SQUARE

Security Quality Requirements Engineering

Term paper

Amrinder Singh

July 9$^{\text{th}}$, 2014

Seminar Informatics

Semester: 6

Lecturer: Prof. Dr. Gerd Beuster

INDEX

1. Abstract

This seminar paper will show how security requirements for complex IT projects can be defined and elicited. Security is an important factor for IT systems which stores confidential data. Therefore the right security requirements has to be elicited and implemented afterwards. The SQUARE (Security Quality Requirements Engineering) method is one way, which gives us an overall guideline through the elicitation process. SQUARE is not recommended to use on small-scaled projects, because the process is rather complex and developed for large-scaled projects. This paper provides an overview about this method. The paper closes with a conclusion about SQUARE.

2. Introduction

In this particular time the aspect of security in IT systems becomes very important. Industry and companies relies on confidentiality combined with big data systems. Companies need to make sure that their systems are protected against intruders and attackers. Therefore they need to find out what their security requirements are.

The method SQUARE (Security Quality Requirements Engineering) has been developed to elicit security requirements for IT systems. The method is very strict and executed in nine steps. However it does not describe how to implement the security requirements after they have been elicited. Security requirements can be very diverse hence the methodology describes the elicitation process rather in an abstract way.

3. Security quality requirements engineering

3.1 History

SQUARE have been developed at the Carnegie Mellon University by the Network Systems Survivability (NSS) Program. However it is still in a work-in-progress. Several case-studies with real world clients have shown good results though [SQUARE05, p.13]. The long term goal is a standardized use in the industry and companies for complex projects. Throughout those case studies the methodology has been reworked over time. NSS is also working on a web based tool to support each step of SQUARE.

SQUARE could be used in any kind of project but in fact it was designed for information technology systems.

3.2 SQUARE Process

The SQUARE methodology is very strict and sequential. In 9-steps the needed security requirements can defined, categorized and prioritized. Afterwards the elicited security requirements can be implemented but SQUARE doesn't describe this part furthermore.

The methodology is rather tedious and should be used on large sized IT projects in the early lifecycle.

The following table shows a list of steps which are performed in SQUARE. The output of each step is most likely a document. Some document outputs are also required for further steps.

1.	Agree on definitions
2.	Identify security goals
3.	Develop artifacts to support security requirements definition
4.	Perform risk assessment
5.	Select elicitation techniques
6.	Elicit security requirements
7.	Categorize requirements as to level (system, software etc.) and whether they are requirements or other kinds of constraints
8.	Prioritize requirements
9.	Requirements Inspection

Figure 1 – 9 Steps in SQUARE

3.3 Responsibilities

In each step of SQUARE responsibilities are defined for the members of the project. SQUARE differentiates between the stakeholder- and the requirements engineering-team. As shown in the figure below there are also "Joint" responsibilities. They define tasks and goals which are executed by the stakeholder- and the requirements engineering team.

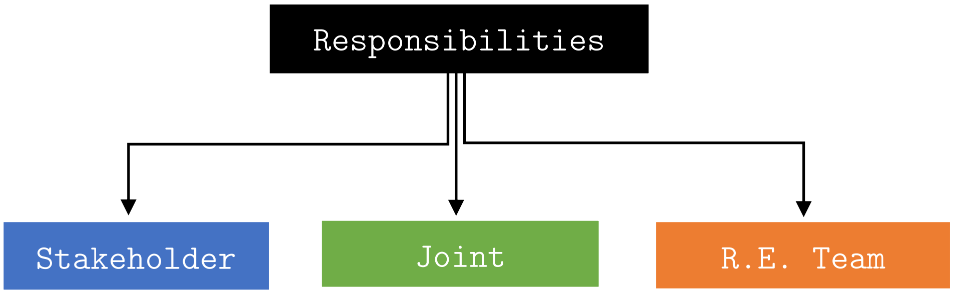


Figure 2 – Responsibilities in SQUARE

Stakeholder:

The stakeholder initiates the project and can be also described as the client, building owner, buyer and so on. His responsibility depends on the current step. In general the stakeholder group take responsibility for superordinated control. For instance they define the project business goals and agree upon suggestions from the requirements engineering team.

Joint:

The joint responsibilities defines tasks which are dedicated the stakeholder group as well as the requirements engineering group.

Requirements Engineering Team:

The requirements engineering team solve issues, which are by technical nature. The specific issue which has to be solved, depends on each task.

3.4 Requirements Engineering

Requirements engineering is a vital component in project management. It describes the process to elicit requirements. Studies have shown that correct requirements engineering can save the economy billions of dollars [SQUARE05, p.17].

“The optimal investment in the security measures is the one that maximizes the security of the application and minimizes both the cost of security measures and of the cost incurred because of security incidents.” [OWASP, p.5].

In many cases the client does not know himself what he actually wants. To solve this problem one has to analyze his the system and the requirements.

Industry can also benefit from security requirements engineering which basically is the same process as requirements engineering, except that it is about eliciting *security* requirements. This is also the reason why SQUARE should always be used in the early project life cycle. However, some hard- and software requirements can only be implemented in an early stage.

4. Security Quality Requirements Engineering Steps

As shown in figure 1 the SQUARE process includes 9 steps. In the following pages each step is going to be explained.

4.1 Agree On Definitions

Before the actual process of eliciting requirements can be started, the client need to make sure to communicate on a same level. Especially in the IT world some definitions are ambivalent. Therefore the requirements engineering - and the stakeholder-team have to agree on definitions they want to use in further steps. There are different ways to execute this step. Most likely it is very dependent on the project circumstances itself. For example: In smaller projects it may be enough to complete surveys, while in larger scaled projects you may have to work with a web-based tool or e-mail surveys.

The following figure shows how the definition of “confidentiality” could be coincided.

confidentiality	The property that information is not available or disclosed to unauthorized individuals, entities, or processes. (i.e., to any unauthorized system entity)	[SANS 03a]
	Ensuring that information is available to only those with authorized access.	[ISO 04]
	Restricting access to information via a hierarchy of classes of access.	[JONES 02]
	Other:	

Figure 3 – Agree on definition example card (source: [SQUARE05])

Responsibilities:

The responsibility of the stakeholder team is, to choose one of the given definition or add a custom one. The requirements engineering team has to support the stakeholder team and provide an initial set of definitions.

4.2 Identify Security Goals

Business Goal The system allows the client to make informed decisions based on which assets are available.
Security Goals G-01 Management shall exercise the effective control over the system‘s configuration and usage. G-02 The confidentiality, accuracy, and integrity of the system‘s data shall be maintained. G-03 The system shall be available for use when needed.

Figure 4 – Business and security goals (source: [SQUARE05])

In this step the formal security goals for the project has to be identified. Afterwards the relevance of the security requirements for the project can be evaluated.

Every company has a different view on how important security is. Also the stakeholders could have different security goals. According to that the stakeholders have to come to a consensus. A common method to define the security goals is brain storming. Once the formal security goals has been identified they must be prioritized.

Responsibilities:

The requirements engineering team is involved with the stakeholder team through the brainstorm process. They discuss about the importance of specific goals. Afterwards the requirements engineering team document and review the result.

4.3 Develop Artifacts

In this step several artifacts have to be collected or generated to support the process of eliciting security requirements. The teams should collect artifacts like:

- use-/Misuse-case scenarios
- system architecture diagrams
- attack trees
- other standardized templates

In some cases systems are not documented at all. Therefore no artifacts can be collected. In this case the requirements engineering team should generate artifacts in first place.

The stakeholder has the responsibility to collect artifacts and present them to the requirements engineering team. If there are artifacts which have to be created in first place the requirements engineering is supposed to do so. The following figure shows how an artifact could like:

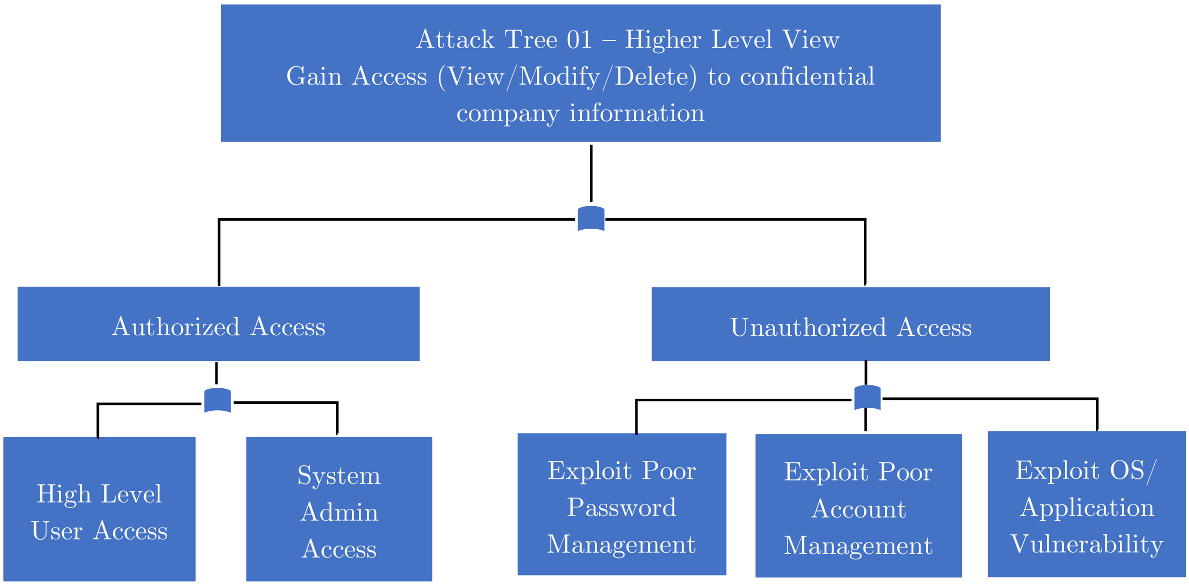

Attack Tree 01 – Higher Level View
Gain Access (View/Modify/Delete) to confidential company information
Authorized Access
Unauthorized Access
High Level User Access
System Admin Access
Exploit Poor Password Management
Exploit Poor Account Management
Exploit OS/ Application Vulnerability

Figure 5 – Example attack tree (source: [SQUARE05])

This example shows an attack tree. The scenario of an authorized and unauthorized access is modeled.

Responsibilities:

The requirements engineering team have to work with the stakeholder and collect or create as many artifacts as possible. They have to verify the accuracy and completeness of all artifacts. As result a set of artifacts for the system is created and used to elicit security requirements.

4.4 Perform Risk Assessment

In this step threats and vulnerabilities have to be identified. There are several risk assessment methods which can be used. Some of the methods may require more skills and effort than the others. Therefore an appropriate method has to be elicited which can be executed by the requirements engineering team. The result of a risk assessment will be useful for eliciting security requirements. A very simple way to perform a risk assessment is to brainstorm threats and categorize them into different tiers. An example is shown below:

	Attackers inside the system	Attackers outside the system
Tier 1	• Insider alters or disables key architecture components.	• Intruder executes malicious code to gain unauthorized access.
Tier 2	• Insider or natural forces physically destroy system components	• Hardware is damaged by natural disaster or environment. • Intruder socially engineers password.
Tier 3	• Terrorist steals system components.	• Hardware fails. • Intruder guesses, crack password

Figure 6 – Example risk assessment (source: [SQUARE05])

This example shows a very simple way of identifying risks. The scenarios have been grouped into three tiers. Each tier represent a specific level of a threat. For example the first tier includes threats which primarily effects the system while the second tier rather contains "hardware damage scenarios".

Responsibilities:

In this step the requirements engineering team have to perform a full risk assessment. In most cases this step is executed by an extern risk expert though. Afterwards results will be shared and reviewed with the stakeholders.

4.5 Select Elicitation Technique

„Select an elicitation technique that is appropriate for the number and expertise of stakeholders, size and scope of the project, and expertise of the requirements engineering team. “

[SQUARE05, p.15]

In the fifth step a technique to elicit security requirements have to be defined. In the IT-world exist several methods which can be used. The challenge in this step is to find the best technique.

4.5.1 Elicitation techniques

Following techniques could be used to elicit requirements:

- Structured/unstructured interviews
- Use/misuse cases [Jacobson 92]
- Facilitated meeting sessions, such as Joint Application Development and the Accelerated
- Soft Systems Methodology [Checkland 89]
- Issue-Based Information Systems [Kunz 70]
- Quality Function Deployment [QFD 05]
- Feature-Oriented Domain Analysis [Kang 90]
- Controlled Requirements Expression [Mullery 79]
- Critical Discourse Analysis [Schiffrin 94]

To decide which one is the best technique, we can compare each other and evaluate them. The following table shows a comparison between each method.

	Misuse Case	SSM	QFD	CORE	IBIS	JAD	ARM
Adaptability	3	1	3	2	2	3	2
Client Acceptance	2	2	2	2	3	2	1
Complexity	2	2	1	2	3	2	2
Implementation Duration	2	2	1	1	2	1	3
Learning Curve	3	1	2	1	3	2	1

Scale: 3 = very good, 2 = fair, 1 = poor.

Figure 7 – Elicitation technique evaluation (source: [SQUARE05])

Based on this result the best method with the most points can be chosen. This way of choose a method is just a simplified example. In fact there are a lot of different methods which were mentioned before, to elicit requirements.

Responsibilities:

The requirements engineering team has to select the best elicit technique and document the decision making process.

4.6 Elicit Security Requirements

The sixth step is all about to execute the method which we choose in the last step. Therefore this step will be different for each project. Nonetheless there are differences between good and bad requirements. A good requirement is unambiguous, measurable and verifiable. For example just compare the following both requirements:

1. „The system shall improve the availability of the existing customer service center. “
2. „The system shall handle at least 300 simultaneous connections to the customer service center. “

The second requirement complies all characteristics which defines a good requirement. In the following chapter we give a short overview about the ARM method, because it is quite a simple and fast way to elicit requirements.

Responsibilities:

Dependent on the elicitation technique the requirements engineering team will support the stakeholder team executing the method. Afterwards the results will be documented.

4.5.2 ARM

The Accelerated Requirements Method (ARM) doesn't earns it name for no reason. It is one of the fastest methods if your purpose is eliciting requirements. In our context we want to elicit security requirements. Therefore ARM helps us and provides a guideline to do so.

The heart of ARM is brainstorming. The stakeholder team is asked to write down seven security requirements in a limited time on a card. Afterwards they have to choose out of those seven, their top three requirements. A security requirement can start with the following sentence:

"An important security requirement of the project is ... ". [SQUARE05, p. 41]

After summarizing the result, duplicates will be deleted. This is how a final set of requirements can look like:

1. The enforcement and usability of an access control system	10. Indelibility (detections and retractions are noted/logged)
2. Security must be manageable and not hinder business	11. Confidentiality (encryption etc.)
3. Information must be kept private from the outside world	12. Partitioned data store (public read only private read/write)
4. Consistent APIs	13. Selectively secure communication
5. Data integrity	14. Represent and support segmented disclosure
6. Strong authentication	15. Role-based restricted views/edit/action access (e.g., summary report info)
7. Reduce/eliminate risks of inappropriate behavior	16. Available 24/7 via remote authenticated access
8. Granular access to data for users (operators) and customers	17. Key action audit (e.g., attribution of who pressed the publish button from where and what changes were made).
9. Accountability (who did what, when, how...)	

Figure 8 – Result set of security requirements

4.7 Categorize Requirements

After one have defined and elicited the security requirements, the categorization have to be done. The purpose of this step is to classify the requirements into different characteristics. After the SQUARE process is completed we can distribute specific requirements better and ensure their implementation. For example some requirements have to be implemented by the hardware department (24/7 availability) and others by the development department (strong authentication). The following table shows one possible example, on how security requirements can be categorized:

	System level	Software level	Architectural constraints
Essential			
Non-essential			

Figure 9 – Security requirements categorization (source: [SQUARE05])

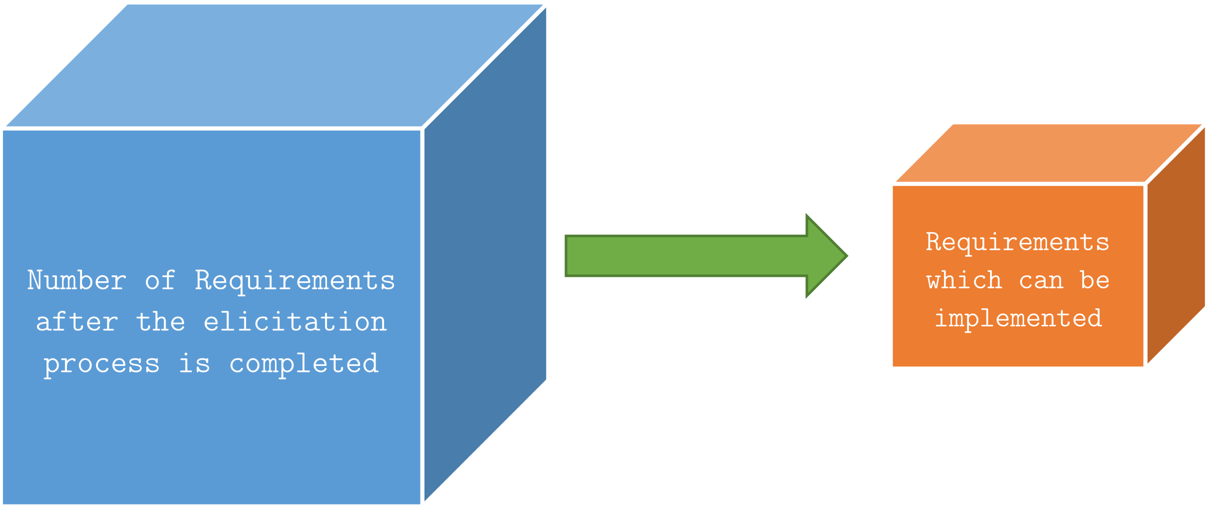
This table shows only one substitutional way on how requirements can be categorized.

The column architectural constraints contains requirements which cannot be implemented due to constraints in the software or hardware design. Once we finished to categorize our requirements the next step also becomes a lot easier.

Responsibilities:

The requirements engineering team provides a guideline to execute the categorizing. They decide to use a specific method. At the end the stakeholder- and requirements engineering team should come to a consensus with the stakeholder team.

4.8 Prioritize Requirements


Number of Requirements after the elicitation process is completed
Requirements which can be implemented

As shown in the figure above we will have a number of requirements which simply cannot be implemented. Reasons for that could be lack of time, resources, a limited project budget, etc. Therefore we need to prioritize our requirements. There are several prioritization techniques that exists. One of them is AHP (analytic hierarchy process) which we will discuss later on.

Responsibilities:

The requirements engineering team will facilitate the prioritization process with the stakeholder team. As decision making base they make use of the risk assessment and categorization results.

4.8.1. ANALYTIC HIERARCHY PROCESS

	SR-1	SR-2	SR-3	SR-4
SR-1	1	5	1/5	3
SR-2	1/5	1	1/5	1/5
SR-3	5	5	1	1
SR-4	1/3	5	1	1

Intensity of value	Interpretation of value
1	Requirements i and j are equal
3	Requirement i has slightly higher value than j
5	Requirement i has strongly higher value than j

Figure 10 – Example Analytic hierarchy (source: [SQUARE05])

The matrix above shows a pairwise comparison between each security requirement. Our measuring factor is the "need value" of a requirement. We can also use the "implementation cost" of a requirement as comparative value. The analytic hierarchy process (AHP) has found a very high client acceptance. Also case-studies with real world clients have delivered successful results.

AHP is a simple but accurate method which can be used in the context of prioritizing requirements. In the industry are several different methods which can be used. Because of the simplicity an example of AHP has been chosen.

4.9 Requirements Inspection

The purpose of this step to verify our requirements. This inspection can be done at different levels of formality. The main goal is to find defect requirements which are inconsistent, unambiguous or mistaken. The following figure shows an example how we can log our results.

SNO	DATE	ORIGIN	DEFECT TYPE	DESCRIPTION	SEVERITY	OWNER	REVIEW	STATUS

Figure 11 – Example requirements inspection

This table is just one simplified example which includes the most important information of a requirement. Of course there are a lot more ways and methods for documenting results, which can be used as alternative.

Responsibilities:

The requirement engineering team should help the stakeholder team to execute the inspection technique. The stakeholder team have to make sure that every requirement is verifiable, in scope with financial means, and able to implement. This is the last opportunity for the stakeholder to remove certain requirements.

Now the SQUARE process is completed and we have created a set of requirements which are prioritized, categorized and viable to implement. The next step is the implementation of our requirements but SQUARE does not give a guideline on how to proceed here.

5. Conclusion

SQUARE is one out of many methods to elicit security requirements. Certainly it is reasonable to make use of such a technique as SQUARE. But the project has to be complex enough if SQUARE also shall be used in a useful context.

Though SQUARE is a very static und document-based method it could be very useful to give unexperienced users an overall guideline, on how to elicit the right requirements.

Whether SQUARE is useful dependents completely on the project itself. SQUARE can be sophisticated and expensive but in large IT project the outcome might be worth it.

6. List of figures

7. Literature

[SQUARE05]	Nancy R. Mead, Eric D. Though, Theodore R. Stehney II: Security Quality Requirements Engineering (SQUARE) Methodology, November 2005, Carnegie Mellon University, USA
[OWASP]	OWASP Application Security Guide for CISOs https://www.owasp.org/images/c/c2/OWASP_Application_Security_Guide_for_CISO.pdf
[Sans]	The SANS Institute. SANS Glossary of Terms Used in Security and Intrusion Detection. http://www.sans.org/resources/glossary.php (2005).
[Jones]	Jones, Capers, ed. Tutorial: Programming Productivity: Issues for the Eighties, 2nd Ed. Los Angeles, CA: IEEE Computer Society Press, 1986.
[Jacobson]	Jacobson, Ivar. Object-Oriented Software Engineering: A Use Case Driven Approach. Boston, MA: Addison-Wesley, 1992.
[Checkland]	Checkland, Peter. Soft Systems Methodology. Rational Analysis for a Problematic World. New York, NY: John Wiley & Sons, 1989.
[Kunz]	Kunz, Werner & Rittel, Horst. "Issues as Elements of Information Systems." http://www-iurd.ced.berkeley.edu/pub/WP-131.pdf (1970).
[QFD]	QFD Institute. Frequently Asked Questions About QFD. http://www.qfdi.org/what_is_qfd/faqs_about_qfd.htm (2005).
[Kang]	Kang, K.; Cohen, S.; Hess, J.; Novak, W.; & Peterson, A. Feature-Oriented Domain Analysis (FODA) Feasibility Study (CMU/SEI-90-TR-021, ADA235785). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1990.

	http://www.sei.cmu.edu/publications/documents/90.reports/90.tr.021.html.
[Mullery]	Mullery, G. P. “CORE: A Method for Controlled Requirements Specification.” Proceedings of the 4th International Conference on Software Engineering. Los Alamitos, CA: IEEE Computer Society Press, 1979.
[Schiffrin]	Schiffrin, D. Approaches to Discourse. Oxford, England: Blackwell, 1994.