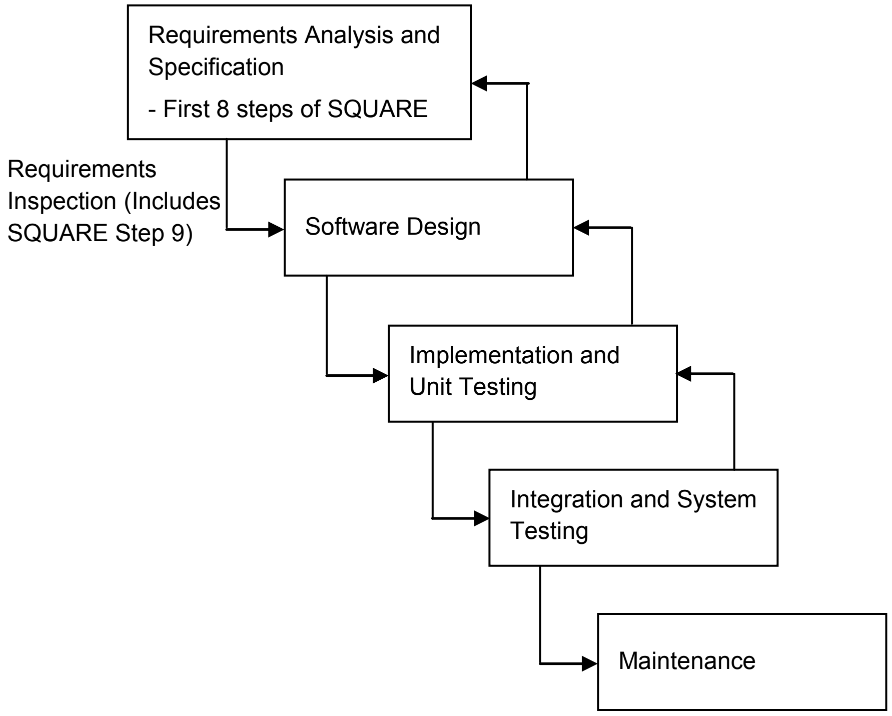
Requirements Analysis and Specification
- First 8 steps of SQUARE
Requirements Inspection (Includes SQUARE Step 9)
Software Design
Implementation and Unit Testing
Integration and System Testing
Maintenance

Figure 2: Waterfall Model with SQUARE

3 Life-Cycle Model: Spiral

The spiral model, as depicted in Figure 3, 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 helps to identify and achieve quality objectives during development.

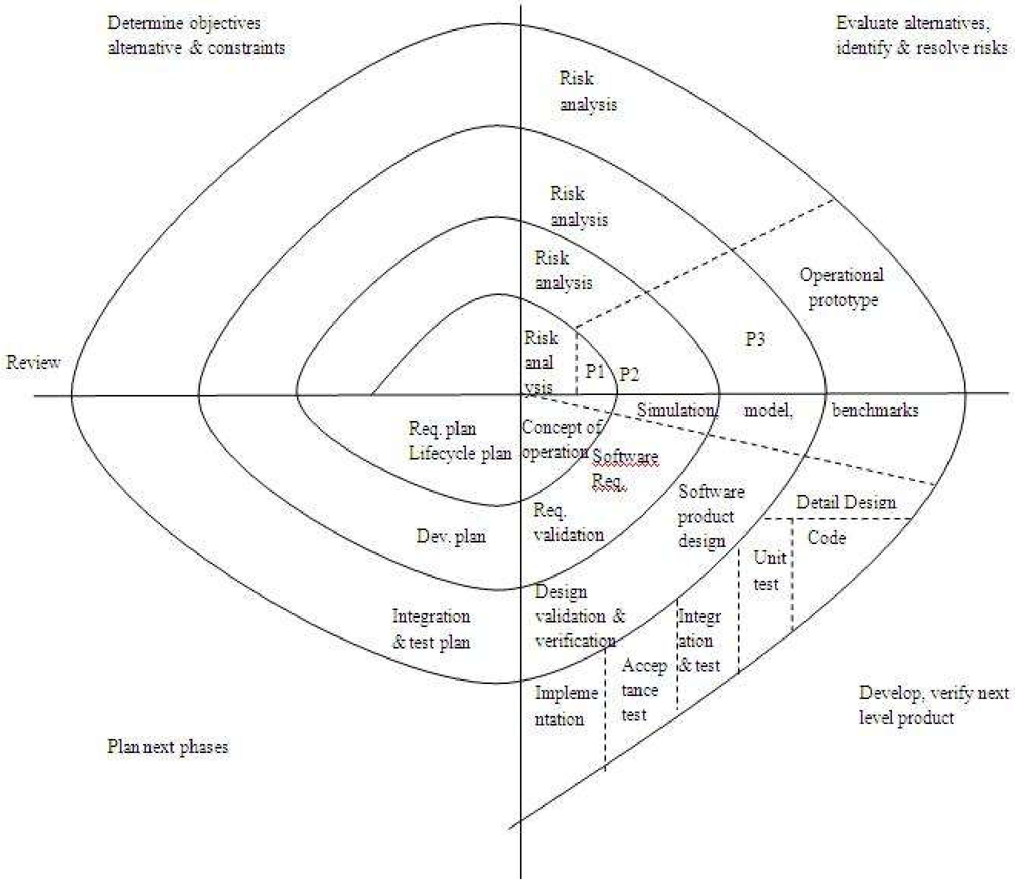

Figure 3: Spiral Model

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.

3.1 CONCEPT OF OPERATION CYCLE

This cycle is to establish the product line approach that the organization wishes to take. During this spiral, the organization focuses on determining the organization's project scope and objectives, operational concept, environmental parameters and assumptions, life-cycle responsibilities, etc. Very large or complex software projects will frequently precede the concept of operation round of the spiral with one or more smaller rounds to establish feasibility and to reduce the range of alternative solutions quickly and inexpensively [4].

3.2 REQUIREMENTS GATHERING CYCLE

Project requirements are gathered during the planning phase. Requirements are validated by generating prototypes that are reviewed. The review feedback of the prototype is used to revise the requirements, and modified prototypes are generated and again presented for second review. The process of obtaining feedback and producing modified prototypes continues until the requirements, scope, and related risks are clearly stated and specified. From this point forward, the phases of coding, testing, and implementing are not fundamentally different from the waterfall approach. The scope of the requirements gathering using the spiral approach generally varies depending on the clarity of the requirements elicited. Multiple cycles may be involved while eliciting requirements, which may extend beyond the time allotted for completion. The spiral model gives more emphasis to risk analysis. In the risk analysis phase, a process is undertaken to identify risk and alternate solutions. When a risk is identified, a prototype is generated to evaluate and handle the risk.

3.3 SPIRAL WITH SQUARE

Incorporating SQUARE with the spiral model involves identifying the cycles of the spiral model where requirements are being handled and then appropriately fitting the various steps of SQUARE into it to identify security requirements early during the life cycle. The number of cycles is determined based on the project size and need. The cycles or spirals where the steps of SQUARE should be carried out are

- concept of operation
- requirements specification

3.3.1 Concept of Operation Cycle

During this cycle, as the feasibility of the project is studied, certain initial steps of SQUARE are also carried out. While determining the project objectives, alternatives, and constraints during this cycle, the requirements team presents the customers with a set of definitions of security terms to discuss and reach agreement on. This facilitates common understanding between the customer and the requirements team about security needs. Once definitions are agreed on, the security goals for the project are identified and documented. Facilitated work sessions, surveys, or interviews may be used to identify the goals, as suggested by the SQUARE methodology. The first two steps are carried out during the first phase (i.e., when the objectives are determined) of the concept of operation cycle.

During the risk analysis phase of the concept of operation cycle, SQUARE Steps 3 and 4 are carried out. The risk analysis phase of spiral emphasizes identifying, evaluating, and resolving risks. Prototypes are generally generated as a method to resolve these risks. Step 4 in SQUARE prescribes this. Before performing this step, artifacts that support the security requirements definition are generated. The output of Step 3 of SQUARE may be scenarios, misuse cases, models, etc. These artifacts act as input for assessing the risks, which is done by a requirements engineer, a risk expert, and the stakeholders.

Once the risks are assessed, the security requirements elicitation technique is selected. This corresponds to Step 5 of SQUARE. This step is carried out during the phase where plans for the next cycle are identified. Elicitation techniques followed for the project may not be suitable for eliciting security requirements. Techniques provided in SQUARE should be followed.

At this point, the concept of operation cycle and SQUARE Steps 1 through 5 are completed. Depending on the complexity of the project, the cycle may be repeated, along with some or all of the SQUARE steps.

3.3.2 Requirements Specification Cycle

Steps 6 through 9 of SQUARE are carried out during this cycle of the spiral model. Functional and other non-functional requirements are elicited for the project, and security requirements are gathered using the technique identified in Step 5. This generates an initial set of requirements, the input for which is the risk assessment results.

Next, the collected requirements are categorized as to level (software, system, etc.) and whether they are real requirements or some other kind of constraint. Work sessions are carried out for this step. At the end of this step, a categorized set of requirements is obtained. Step 7 is carried out during the verification phase of the spiral cycle.

Upon categorizing the requirements, priorities are generated. Step 8 of SQUARE suggests prioritizing the requirements using methods such as Triage and Win-Win. This step is also carried out during the phase where the next level product is developed and verified. The step generates a prioritized requirements list.

To mark the completion of each cycle in spiral, reviews are carried out, as marked on the x axis of Figure 4. Step 9 of SQUARE suggests requirements inspection, which is carried out by the inspection team through inspection methods such as Fagan and peer reviews. This step provides initial selected requirements.

Multiple requirements specification cycles may be carried out until clear understanding of the requirements is achieved and the specification is completely documented. Feedback received at the end of review determines whether another requirements cycle would be required.

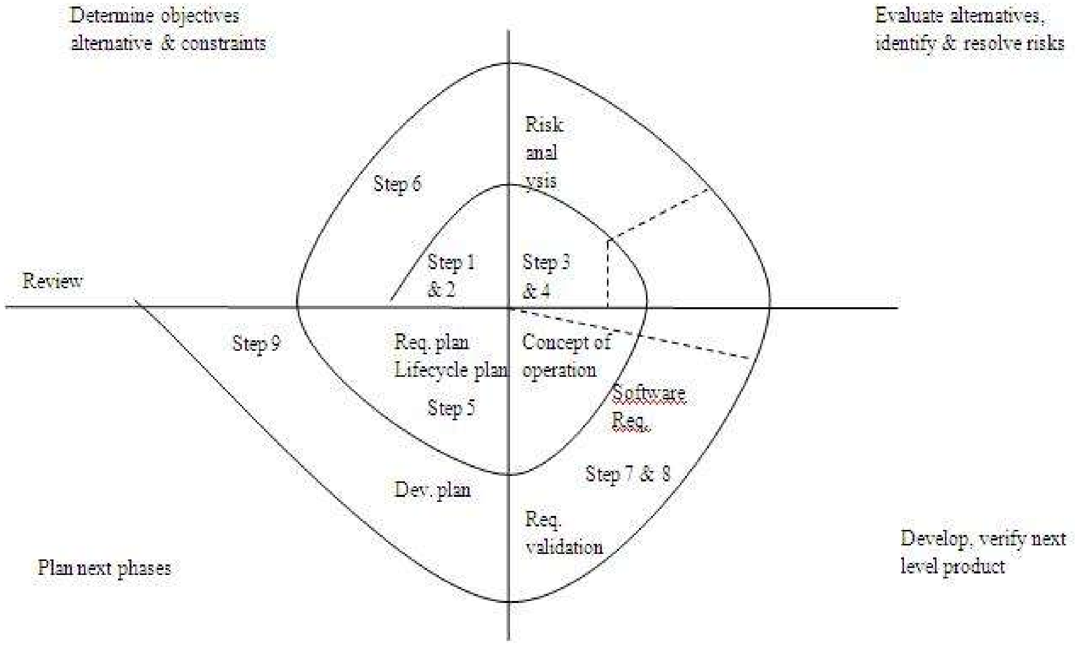

Figure 4:   Spiral with SQUARE

4 Life-Cycle Model: Rational Unified Process

This section deals with Rational Unified Process, a framework for software development. The best practices of RUP, its phases of development, and outcomes of the phases that are necessary for incorporating SQUARE are described.

RUP is a generic process framework that prescribes the following best practices of software engineering [5, 6]:

- Develop software iteratively.
- Manage requirements.
- Use component-based architectures.
- Visually model software.
- Verify software quality.
- Control changes to software.

Rational Unified Process supports risk-driven development [7]. 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 prescribes 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.

Rational Unified Process divides the development cycle into four consecutive phases [6]:

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

The Inception phase primarily involves identification of a business case. A 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. 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 of the Inception and Elaboration phases [6].

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)

4.1 RATIONAL UNIFIED PROCESS WITH SQUARE

Rational Unified Process captures non-functional requirements as a supplementary specification during the Elaboration phase. A project for which security is critical requires more details to be captured early in the life cycle, such as during RUP's Inception phase, to successfully address security concerns. This section describes how SQUARE can be used in RUP's Inception and Elaboration phases to develop security requirements.

4.1.1 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 on. After agreement on the definitions is reached, security goals are 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 necessary for a full understanding of the relevant system are developed, such as the following:

- 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 attacks. After the threats are identified by the risk assessment method, they must be classified according to likelihood. This aids in prioritizing the security requirements that are generated at a later stage. A traceability matrix can also be developed indicating a clear mapping between threats and security requirements. This mapping might result in identifying additional security requirements.

The next step is to elicit requirements. The interaction of the entities with the system are captured and modeled by the requirements team. One of the best practices suggested by RUP is to identify requirements using use cases, as is obvious from the fact that RUP recommends using UML for communicating requirements. The external interfaces interacting with the system are identified and the functional use case is established. Also, an appropriate elicitation technique is chosen and security requirements are gathered 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:	

Outcomes

Along with the outcomes suggested by the Inception phase of RUP, SQUARE results in the following outcomes:

- agreement on definitions
- security goals
- initial risk assessment, including security risk
- initial use case model, including security misuse cases (20–30% complete)

The review process of the first milestone in RUP (life-cycle objective) should include these outcomes.

4.1.2 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 as shown in Table 3 [1].

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, 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 risks are sufficiently mitigated, which will ultimately help to determine the cost and schedule for the completion of development.

Outcomes

Along with the outcomes suggested by the Elaboration phase of RUP, SQUARE results in the following outcomes:

- use case model (functional requirements + security requirements described in misuse cases, 80% complete)
- revised risk list
- categorized and prioritized security requirements

The review process of the second milestone in RUP (life-cycle architecture) should include these outcomes.

Table 4 summarizes integration of the SQUARE methodology into RUP.

Table 4: Summary of RUP with SQUARE

RUP Phase	SQUARE Steps
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 definition (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 techniques (Step 5)** Select the 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. **Requirements inspection (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. **Categorize requirements as to level (system, software, etc.) and whether they are requirements or other kinds 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. **Requirements inspection (Step 9)** The second milestone review process of RUP should include inspection of the security requirements.
Construction	Not Applicable
Transition	Not Applicable

5 Agile Methodologies

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 [8]:

> *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

5.1 AGILE METHODOLOGIES IN USE

There are a number of methodologies that follow agile principles. Scrum [7] provides a project management framework that focuses development into 30-day “Sprint” cycles, the outcome of which is a specified set of “Backlog” features. The core best practice in Scrum is the adoption of daily 15-minute team meetings for coordination and integration.

The Dynamic Systems Development Method [7, 9], which was developed in the U.K., is an extension to rapid application development (RAD) practices. It follows a phase-wise development approach, uses iterative development, and has a strong focus on developing software that meets users’ needs.

Extreme programming [10] focuses on delivering the software the customer needs when it is needed. XP encourages developers to respond to varying customer requirements even late in the development life cycle. Risky projects with dynamic requirements are suitable for XP.

Lean development, feature-driven development, and adaptive software development are other methodologies that follow agile principles.

5.1.1 DSDM Wins

The SQUARE team researched these different agile methodologies and identified the Dynamic System Development Method (DSDM) as the most appropriate one to fit with SQUARE. DSDM was chosen because it is a traditional agile model that has explicit phases into which SQUARE steps can fit appropriately. Methodologies like Scrum and XP that recommend performing activities based only on need are not a good match because to reap the full benefit of SQUARE all of its steps should be followed. Hence a traditional phase-wise development framework is more suitable for SQUARE. The next section describes how DSDM and SQUARE can be combined to address security requirements.

6 Life-Cycle Model: DSDM

DSDM provides a flexible yet controlled process that can be used to deliver new systems. It combines effective use of people's knowledge with tools and techniques such as prototyping to achieve tight project delivery timescales [9]. DSDM is based on the premise that most software projects fail due to people issues rather than technology issues. Hence it directs people to work more effectively to achieve business goals. DSDM is based on the following nine principles:

1. Active user involvement is imperative.
2. DSDM teams must be empowered to make decisions.
3. The focus is on frequent delivery of products.
4. Fitness for business purpose is the essential criterion for acceptance of deliverables.
5. Iterative and incremental development is necessary to converge on an accurate business solution.
6. All changes during development are reversible.
7. Requirements are baselined at a high level.
8. Testing is integrated throughout the life cycle.
9. A collaborative and co-operative approach between all stakeholders is essential.

The DSDM process divides the software development life cycle into seven phases. The pre-project phase ensures that the project is set up properly. Once the project is approved to go ahead, the feasibility study begins. Assessments on whether DSDM is a right approach for the project, likely costs, and technical feasibility are some of the factors evaluated in this phase. After the project is deemed to be feasible, the business study phase commences, which focuses on understanding the business requirements and technical constraints associated with the project. The next phase, functional model iteration, involves the creation of a functional model and prototypes in order to build on requirements identified during the business study. The design and build iteration phase aims to refine and test the functional prototype created during the previous phase. However, both the functional model iteration and the design and build iteration consist of cycles of four activities [9]:

1. Identify what is to be produced.
2. Agree how and when to do it.
3. Create the product.
4. Check that it has been produced correctly (by reviewing documents, demonstrating a prototype, or testing part of the software).

The objectives of the implementation phase are to operate the tested system in the users' working environment and to provide required training to the end user. The final post-project phase ensures effective maintenance of the delivered solution.

6.1 DSDM WITH SQUARE

After completion of DSDM's pre-project phase and feasibility study, SQUARE can be performed starting with the business study. The outcome of the feasibility study would be a feasibility report, an outline plan, and a risk log. The feasibility study should also identify whether the project has criticality in terms of security. This will indicate whether it is appropriate to adopt a method such as SQUARE into the project life cycle. The initial risk log contains only high-level project risks such as budget risk.

The business study and functional model iteration phases are next described in detail, showing how SQUARE can be incorporated into them.

6.1.1 Business Study

The first step of SQUARE, "Agree on definitions," is carried out immediately after scoping the business process to be supported. The stakeholders and the requirements team agree on a set of security definitions pertaining to the project in order to have a common understanding. This set of definitions is included in the business area definition, which is an important work product of this phase.

After identifying the high-level candidate goals and business drivers, the stakeholders and the requirements engineering team determine the high-level security goals. These are added to the business area definition. The requirements team then identifies scenarios, misuse cases, and other artifacts necessary to support security definition. Again, these are documented in the business area definition.

In DSDM, risk management is an ongoing process throughout the project. Hence the risk log is opened at the start of the project. The risk assessment checkpoint is provided at the end of functional model iteration because appropriate risk mitigation measures are feasible only until that point. However, they are captured throughout the project life cycle. Perform risk assessment, SQUARE Step 4, is also done, and the risks are captured in the risk log.

DSDM does not recommend any specific requirements elicitation technique. The team decides which technique is appropriate to the project. The elicitation technique used for functional requirements may not be suitable for the security requirements elicitation. Hence, an appropriate security requirements elicitation technique that suits the client organization and the project as suggested by SQUARE in Step 5 is chosen. For instance, the Accelerated Requirements Method (ARM) has been successful in eliciting security requirements.

After high-level and low-level functional requirements have been gathered, security requirements are elicited using the chosen technique and documented. Hence there is a slight modification in the process because separate techniques for gathering functional and non-functional requirements are not usually followed. The security requirements are refined and categorized based on essential, non-essential, system level, software level, and architectural constraints.

A prioritized requirements list is another key product for this phase. This list should include prioritized security requirements. DSDM recommends prioritizing requirements using MoSCoW rules [11]. Some of the methods used with SQUARE to prioritize security requirements are Win-Win, Triage, and Analytical Hierarchy Process (AHP).

After the security requirements are prioritized, requirements inspection is performed to generate a set of accurate and verifiable security requirements.

All nine steps of SQUARE are performed in the business study phase. The key products that result after applying SQUARE are

1. business area definition
   - agreed-to security definitions
   - security goals document
   - security scenarios and misuse cases to support security requirements definition
   - high- and low-level security requirements along with functional requirements
2. Updated risk log
   - security risks added to high-level project risks
3. Development plan
4. System architecture definition
5. Prioritized requirements list
   - categorized and prioritized security requirements

6.1.2 Function Model Iteration

In the functional model iteration phase, the risk log is revisited to accommodate new risks for the project. If any new security risks are added, SQUARE Steps 6, 7, 8, and 9 are revisited. The prioritized security requirements are updated to reflect changes. In the case of additional security requirements, Steps 7, 8, and 9 are carried out again. This phase acts as a checkpoint to risk assessment because risk mitigation measures are feasible to incorporate only up to this point. Risks identified later on will have to be handled reactively. The same steps can be followed even if requirements change in later phases of the project life cycle.

The following are the key products in this phase:

- functional model and functional prototype
- non-functional requirements list (apart from security requirements)
- functional model review records
- implementation plan
- time box plan
- updated risk log

7 Conclusion

Security requirements are of paramount importance, as they directly reflect the quality of the product. Current processes support capturing requirements like security, but doing so is often an afterthought relative to functional (end user) requirements. Detailed scenarios for quality attributes are not captured during the requirements phase. 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.

The SQUARE team identified commonly used software development methodologies and incorporated SQUARE steps at appropriate phases of their life cycles. This report can help any organization to use SQUARE to elicit security requirements in security-critical projects within the development process they already follow.

References

URLs are valid as of the publication date of this document.

[1] N. R. Mead, E. Hough, and T. Stehney, "Security Quality Requirements Engineering (SQUARE) Methodology, Software Engineering Institute, Carnegie Mellon Univ., Pittsburgh, PA, Tech. Rep. CMU/SEI-2005-TR-009, 2005. http://www.sei.cmu.edu/publications/documents/05.reports/05tr009.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] P. Kruchten, *The Rational Unified Process: An Introduction*, 3rd ed. Boston: Addison-Wesley, 2003.

[6] Rational Software, "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

[7] S. W. Ambler, "A Manager's Introduction to Rational Unified Process," 2005. http://www.ambysoft.com/downloads/managersIntroToRUP.pdf

[8] "Principles behind the Agile Manifesto," 2001. http://agilemanifesto.org/principles.html

[9] DSDM Public Version 4.2. http://www.dsdm.com/version4/2/public/default.asp

[10] "Extreme programming: A gentle introduction," 2006. http://www.extremeprogramming.org

[11] "MoSCoW," Wikipedia. http://en.wikipedia.org/wiki/DSDM#moscow

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. **AGENCY USE ONLY**
(Leave Blank)

2. **REPORT DATE**
May 2008

3. **REPORT TYPE AND DATES COVERED**
Final

4. **TITLE AND SUBTITLE**
Incorporating Security Quality Requirements Engineering (SQUARE) into Standard Life-Cycle Models

5. **FUNDING NUMBERS**
FA8721-05-C-0003

6. **AUTHOR(S)**
Nancy R. Mead, Venkatesh Viswanathan, Deepa Padmanabhan, Anusha Raveendran

7. **PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

8. **PERFORMING ORGANIZATION REPORT NUMBER**
CMU/SEI-2008-TN-006

9. **SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**
HQ ESC/XPK
5 Eglin Street
Hanscom AFB, MA 01731-2116

10. **SPONSORING/MONITORING AGENCY REPORT NUMBER**

11. **SUPPLEMENTARY NOTES**

12A **DISTRIBUTION/AVAILABILITY STATEMENT**
Unclassified/Unlimited, DTIC, NTIS

12B **DISTRIBUTION CODE**

13. **ABSTRACT (MAXIMUM 200 WORDS)**

SQUARE (Security Quality Requirements Engineering) is a method for eliciting and prioritizing security requirements in software development projects. This report describes how SQUARE can be incorporated in standard life-cycle models for security-critical projects. Life-cycle models and process methods considered for the report are the waterfall model, Rational Unified Process, the spiral model, and Dynamic Systems Development Method (an agile method).

This report is for information technology managers and security professionals, management personnel with technical and information security knowledge, and any personnel who manage security-critical projects that follow standard life-cycle models.

14. **SUBJECT TERMS**
SQUARE, requirements engineering, life-cycle models, security requirements, spiral model, waterfall model, RUP, DSDM

15. **NUMBER OF PAGES**
35

16. **PRICE CODE**

17. **SECURITY CLASSIFICATION OF REPORT**
Unclassified

18. **SECURITY CLASSIFICATION OF THIS PAGE**
Unclassified

19. **SECURITY CLASSIFICATION OF ABSTRACT**
Unclassified

20. **LIMITATION OF ABSTRACT**
UL