UNIVERSITY OF TARTU

FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

Institute of Computer Science

Software Engineering

Wajid Ali Khilji

Evaluation Framework for Software Security Requirements Engineering Tools

Master's Thesis (30 ECTS)

Supervisor: Dr. Raimundas Matulevičius

Tartu 2014

ABSTRACT

Evaluation Framework for Software Security Requirements Engineering Tools

In software development requirements are considered as building blocks of software system, which also are considered to be responsible in event of failure. Bad requirements can lead to software features that are not to the specifications. For that reason requirement gathering process is considered as the most sensitive and complicated process among all software engineering lifecycle processes. In current age where cyber-attacks are common security requirements also comes into place and plays a very important role in software development process. In order to elicit security requirements new type of tools are begin to form a shape called security engineering tools which help in eliciting security requirements. That considered being the most efficient way of eliciting security requirements. Moreover these tools empower users with artifacts specifically to cater security needs, which save time and efforts for engineers in return. Nevertheless these tools are still at their infantry and are lacking mass adoption by software security engineers. Reason because these tools have steep learning curve which can add-up to development time and end up pushing more cost to the project. In order to decide which tool to select for a particular project require engineers to use these tools which in return will consume tremendous amount of time. Moreover using unstructured tool selection process can also leads to wrong tool selection which will be the waste of time and efforts. In this research work we are going to construct structured approach which will help engineers in security engineering tool selection process. In order to aid this process analysts and architects will be able to rate the features they want the most in a particular security engineering tool. In return from this process they will be able to choose between security engineering tools and select the best one. Finally using approach constructed in this research work will save time, efforts, and costs. In our approach we will analyze security engineering processes, methods and tools, to construct a framework that will help aid engineers in security engineering tool evaluation process.

Key Words: Security, requirements, engineering, tools

Hindamisraamistik tarkvara turvalisusnõuete tööriistade jaoks

Tarkvaraarenduses on nõuded kui süsteemi vundament, mis vastutavad ka ebaõnnestumiste eest. Valed nõuded võivad viia tarkvara eripäradeni, mis tegelikult ei vasta spetsifikatsioonidele. Sel põhjusel peetakse nõuete koostamist kõige keerulisemaks ja olulisemaks sammuks tarkvaraarenduse elutsükli kõikide protsesside jooksul. Tänapäeval, kus küberrünnakud on tavalised, mängivad turvalisuse nõuded väga olulist rolli tarkvaraarenduse protsessis. On levimas uut tüüpi tööriistad, mille kasutamist peetakse kõige efektiivsemaks meetodiks turvalisusnõuete väljatöötamisel. Lisaks võimaldavad need tööriistad lahendada turvalisusega seotud küsimusi kasutajal endal, hoides märgatavalt kokku inseneride aega. Siiski on nende tööriistade areng alles algstaadiumis ning neid ei ole tarkvarainseneride poolt massiliselt kasutusele võetud.

Põhjus on väga pikas uue tarkvara õppimise ja sellega kohanemise protsessis, mis põhjustab ajakadu arendusprotsessis ning lisab projektile kulusid. Projekti jaoks konkreetse tööriista valimisel võib tutvumine ja katsetamine võtta inseneridel hulgaliselt aega. Lisaks sellele võib struktureerimata valikuprotsess viia vale tööriista kasutuselevõtmisele, mis raiskab omakorda kõigi aega ja pingutusi. Selles uurimuses kavatseme me koostada struktureeritud lähenemise, mis aitab insenere turvalisusnõuete tööriistade valimisel. Protsessile kaasaaitamiseks saavad analüütikud ja arhitektid hinnata tarkvara omadusi, mida nad enda seisukohast olulisimateks peavad. Sellest lähtuvalt saavad nad valida kindlate tööriistade vahel ning teha parima valiku. Antud uurimustöös kontstrueeritud lähenemisega on võimalik säästa aega, vaeva ja kulutusi. Uurimuse koostamise käigus uurime me tarkvaraarenduse turvaprotsesse, meetodeid ja tööriistu ning püüame luua raamistikku, mis oleks inseneridele turvalisusnõuete tööriistade hindamisel abiks

Võtmesõnad:

Turvalisus, nõuded, masinaehitus, töövahendid

Table of Contents

List of Figures

List of Tables

List of Abbreviations

1

Introduction

Software requirements are the foundation of any software system. As such similar to the building and construction requires a base, requirements play same utmost importance in developing software systems. Unlike developers of 80's now a day's software architects are more aware of this phenomenon that, success of any software project relies on complete, concrete and concise requirements. One of the major factors contributing to the requirements is security engineering. In current age of cyber warfare, discussing about requirements without discussing security constraints may lead to loss of valuable assets like information, methodology, and business workflow. In some critical cases may lead to loss of human life, such of the examples can be learnt from the fact that software systems are now part of human society. Power, healthcare, education, governments, military and telecommunications almost all social sectors are now becoming part of a giant cloud (internet).

These elements take software architects and analysts attention to the methods, tools, and languages available for security requirements which can help illustrate the software system in terms of possible vulnerabilities. However available tools for security requirements are already on their infantry. And there is no customary procedure that provides standard framework support for deciding which security requirements elicitation tool to choose. In this research we will analyze some processes and tools to construct a framework that will support and accelerate SRE tool decision making process.

1.1. Motivation

"The role of security requirements is to provide information about the actual needs of a system or application with respect to security in order to accomplish its business goals" (Braz, et al., 2008). Common agenda of eliciting security requirements in early stages of lifecycle is to reduce increasing costs for the later stages. Although fixing bug from a developed software system will be more costly, than avoiding vulnerabilities at the beginning of development lifecycle. So the relevant question to ask at this point is: how to reduce these vulnerabilities? There could be several solutions towards addressing vulnerabilities, but in most cases assessing vulnerabilities could be a difficult task, and keeping track of risks and threats could also be a challenging task.

This can be achieved by using requirements engineering tools, but the fact that these are software requirements specific and does not provide security requirements artifacts can lead to missing or inappropriate security requirements. The appropriate solution for eliciting security requirements will be to use security requirements elicitation tools.

Security engineering tools are the software tools that help accelerate security requirements elicitation process. One similar definition in this context is of requirements engineering tools as discussed: “Requirements engineering (RE) tools are software tools which provide automated assistance during the RE process and support the RE activities” (Matulevičius, 2005). Similar to RE tools SE tools also provide the functionality of documenting, validating, and analyzing the security requirements.

One of the benefit of using SE tools is it reduces the possibility of eliciting unclear or ambiguous security requirements. Moreover the use of RE tool for eliciting security requirements is not that inferior as well. While SE tools were not readily available a decade ago, analysts and architects were forced to follow traditional approach in eliciting security requirements, i.e. use of RE tools. However not using standard SE tool can create possibility of capturing unwanted, unclear, or ambiguous security requirement, which in result will leave vulnerabilities into the software system.

1.2. Scope

Among several security engineering tools, deciding on which tool to choose can be time consuming and efforts adding process. Also all the available tools use variant approaches that can add to the learning curve as well. In this research work we will construct a method that will address the issue of choosing security requirements engineering tool. We keep our scope limited to analyzing security assurance processes (discussed in chapter 3) and security engineering tools (discussed in chapter 4). First will be to analyze the security assurance processes that will give us our core requirements, these will be used to test the security engineering tools, and provide us with the means or functionality that has been fulfilled from a particular requirement. On the basis of means we will construct a security requirements engineering tools selection framework.

1.3. Research Problem

One aspect of the security requirement is they address the needs of software in terms of security, and try to provide as much solutions as possible to cater security needs. The common way of investigating security is to define assets, roles, threats, vulnerabilities, risks, treatment and mitigations in a particular system. That in result will deliver unstructured and variant security requirements that can lead to misunderstanding and difficulties while implementation. And the quality of security requirements will not be of high standards. Moreover security requirements have the same changing nature of normal requirements which makes it complicated to jot down security requirements in concrete form.

As a consequence the solution for this problem is to follow a structured approach that in result will create high quality security requirements. Security engineering tools deliver a way to improve security requirement by providing several artifacts to address security related problems, and limit the user with security related artifacts. The function of these tools is to follow standards and procedures to construct security needs. These security needs finally will create standardized security requirements of high quality. There are several security engineering tools available, as the security needs are increasing day by day; these tools are becoming mature to capture requirements beyond common imagination of human mind. However these tools are still lacking the mass adoption because of acceptable maturity level. And which tool to choose from may take time and efforts. In order to achieve tool selection process we have to define research problem mentioned below.

- ***Research Problem***

  How to evaluate security requirements engineering tools?

In this research work we will analyze the SE processes and tools that will provide us with the requirements for security requirements engineering tools selection process? Our focus will be limited to popular tools and processes because of limited amount of time available for this research work. Before analyzing SE tools it is important to ask what should be the characteristics of SE tools. In order to simplify the problem, we divide it into two questions.

- ***Research Question1***

  What are the requirements when selecting security engineering tools that offer better means to support security requirement process and maintain high quality security requirements?

In order to achieve verified results it is important that we evaluate the outcomes:

- ***Research Question 2***

  What are the means that fulfill security engineering tool requirements that provide improved security requirements artifact?

That gives us our two questions that will be discussed and resolved in this thesis. These questions are derived from research problem and will be addressed resolving actual problem. However we only have divided this into two questions for simplicity.

1.4. Structure of Work

This thesis is structured into chapters, mentioned in Table 1-1 are the number, name and description of the chapter. The initial two chapters are for introducing the research work, and applications are in chapter 3 and 4. The findings can be visited in chapter 5, and results in chapter 6. Finally chapter 7 is for conclusion.

Table 1-1 Thesis Structure

CH: ID	Chapter Name	Description
1	Introduction	Introducing the Thesis, research question
2	Security engineering and security requirements elicitation	Security engineering definition, requirements elicitation techniques, and information security standards.
3	Security development lifecycles	Software security assurance processes.
4	Security engineering methods & tools	Security engineering methodologies, analysis of tools.
5	Framework for security requirements engineering tools	Security requirements, framework construction, and use of evaluation framework.
6	Tool Assessment	How this method was evaluated comparing one additional method and the results were given.
7	Conclusion	The conclusion and future work, related to this thesis work.

In this thesis our approach will be straight forward, flowing with the tools and SSAP's, which is quite a challenging task because of various differences in both. For the sake of simplicity we will eliminate some sub-processes in SSAP's and take only the once that are relevant to our research.

In this chapter we will introduce some of the common definitions that are used in this research work. This will give us a basic understanding of the terms used, and also will be a kick-starter for the upcoming chapters. And finally we will define the expected outcomes of this research.

- ***Framework Definition***

In research work of Matulevičius about process support for requirements engineering, the framework definition is given as follows: The purpose of the frameworks is to provide a skeleton structure for the RE- tool evaluation and comparison (Matulevičius, 2005). A framework also works as a template guiding on how to proceed with the current process.

- ***Process Definition***

The process is defined as "A systematic series of actions directed to some end" (Dictionary.com, 2014). This describes the process as a series of (tasks, events and activities) that combined together gives a complete set of process that will lead to a single outcome. Term process is commonly used while defining the lifecycle of a software development, as software lifecycle is mainly composed of series of different correlated or individual processes.

- ***Lifecycle Definition***

Lifecycle is defined as “A series of stages through which something (as an individual, culture, or manufactured product) passes during its lifetime” (Merriam-Webster, 2014). While the word itself is taken from the evaluation of lifecycle, but also this has more resemblance to software lifecycle. Because software engineering is not only about developing software, it also includes security constraint, support and maintenance which can last for many years. Moreover the word lifecycle is used in other wide range of fields like (enterprise, product development, software release and more). Lifecycle can also be consist of several sub-sections delivering artifacts, or sub-objects.

- ***Artifact Definition***

Artifact is defined as “An object made by a human being, typically an item of cultural or historical interest” by (Oxford Dictionaries, 2014). While artifact in this research is referred to the sub-elements like (activities, tasks or events) involved during the lifecycle of the software development.

1.5. Contribution

In this research work our main focus is to analyze security engineering process and tools that will be used to form a framework that will help in choosing between SRE tools, i.e. what is the best available, well suited and well developed SRE tool? In chapter 5 we will discuss about the tools and processes that will help us in producing SRE tool selection method based on available functionality, and that will be our basis to introduce a framework. This thesis will produce some features, characteristics and validated requirements for SRE tools.

- ***Features***

There will be two main features in this research work: 1. Analysis of security engineering lifecycles. 2. Requirements elicitation from SRE tools. As mentioned the motive will be to collect these requirements as precise as possible to avoid variation between processes and SRE tools. The elicited requirements from processes will then be merged and tested against the SRE tools.

- ***Characteristic***

We will develop a Framework that will help leverage the SRE tool selection processes. Moreover there are several frameworks addressing in different aspects of software engineering field. This framework will be different in sense that this will be addressed to security aspects of the software development.

- ***Requirements validated for SRE-tool***

Finally after achieving the goal “having a SRE tool selection framework” at place we can define validation method. Questions like how to validate tools using constructed framework will be addressed in chapter 6 of this thesis.

2

Security Engineering and Security Requirements Elicitation

Security engineering is a field of software engineering, which includes safety, security, vulnerabilities and their treatment mechanisms. In a whole it is a big field to deal with, because more and more security vulnerabilities are on their way since the invention of internet and cloud based software. Modeling security requirements mainly tends our focus towards Use-cases. As mentioned by (Sindre, et al., 2005) "Use cases have become popular for determining, communicating, specifying, and documenting requirements". In this research they also mention that most of the stakeholders are comfortable with descriptions of operational action sequences than declarative specifications.

There are also problems with use-case-based approaches to requirements engineering, says (Sindre, et al., 2005), such as over simplified assumptions about the problem domain and premature design decisions. But with slight modification use-cases can provide functionality of security requirements. As though there are several methods and approaches to elicit security requirements. Our task will be to demonstrate, what these methods are and how these help engineers elicit the security requirements in this chapter.

Talking about the security requirements elicitation, we will introduce security engineering along with security engineering analysis framework depicted in Figure 2-1, then we will discuss about requirements elicitation techniques as our main goal in this chapter will be to cover as much about the elicitation techniques most preferably adapted and used in the software industry. Most of the requirements in this research will be elicited by analyzing the SSAP's and SRE tools. Where main method of gathering requirements consists of brainstorming and reading the SSAP's and tools itself. Moreover when it comes to requirements gathering there are 10 most common approaches defined in (Mochal, 2008). Also in this chapter we have introduced security engineering approaches, mentioned in Table 2-1, and information security standards, which resolves different problem than our focus approach depicted in Figure 2-2.

2.1. Security

One of the well-known authors of the book security engineering Ross Anderson describes the security as: “Security engineering is about building systems to remain dependable in the face of malice, error, or mischance” (Anderson, 2001).

The software should be dependable in the face of Malice: Desire to inflict injury, harm, or suffering on another, either because of a hostile impulse or out of deep-seated meanness (Dictionary.com, 2014). The software should be dependable in the face of Error: a deviation from accuracy or correctness; a mistake, as in action or speech (Dictionary.com, 2014). The software should be dependable in the face of Mischance: a mishap or misfortune (Dictionary.com, 2014).

Security engineering is a discipline which requires expertise from many different domains i.e. software cryptography and hardware temper-resistance. Moreover it involves the tools, processes, and methods (Anderson, 2001). According to Anderson a good security requires four things to come together: Policy, Incentives, Mechanism and Assurance. See the framework diagram depicted in Figure 2-1.

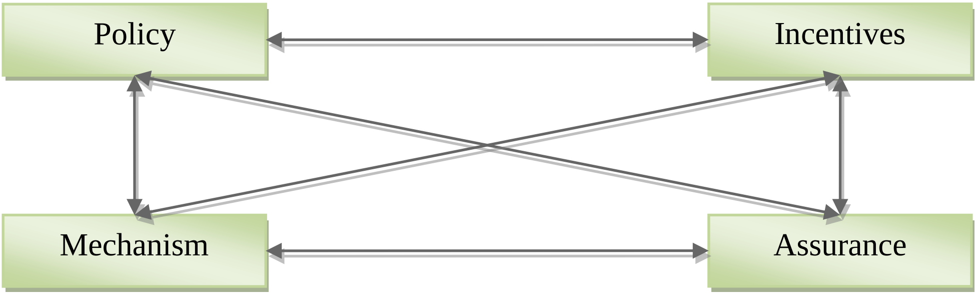


Figure 2-1 Security Engineering Analysis Framework > Adapted from (Anderson, 2001)

Security engineering analysis framework is an abstract declaration of security engineering process, which provides all necessary attributes associated with the security engineering. Almost all companies have policy defined where security constraints are declared. Either physical tangible assets or non-tangible information related objects, in this framework both can fit very well. However our priority is information security that comes with the software security. For example: In almost all companies security policy is defined in some sense. And applying a security mechanism will provide with an incentive i.e. secure software. And finally assurance will give satisfaction of achievement of the objective security policy.

Some more definitions of security are:

- Software security is the idea of engineering software so that it continues to function correctly under malicious attack (McGraw, 2003).
- Security engineering is the field of engineering dealing with the security and integrity of real-world systems (Science Daily, 2014).

2.2. Requirements and Security Requirements

Most of the authors agree on the requirement definition that it is the specification of stakeholders needs. IEEE defines requirement as (Institute of Electrical and Electronic Engineers, 1990).

1. A condition or capability needed by a user to solve a problem or achieve an objective.
2. A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document.
3. A document representation of a condition or capability as in definition 1 or 2.

Our motive is to make a clear distinction between requirements and security requirements. As both comes under one heading, and has the same objective of “demonstrating specifications”. They are lot more different in sense that security requirements in it can have large amount documentations stating the software specifications on security perspective. After defining few more requirements definition from the different authors we will discuss about security requirements.

“Requirements definition includes, but is not limited to, the problem analysis that yields a functional specification. It is much more than that. Requirements definition must encompass everything necessary to lay the ground work for subsequent stages in system development” (T. Ross, et al., 1977).

One of the definitions of software requirement specification is: a software requirement specification is a comprehensive description of the intended purpose and environment for software under development (TechTarget, 2014).

In software requirements, most commonly three types of requirements are categorized in general (Sommerville, 2004).

- ***Functional Requirements:***
  “Statements of services the system should provide how the system should react to particular inputs and how the system should behave in particular situations.” (Sommerville, 2004).
- ***Non-Functional Requirements:***
  “Constraints on the services or functions offered by the system such as timing constraints, constrains on the developments process, standards, etc.” (Sommerville, 2004).
- ***Domain Requirements:***
  “Requirements that come from the application domain of the system and that reflect characteristics of that domain.” (Sommerville, 2004).

Now let’s define security requirements, “security requirements have traditionally been considered to be” non-functional requirements” or “quality” requirements. Like other quality requirements (e.g., performance, usability, cost to run), they do not have simple yes/no satisfaction criteria. Instead, one must somehow determine whether a quality requirement has been satisfied”.

Like a requirements expert who try to elicit the requirements via questioning from the customer and user. A security requirements expert may gather security requirements via questioning the attackers and malicious users’ perspective. Getting from the fact that security requirements don’t always have yes/no answer to the problem, these however in a whole are extended to overall lifecycle of the software development, e.g. from requirements gathering to response.

2.3. Security Engineering Approaches

Nicolas Mayer has defined four types of approaches that are most common among the security engineering institutes, Security Oriented, Risk-based, Requirement Engineering, and Model-based approach (Mayer, 2009). Different security engineering methodologies however follow different approach in sense that their focus varies. While all of these methodologies follow Requirements engineering approach and varies in other aspects. See Table 2-1 that shows the methodology and approach adapted by the security engineering methods.

Table 2-1 Summary of software and security engineering state of the art > Adapted from (Mayer, 2009)

Software and Security Engineering	**References**	**Security Oriented**	**Risk-based approach**	**RE approach**	**Model-based approach**
	Firesmith	++	+	++	-
	Haley *et al*. and Moffet and Nuseibeh	++	+	++	-
	DITSCAP automation framework	++	++	++	-
	SQUARE	++	++	++	-
	KAOS extended to security	++	-	++	++
	Misuse cases	++	-	++	++
	Abuse cases	++	-	++	++
	Mal-activity diagrams	++	-	++	++
	Abuse frames	++	-	++	++
	Secure Tropos	++	-	++	++
	Tropos Goal-Risk framework	-	++	++	++

Legend:

++: Completely covered and at the core of the document

+: Partially covered or not playing a central role

-: Not covered

2.4. Information Security Standards

In thesis work of Nicolas Mayer (Mayer, 2009) the overview of "ISO/IEC 2700X series of standards" is provided. Figure 2-2 is showing standard ISO/IEC 2700X series at the core of the diagram, from where other standards are derived. These standards are focused on risk management (RM). These are also known as ISMS information security management system standards. "The ISO standards providing requirements and guidance about best management practices are part of the most well-known standards. The most popular management system series of standards are the ISO 900X series about quality management systems" (Mayer, 2009). In his work Mayer mentions several standards, i.e. international organization of standardization/international electro-technical commission (ISO/IEC), technical specification (ISO/TS) and international workshop agreement (IWA) standards. From these our main focus is ISO/IEC 2700X series which as mentioned is dealing with information security.

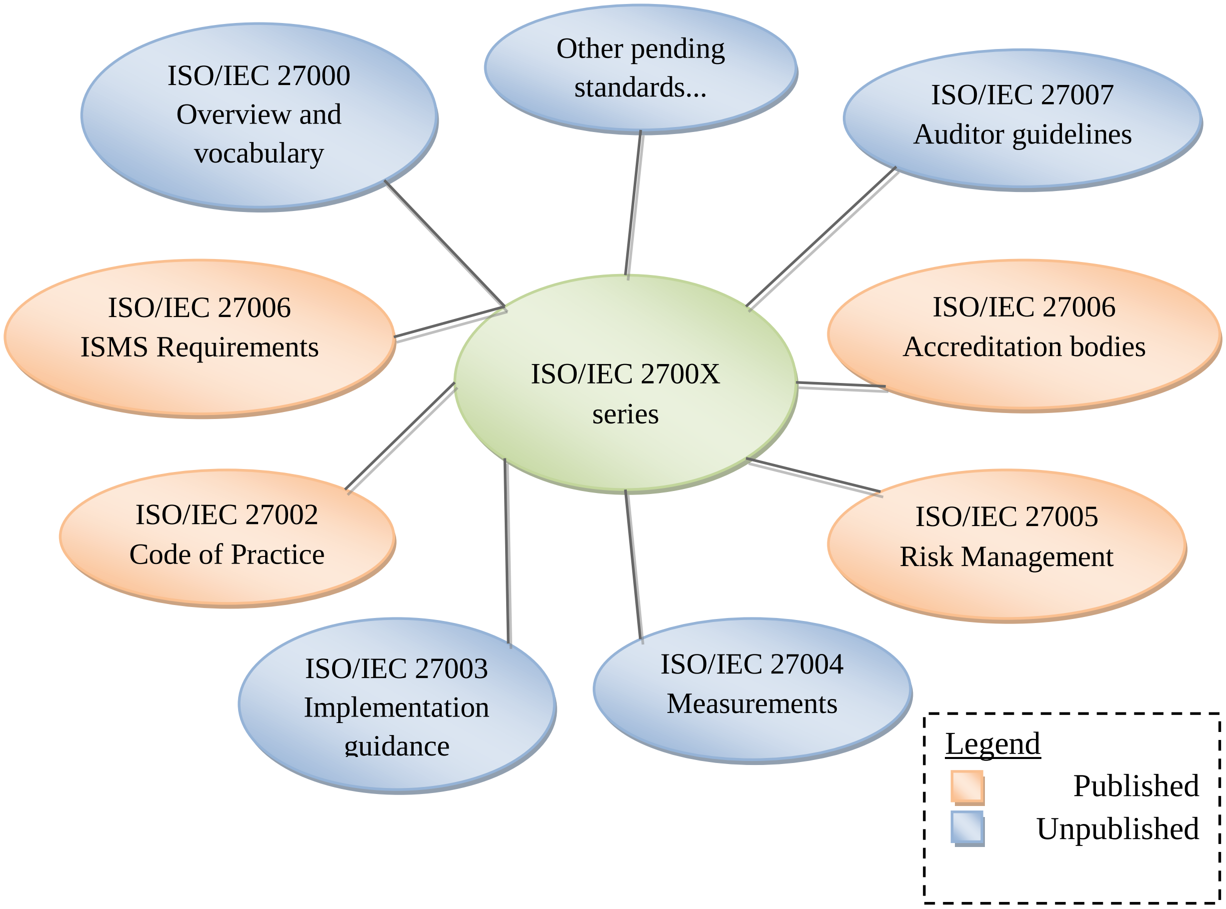

ISO/IEC 27000 Overview and vocabulary
Other pending standards...
ISO/IEC 27007 Auditor guidelines
ISO/IEC 27006 ISMS Requirements
ISO/IEC 2700X series
ISO/IEC 27006 Accreditation bodies
ISO/IEC 27002 Code of Practice
ISO/IEC 27005 Risk Management
ISO/IEC 27003 Implementation guidance
ISO/IEC 27004 Measurements
Legend
Published
Unpublished

Figure 2-2 The ISO/IEC 2700X series of standards > Adapted from (Mayer, 2009)

"An ISMS is a systematic approach to managing sensitive company information so that it remains secure. It includes people, processes and IT systems by applying a risk management process" (ISO, 2014). Security standards however are there for assuring the information security, unlike security engineering processes which are addressed towards producing secure software.

2.5. Chapter Summary

In this chapter we have introduced security engineering definition, which is an abstract to our topic, requirements and security requirements definition, to give reader understanding of security requirements elicitation approaches. Finally we have introduced with information security management standards (ISMS) to illustrate that ISMS resolves different problem adopting different approach than software security assurance processes (discussed in later chapters).

Coming up next we will introduce SSAP's and some additional definitions that will help us in understanding the chapter context. Our task will be to elicit requirements from SSAP's i.e. what a particular SSAP require in order fulfill the criteria of being able to develop secure software. This will help us in testing the security engineering tools.

3

Security Development Lifecycles

The need for stable and non-redundant Security development lifecycle is crucial, because “almost every software controlled system faces threats from potential adversaries”, as mentioned in (T. Devanbu, et al., 2000). Almost all of the software security assurance processes agree that software security is inbuilt feature in the software through whole lifecycle of the software development process. That’s why it is essential to mention that security can also be more difficult to implement in the later software development process or in already developed software.

As a matter of fact, requirements are the major part of any software system, and chances are these requirements will evolve in later development process. To register these requirements in early stage of lifecycle model, engineers try to surface with the solutions. In most of the software development processes requirements part is initiated in the beginning of the lifecycle, however each development phase has its own levels and standards to follow. In order to enable security into the software in stages security engineers at Microsoft have developed optimization model shown in Figure 3-2 where phases can be seen divided into types known as (Basic, Standardized, Advanced and Dynamic). The emphasis of this chapter will be to analyze the available well developed security engineering processes that focus on developing secure software. It will lead our focus to primary outcome from this chapter “derived requirements” by analyzing the core requirements and implementation from security development lifecycles.

Notice the fact that security requirements are non-functional requirements. It is worthy to mention that it increases development costs, and more, its time consuming as well. Placing security concern in early stages of software development lifecycle will reduce development costs (Microsoft., 2014).

To start with we introduce the general lifecycle model depicted in Figure 3-1 which will be the basis for software security assurance processes (SSAP), than we will introduce optimization model which is a good example on showing how security can be divided into levels depicted in Figure 3-2. Next we will go through all available or commonly used SSAP’s, and try to elicit requirements as precise as possible. And finally we will try to merge these elicited requirements and form a single table mentioned in Table 5-1. These elicited requirements will be used to analyze the SRE tools that are discussed in later chapter. Moreover there will be an example on how software security is enabled into the software under heading 3.3.

3.1. Stages of Software Development Lifecycle (SDLC)

SDLC is commonly referred to software development lifecycle that consist of four lifecycle stages suggested in General Lifecycle Model (Ragunath, et al., 2010). In Figure 3-1 the major steps involved in lifecycle are depicted along with one addition of release process. This framework is often taken as framework for introducing any software development lifecycle process (including SSAP Software security assurance processes) discussed later in this chapter.

Introducing additional process of release is vital here because all SSAP's include response plan which is needed during response from the field. In SSAP's as mentioned all sub-processes have separate security requirements, in case of release there will be a response plan e.g. how to address critical threat after release.

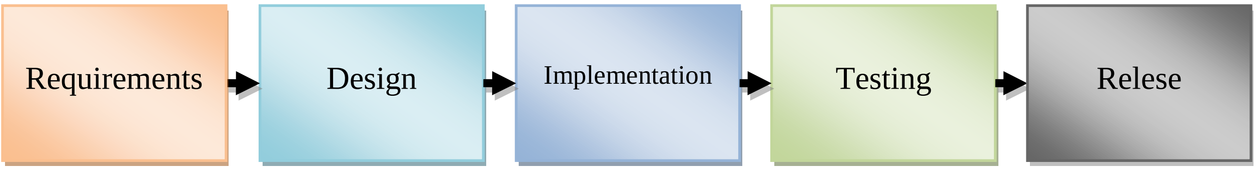


Figure 3-1 General Life Cycle Model > Adapted from (Ragunath, et al., 2010)

3.2. Optimization Model

Figure 3-2 is showing the levels of SDLC that has been sub grouped under (Basic, Standardized, Advanced and dynamic) it focuses on development process improvement, contrast to other development models. It also suggests that process improvement is one of the key features of security development lifecycle, discussed under next heading (Microsoft.., 2010).

3.3. Example on Security Engineering Process

One of the examples mentioned can be seen at (Microsoft., 2014), where three years old services written in C++ with 11,000 LOC were using unauthenticated access in database-driven web product. After gaining knowledge about threat modeling (See Table 4-1 Requirements from SDL), two team members uncover vulnerability in sensitive data. "One developer elects to address the possible SQL injection vulnerabilities identified by the threat modeling". He uses stored procedure in places where they were not used, modifies the access rights and also removes the interactive user's permissions for deleting database objects.

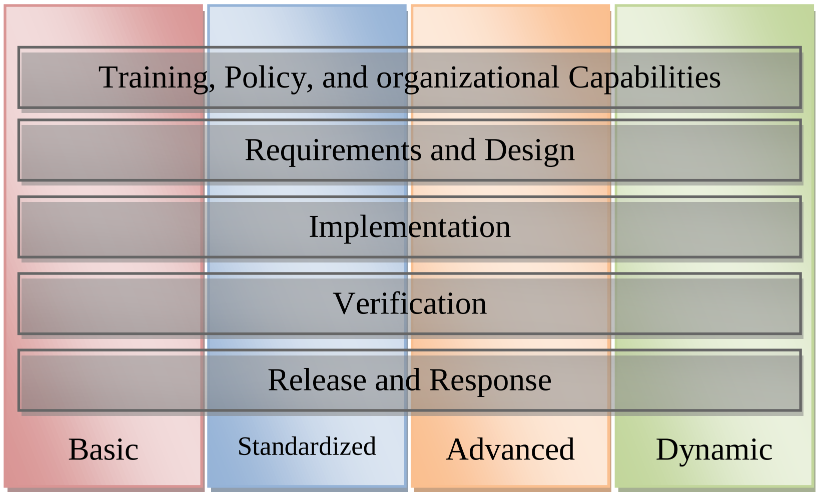


Figure 3-2 SDL Optimization Model with Capability and Maturity Levels > Adapted from (Microsoft, 2010)

Example above gives the understanding that team was working on legacy system and according to (Microsoft.., 2010) “Integration of secure development concepts into an existing development process can be intimidating and costly if done improperly”. So the most important part for them was to uncover some of the critical vulnerabilities. To do so they started reading “Threat modeling” in book “The Security Development Lifecycle”, and try to come up with the most appropriate solution, in this case fixing the access rights and addressing the issues with SQL injection.

3.4. Security Engineering Assurance Processes

Software Security Assurance Processes (SSAP) are for supporting secure software development, further more they facilitate software security at the core of the process evaluation which in turn construct a software skeleton for developers. There are three security engineering methods discussed in this thesis:

- SDL (Howard, et al., 2006)
- 7 Touch points (Addison-Wesley Software Security Series, 2006)
- OWASP (OWASP, 2014).

3.4.1. Security Development Lifecycle (SDL)

Microsoft security development lifecycle (SDL) is a security assurance process that is focused on secure software development (Howard, et al., 2006). It is based on traditional SDLC, in addition to this, SDL consist of further added processes for security. Training and (Response & Release) are now the important parts of the secure software development. All processes in this lifecycle have been broken down under sub-processes (knows as artifacts in our research).

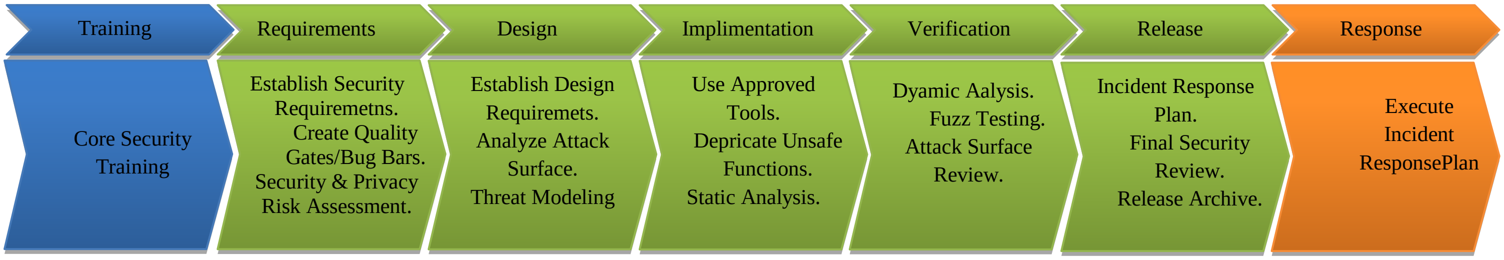


Figure 3-3 Security Development Lifecycle > Adapted from (Microsoft.., 2010)

Figure 3-3 shows the processes along with artifacts, and from these artifacts we will try to extract requirements that will be used as a base for testing the SRE tools. Table 3-1 is constructed in a way that it gives phase-by-phase artifact heading following a requirement ID. Requirements column is derived by analyzing the artifact description, i.e. what requirements can be gathered in a particular artifact, and what techniques can be used. The requirements in Table 3-1 are numbered in a way that requirement SRS004 mentioned has artifact (Perform security and privacy risk assessment).

Table 3-1 Requirements from SDL

SDL REQ ID	Artifact	Requirements
PH1: Training		
SRS001	Core Training	Tool should provide basic training material that will help understand the implementation of secure software.
PH2: Requirements		
SRS002	Establish security and privacy requirements	Tool should provide the means of establishing security and privacy requirements, defining minimum criteria for security and privacy for an application, and provide work item tracking system.
SRS003	Create quality gates/bug bars	Tool should provide bug severity threshold, with ratings e.g. critical, important.
SRS004	Perform security and privacy risk assessment	Tool should be able to provide means of examining software design based on cost and regulatory requirements.
PH3: Design		
SRS005	Establish design requirements	Tool should provide means of validating design specification against functional specification, i.e. accurate and complete design specification, and minimal cryptographic design requirements.
SRS007	Perform attack surface analysis	Tool should provide through analysis of overall attack surface, i.e. defining system privileges and employing layered defenses.

SRS008	Use threat modeling	Tool should be able to provide structured approach to threat scenarios, i.e. identification of security vulnerabilities, and determining risks from these threats, and establishing appropriate mitigation.
PH4: Implementation		
SRS009	Use approved tools	Tool should provide list of approved tools and associated security checks (such as compiler options and warning).
SRS010	Deprecate unsafe functions	Tool should be able to provide project functions API's
SRS011	Perform static analysis	Tool should provide security code review analysis policy.
PH5: Verification		
SRS012	Perform dynamic analysis	Tool should be able to provide security code verification functionality, i.e. run-time
SRS013	Perform fuzz testing	Tool should be able to provide testing policy, i.e. deliberate program failure by introducing malformed random data.
SRS014	Conduct attack surface review	Tool should provide means of attack surface review.
PH6: Release		
SRS015	Create incident response plan	Tool should provide means of response plan to address new threats.
SRS016	Conduct final security review	Tool should provide means of reviewing all security activities performed during lifecycle.
SRS017	Certify release and archive	Tool should provide means of ensuring privacy and security requirements were meat, and archiving of data essential to post release servicing tasks.
PH7: Response		
SRS018	Execute incident response plan	Tool should be able to provide implementation of incident response plan,

3.4.2. Software Security (7 Touch points)

7 Touch points STP is the security development process that focuses on set of best practices. (Addison-Wesley Software Security Series, 2006) Seven touch points provide seven features improvements in regards to the software security. This security assurance process is also follows the traditional approach of general development model depicted in Figure 3-1 along with additional processes, Test plans and feedback from the field.

In Figure 3-4 seven processes are depicted, and unlike SDL they have artifacts mentioned above these processes. In this research work we will take these artifacts and try to gather the requirements as we did for SDL previously. These requirements can be seen in Table 3-2 where in requirements column requirements are defined as precise as possible.

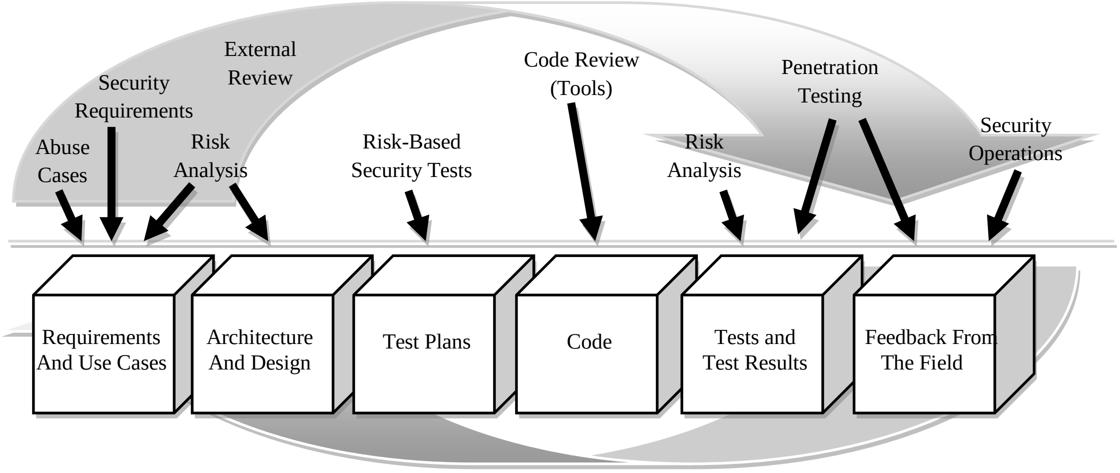


Figure 3-4 Software Security 7 Touch Points > Adapted from (Addison-Wesley Software Security Series, 2006)

Table 3-2 Requirements from 7 Touch Points

STP REQ ID	Artifact	Requirement
SRT001	Code review tools	Tool should provide the ability of static and dynamic code review
SRT002	Risk analysis	Tool should provide means of analyzing the associated risks with the development software
SRT003	Penetration testing	Tool should provide means of conducting penetration testing (Vulnerability scanning)
SRT004	Risk based security tests	Tool should provide means of producing test runs at individual unit level.
SRT005	Abuse cases	Tool should provide functionality of drawing abuse cases (similar to misuse cases)
SRT006	Security requirements	Tool should provide list of all security activities performed during lifecycle.
SRT007	Security operations	Tool should provide means of reviewing all security operations associated with product and company

3.4.3. OWASP CLASP

One last in SSAP’s list is OWASP, which is an abbreviation for Open Web Application Security Project. It is an open source project developed by software security community (Open software security community, 2014). It also provides several artifacts under which general life cycle model framework shown in Figure 3-1 considered the bases in development of OWASP. There is no particular diagram available to depict SSAP itself; however the CLASP views can be seen in Figure 3-5. That separates the CLASP views in different appropriate perspectives for developing secure software. Table 3-3 is showing the elicited requirements from OWASP SSAP, where the columns are divided as, requirements ID, next artifacts definition, and finally requirements.

As OWASP provides open source platform so we can analyze extensive features of an SSAP. In this section we have introduced the requirements shown in Table 3-3 OWASP process in Figure 3-5 and OWASP top 10 security risks in Table 3-4.

Table 3-3 Requirements from OWASP

OCL REQ ID	Artifact	Requirement
SRC001	Institute Awareness Program	Tool should provide basic training material and instructions that will help understand the implementation of secure software.
SRC002	Perform Application Assessments	Tool should provide means of analyzing security requirements and design, security test and Source level security review
SRC003	Capture Security Requirements	Tool should provide building misuse cases, security policy, attack surface and trust boundaries illustration
SRC004	Implement Secure Development Practices	Tool should provide guide on how to annotate classes with security properties, secure design, resources, contracts and interfaces.
SRC005	Build Vulnerability Remediation Procedures	Tool should provide guide on how to address reported security issues and security issue disclosure process.
SRC006	Define and Monitor Metrics	Tool should provide means of creating metrics, in order to evaluate results,
SRC007	Publish Operational Security Guidelines	Build operational security guide, specify database security configuration.

- The Comprehensive, Lightweight, Application Security Process (CLASP)

  “The CLASP provides a well-organized and structured approach for moving security concerns into early stages of the software development lifecycle, whenever possible” (OWASP, 2014). There are five high level perspectives called CLASP views, these views broken down into activities which contain process components. To understand CLASP process Figure 3-5 illustrates the inner working of this process, as from View > Activity > Process component.

  CLAPS views are categorized in a way that engineers can see through different perspective from each view. First there is a concept view to get through the basics of the process. There is role-based view to help understanding the authentication management. Activity-assessment view to understand the costs, applicability, and risk of inaction. Activity-implementation view to analyze the security related activities. And finally there is a vulnerability view to state risks, problems, consequences etc.

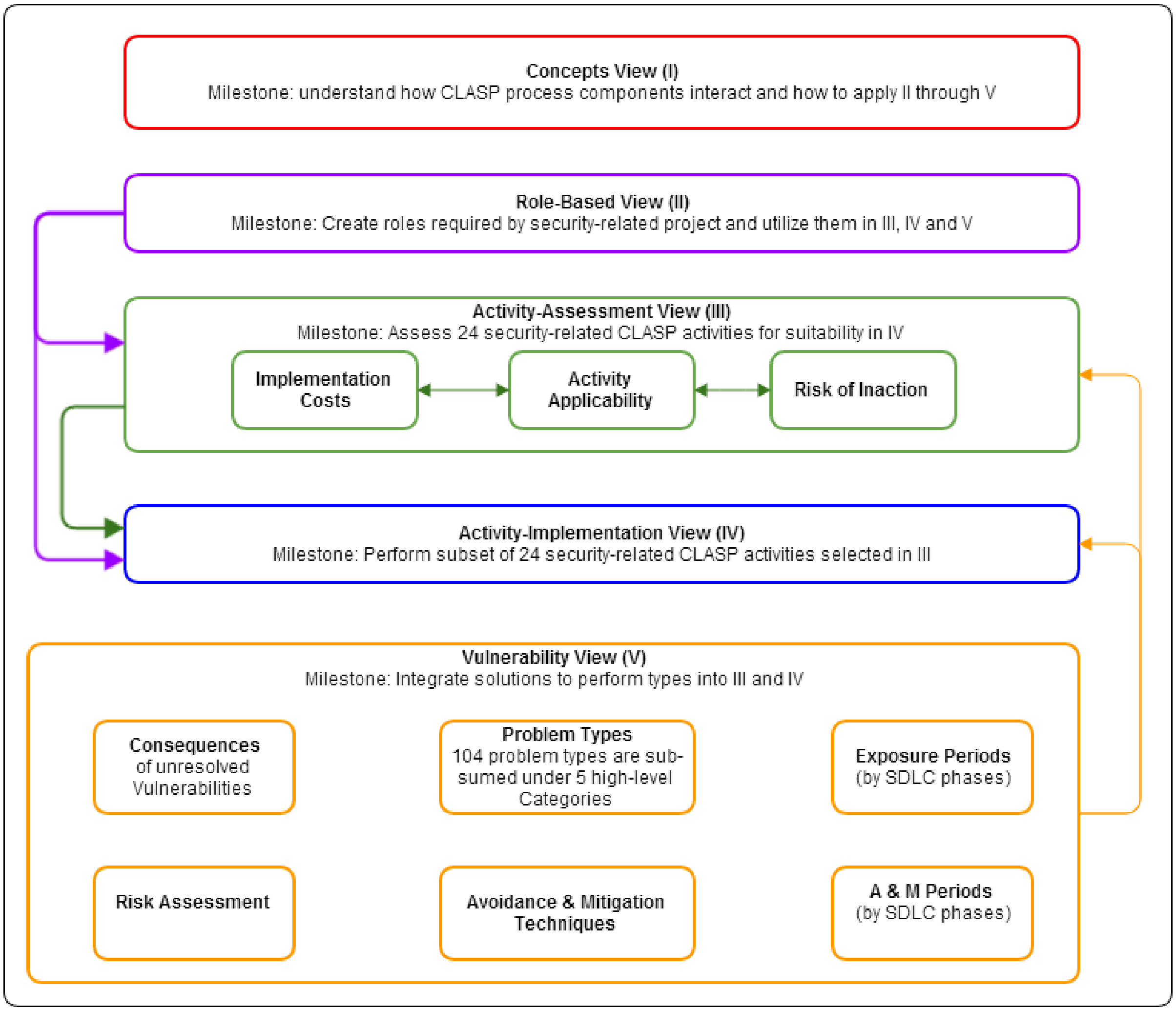

Concepts View (I)
Milestone: understand how CLASP process components interact and how to apply II through V
Role-Based View (II)
Milestone: Create roles required by security-related project and utilize them in III, IV and V
Activity-Assessment View (III)
Milestone: Assess 24 security-related CLASP activities for suitability in IV
Implementation Costs
Activity Applicability
Risk of Inaction
Activity-Implementation View (IV)
Milestone: Perform subset of 24 security-related CLASP activities selected in III
Vulnerability View (V)
Milestone: Integrate solutions to perform types into III and IV
Consequences of unresolved Vulnerabilities
Problem Types 104 problem types are sub-sumed under 5 high-level Categories
Exposure Periods (by SDLC phases)
Risk Assessment
Avoidance & Mitigation Techniques
A & M Periods (by SDLC phases)

Figure 3-5 CLASP Views and their interactions > Adapted from

- Anti-Requirements in OWASP perspective

An anti-requirement is a requirement of a malicious user that subverts an existing requirement (Crook, et al., 2002). An anti-requirement is mainly created by a malicious user; however it can be created by security requirement engineer by the use of misuse/abuse cases or other modes of requirements elicitation tools. In Table 3-4 OWASP top 10 security risks are mentioned, these can be taken as the most updated security risks because OWASP community prints “OWASP top 10” article on yearly basis.

Table 3-4 OWASP Top 10 > Adapted from (OWASP, 2013)

	Security Risk	Description
Security Risks	Injection	Injection flaws, such as SQL, OS, and LDAP injection occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.
	Broken Authentication and Session Management	Application functions related to authentication and session management are often not implemented correctly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities.
	Cross-Site Scripting (XSS)	XSS flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation or escaping. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.
	Insecure Direct object references	A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, or database key. Without an access control check or other protection, attackers can manipulate these references to access unauthorized data.
	Security Misconfiguration	Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server, and platform. Secure settings should be defined, implemented, and maintained, as defaults are often insecure. Additionally, software should be kept up to date.
	Sensitive Data Exposure	Many web applications do not properly protect sensitive data, such as credit cards, tax IDs, and authentication credentials. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data deserves extra protection such as encryption at rest or in transit, as well as special precautions when exchanged with the browser.
	Missing Function level Access Control	Most web applications verify function level access rights before making that functionality visible in the UI. However, applications need to perform the same access control checks on the server when each function is accessed. If requests are not verified, attackers will be able to forge requests in order to access functionality without proper authorization.
	Cross-Site Request Forgery (CSRF)	A CSRF attack forces a logged-on victim's browser to send a forged HTTP request, including the victim's session cookie and any other automatically included authentication information, to a vulnerable web application. This allows the attacker to force the victim's browser to generate requests the vulnerable application thinks are legitimate requests from the victim.
	Using Components with known vulnerabilities	Components, such as libraries, frameworks, and other software modules, almost always run with full privileges. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications using components with known vulnerabilities may undermine application defenses and enable a range of possible attacks and impacts.
	Un-validated Redirects and forwards	Web applications frequently redirect and forward users to other pages and websites, and use untrusted data to determine the destination pages. Without proper validation, attackers can redirect victims to phishing or malware sites, or use forwards to access unauthorized pages.

3.5. Chapter Summary

In this chapter we have introduced SSAP's and elicit requirements that will be testing against tools in the next chapter. We have discussed aspects of software development lifecycle, and how secure software development processes are related to general lifecycle model. We fulfilled the task of eliciting requirements from SSAP's which in return gave us seven requirements mentioned in Table 5-1 to test on SRE tools.

Coming up next security engineering tools description and their evaluation, we will introduce the methodologies which have been adapted by SRE tools. After describing the methodologies we will start testing the tools against requirements depicted in Table 5-1.

4

Security Engineering Methods and Tools

Security engineering tools are well known for describing threats, vulnerabilities, risks and mitigations most of which is done by defining the model via use-case, misuse-case, and abuse-case diagrams. With time and seeing the desperate need towards security, requirements gathering has been evolved to elicit security requirements using similar approach but with variant artifact. These artifacts are defined and used in software security engineering tools. In this chapter we will introduce CORAS methodology (CORAS, 2014) which is depicted in Figure 4-1, SQUARE methodology (Mead, et al., 2005) can be visited in Table 4-1, and TROPOS methodology (Mouratidis, et al., 2008) in Table 4-2. Next we will introduce the security engineering tools that have adapted these methodologies, and test our proposed requirements from previous chapter against tools.

Apart from use-case, misuse-case and abuse-cases there are several other techniques available in eliciting software security requirements, some of which are:

1. Security Risk oriented BPMN (Altuhhova, et al., 2013)
2. Secure UML (Basin, et al., 2009)
3. UML SEC (Jürjens, 2001)
4. Misuse Cases (Sindre, et al., 2005)
5. Mal-Activity Diagram (Sindre, 2006)

However each tool that we are going to test in this chapter adapts different approach in eliciting security requirements, and their approach is mostly based on unified modeling languages.

4.1. Security Engineering Methods

In security engineering methods we have chosen CORAS, SQUARE, and TROPOS because these methods have been adapted by well-developed software security requirements engineering tools, named CORAS, SQUARE, and SecTro2 tools. We will introduce the methodologies first and then we will analyze the tools one by one.

4.1.1. CORAS

CORAS is a method for conducting security risk analysis, it provides a customized language for threat and risk modeling and comes with detailed guidelines explaining how the language should be used to capture and model relevant information during the various stages of the security analysis. (CORAS, 2014). CORAS also includes with the tool that follows CORAS method depicted in Figure 4-1.

There are eight steps involved in CORAS method that are considered as steps to conduct security risk analysis. Figure 4-1 represents basis for this methodology that includes asset, threat, risk, and treatment diagrams. (CORAS, 2014). While analyzing the tool we will be able to see the CORAS method in action under heading security engineering tools analysis.

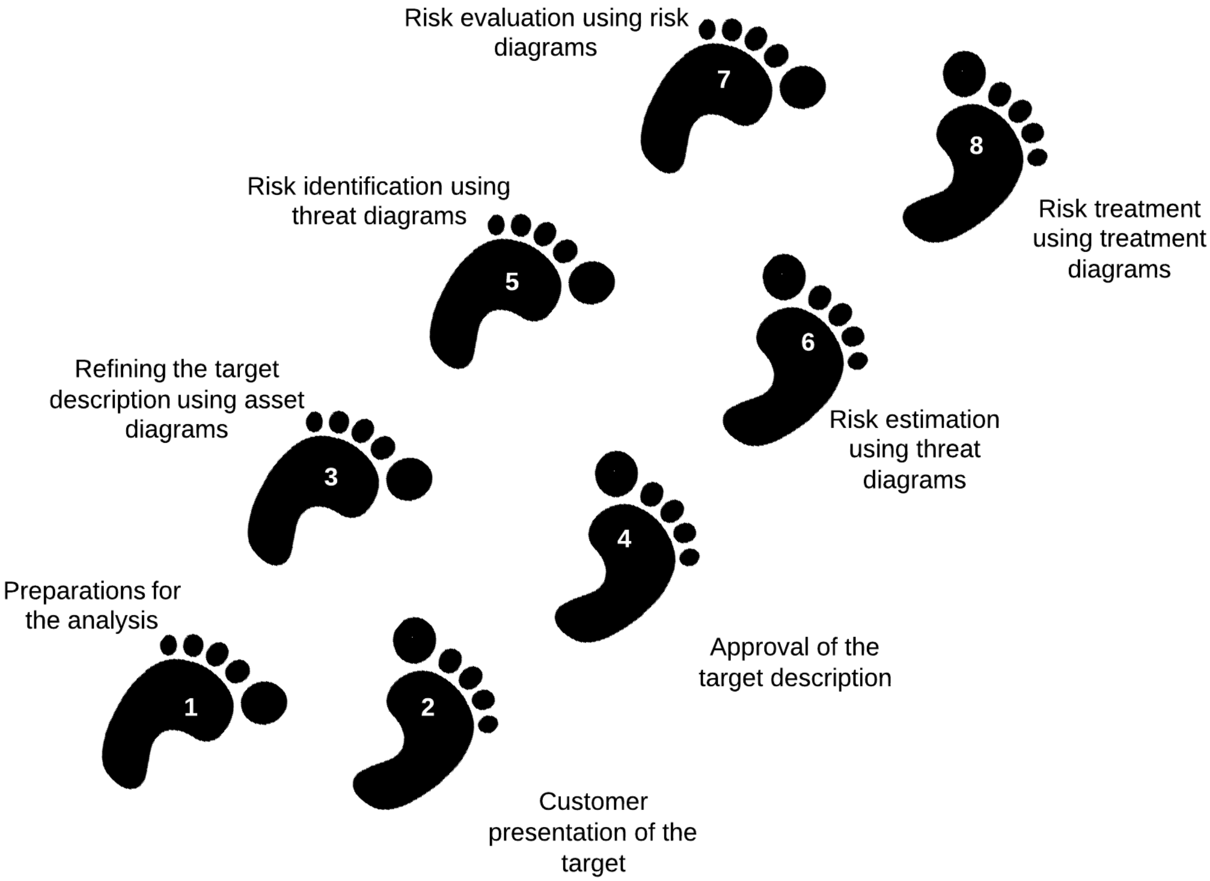

Risk evaluation using risk diagrams
7
8
Risk identification using threat diagrams
Risk treatment using treatment diagrams
5
6
Refining the target description using asset diagrams
Risk estimation using threat diagrams
3
4
Preparations for the analysis
Approval of the target description
1
2
Customer presentation of the target

Figure 4-1 CORAS > Adapted from (CORAS, 2014)

4.1.2. SQUARE

The SQUARE is the short form for Security Quality Requirements Engineering. It is the process that provides means for eliciting, categorizing, and prioritizing security requirements for information technology systems and applications (Mead, et al., 2005). Announced four years ago SQUARE tool is also available to download which we will be analyzed in this chapter.

In SQUARE method there are nine discrete steps that surrounds over all method. Table 4-1 is divided into five columns which gives an overview of the inner workings of the SQUARE method. Each step identifies the inputs, major participants, suggested techniques and output, where output from each step severs as an input for the next step.

Table 4-1 Steps in SQUARE Process > Adapted from (Mead, et al., 2005)

	Step #	Step	Input	Techniques	Participants	Output
SQUARE STEPS	1	Agree on definitions	Candidate definitions from IEEE and other standards	Structured interviews, focus group	Stakeholders, requirements team	Agreed-to definitions
	2	Identify security goals	Definitions, candidate goals, business drivers, policies and procedures, examples	Facilitated work session, surveys interviews	Stakeholders, requirements engineer	Goals
	3	Develop artifacts to support security requirements definition	Potential artifacts (e.g., scenarios, misuse cases, templates, forms)	Work session	Requirements engineer	Needed artifacts: scenarios, misuse cases, models, templates, forms
	4	Perform risk assessment	Misuse cases, scenarios, security goals	Risk assessment method, analysis of anticipated risk against organizational risk tolerance, including threat analysis	Requirements engineer, risk expert, stakeholders	Risk assessment results
	5	Select elicitation techniques	Goals, definitions, candidate techniques, expertise of stakeholders, organizational style, culture, level of security needed, cost benefit analysis,	Work session	Requirements engineer	Selected elicitation techniques

			etc.			
	6	Elicit security requirements	Artifacts, risk assessment results, selected techniques	Joint application development (JAD), interviews, surveys, model-based analysis, checklists, lists of reusable requirements types, document reviews	Stakeholders facilitated requirements engineer	Initial cut at security requirement
	7	Categorize requirements as to level (system, software, etc.) and whether they are requirements or other kinds of constraints	Initial requirements, architecture	Work session using a standard set of categories	Requirements engineer, other specialists as needed	Categorized requirements
	8	Prioritize requirements	Categorized requirements and risk assessment results	Prioritization methods such as Triage, Win-Win	Stakeholders facilitated by requirements engineer	Prioritized requirements
	9	Requirements inspection	Prioritized requirements, candidate formal inspection technique	Inspection method such as Fagan, peer reviews	Inspection team	Initial selected requirements, documentation of decision making process and rationale

4.1.3. Secure TROPOS

In TROPOS methodology there are five main development phases: Early requirements, late requirements, architectural design, detailed design and implementation (Bresciani, et al., 2004). The major difference mentioned in this research paper is the notion of early requirements, as most of the developers can work well with later four phases. In Table 4-2 available phases of Tropos can be seen.

TROPOS also comes with certain stages that are required in the secure software development lifecycle. There are 6 stages in TROPS methodology (Mouratidis, et al., 2008) which states similar stages as of secure software development lifecycle. The main artifacts discussed in

TROPOS are Actor, Goal, Plan, Resource, Dependency, Capability and Belief, which are used to model the requirements in security domain.

Table 4-2 TROPOS Phases > Adapted from (Mouratidis, et al., 2008)

	Stage ID	TROPS Stage
TROPOS PHASES	1	Context and Asset Identification
	2	Security objective determination
	3	Risk analysis and assessment
	4	Risk treatment
	5	Security requirement definition
	6	Control selection and implementation

4.2. Security Engineering Tools

Security engineering tools are for aiding the requirements gathering process for secure software development at smooth pace. These tools are mostly based on unified modeling language (UML) however they do follow various approaches to address the same problem of eliciting security requirements. As we have already chosen methodologies CORAS, SQUARE, and TROPOS we will analyze the tools that have adapt these three methodologies.

1. CORAS (CORAS methodology) (CORAS, 2014)
2. SQUARE (SQUARE methodology) (Mead, et al., 2005)
3. SecTro2 (TROPOS methodology) (Mouratidis, et al., 2008)

4.2.1. Capabilities of CORAS Tool

CORAS is UML based security requirements gathering tool, which emphasis on generating risk and threat scenarios with the help of associated diagrams. It provides from basic diagram artifacts to advance artifacts. CORAS tool includes various artifacts to help understand the security constraint and requirements. In order to well organize the elicited security requirements CORAS tool separate these artifacts into categories i.e. connections, basic CORAS, high level CORAS, dependent CORAS, and legal CORAS. In appendix-A these artifacts can be seen.

To help understand the available features of the tools we introduce one scenario based on Telemedicine Company (Clinical health care at distance). That is available in CORAS tutorial slides from (Lund, et al., 2011), which initiates a scenario where a hacker tries to break-into the system and steals the health records. In Figure 4-2 a deliberate threat diagram has been depicted where hacker/eavesdropper is on the most left and health records (assets) are on the most right. The lock (insufficient security) is vulnerability, (system break-in) is threat scenario, and finally (health records theft) is unwanted incident.

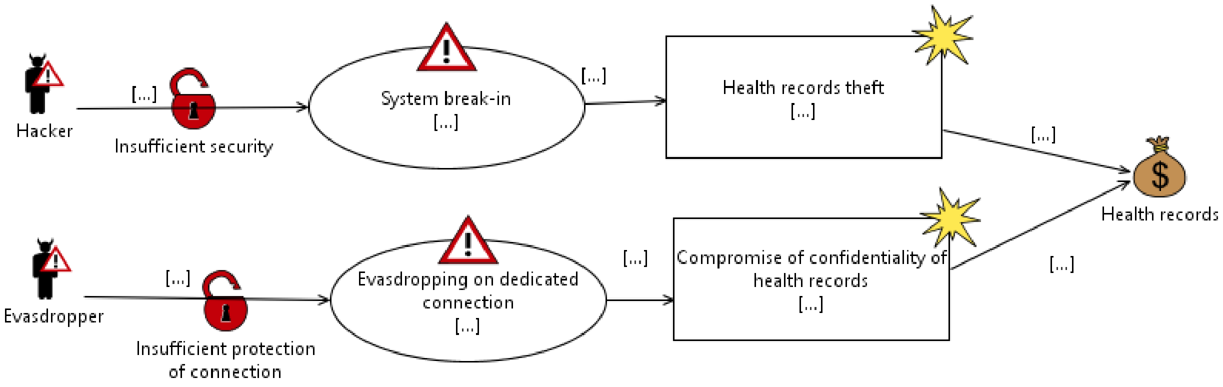


Figure 4-2 Deliberate Threat > Adapted from (Lund, et al., 2011)

Figure 4-2 is depicting a scenario of deliberate threat, in the face of a hacker/eavesdropper who tries to break-in to the system taking advantage of insufficient security and eventually staling health records.

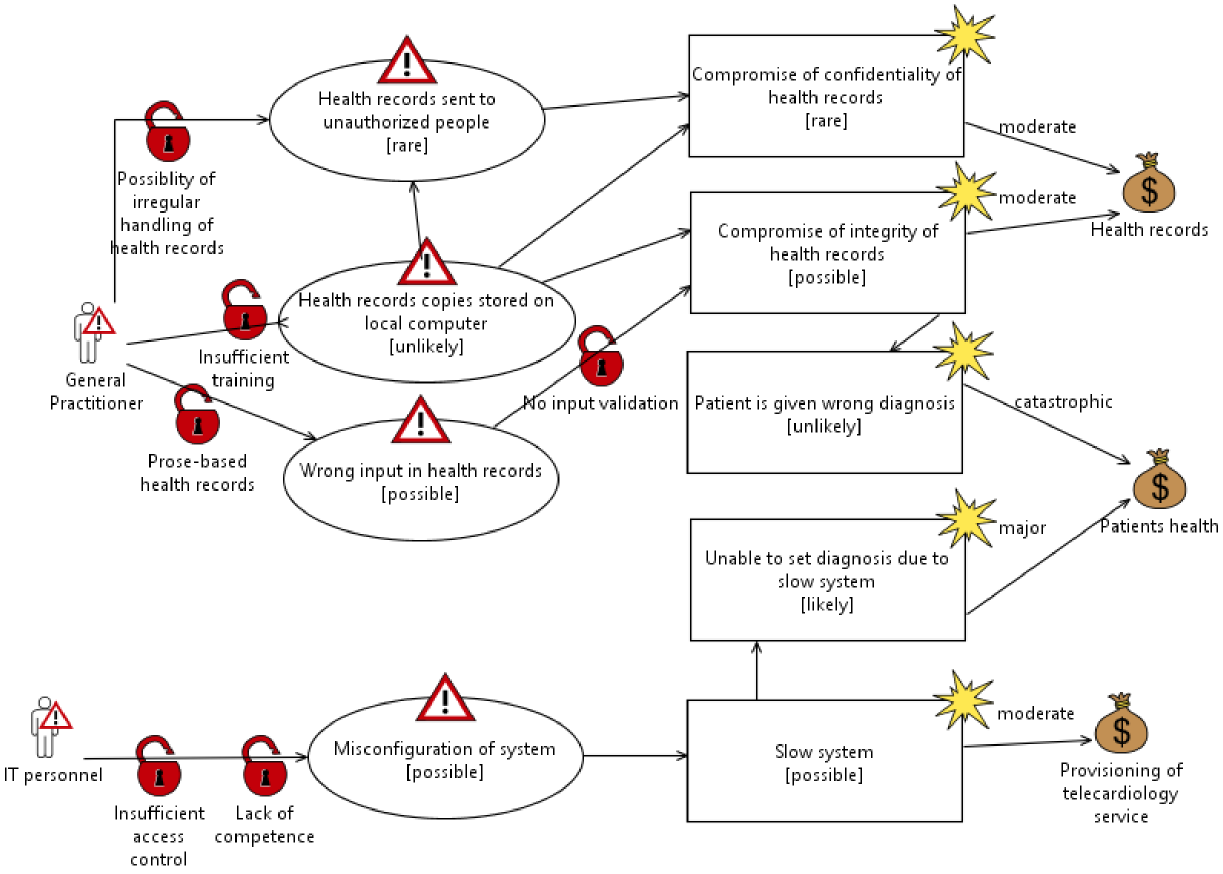


Figure 4-3 Accidental Threat > Adapted from (Lund, et al., 2011)

In Figure 4-3 the same threat diagram is depicting an accidental threat, with additional assets, vulnerabilities, unwanted incident, and threat scenario. In this case the diagram shows most of the possibilities of accidental threats, unlike previous diagram of deliberate threat.

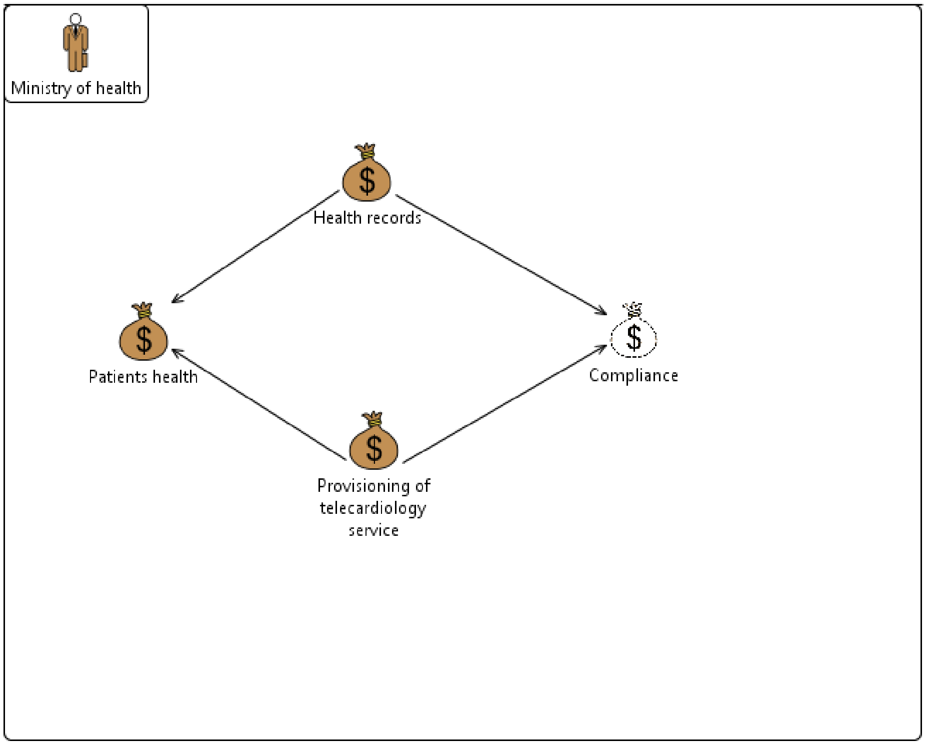


Figure 4-4 Telemedicine Assets > Adapted from (Lund, et al., 2011)

Figure 4-4 shows the asset diagram for telemedicine scenario, where all the assets are liable to compliance of telemedicine company. In case of any unwanted incident shown in Figure 4-4 telemedicine will equally contribute in compromise of compliance.

Figure 4-5 defines the risks associated with the telemedicine system, and their severity inside the brackets. Risk diagram's incident likelihood calculation formulas are also the key for calculating the severity mentioned in (Refsdal, 2014).

And finally in Figure 4-6 the treatment diagram is depicting most of the treatment scenarios in the box with the green spanner. All vulnerabilities cannot be addressed so most of them are treated accordingly.

As we have already depicted most of the available feature diagrams in CORAS now we can test the tool against requirements collected in previous chapter. Table 4-3 shows the requirement ID, requirement name, requirement fulfillment description, and means to fulfill requirement in CORAS tool. The requirements fulfillment from CORAS tool can also be seen in Figure 4-7.

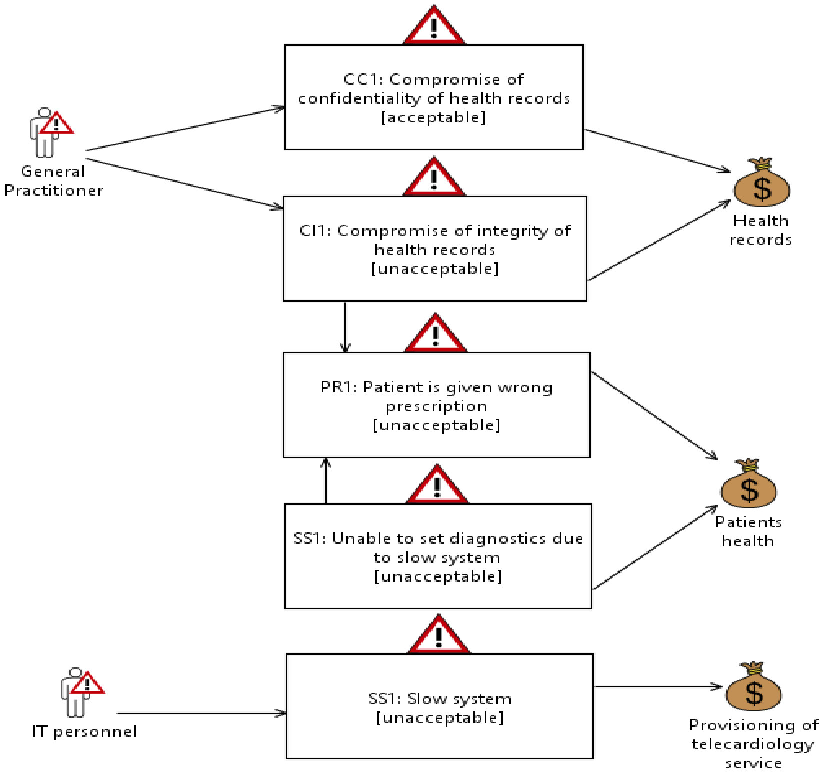

CC1: Compromise of confidentiality of health records [acceptable]
General Practitioner
CI1: Compromise of integrity of health records [unacceptable]
Health records
PR1: Patient is given wrong prescription [unacceptable]
Patients health
SS1: Unable to set diagnostics due to slow system [unacceptable]
IT personnel
SS1: Slow system [unacceptable]
Provisioning of telecardiology service

Figure 4-5 Telemedicine Risks > Adapted from (Lund, et al., 2011)

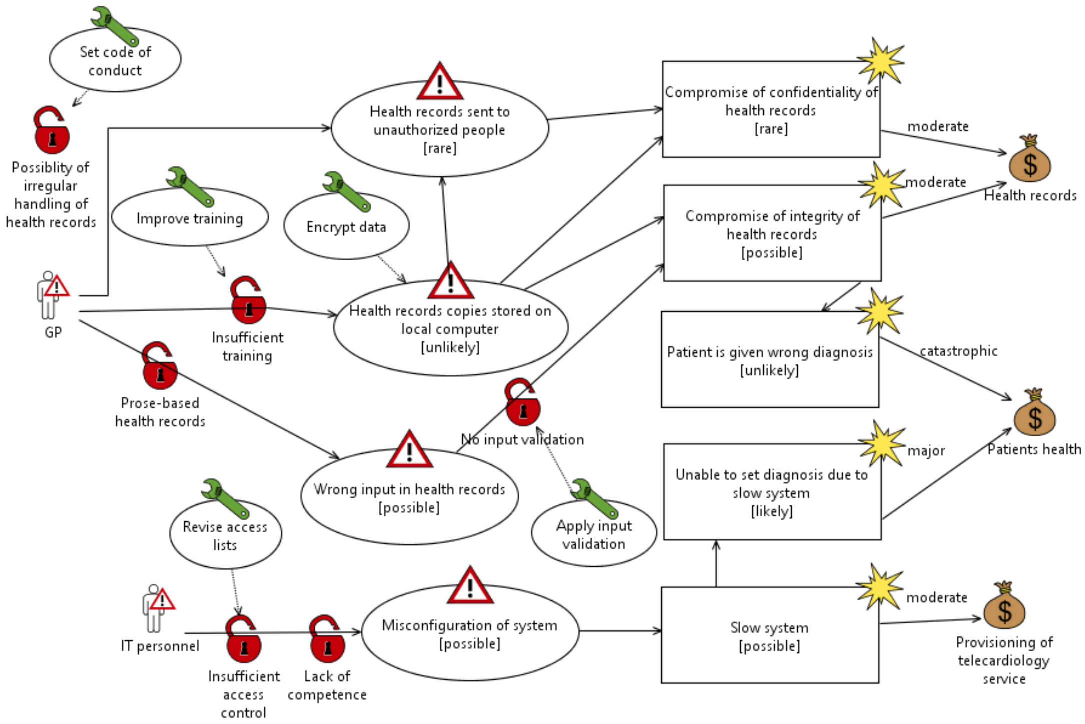

Set code of conduct
Possiblity of irregular handling of health records
Health records sent to unauthorized people [rare]
Compromise of confidentiality of health records [rare]
moderate
Improve training
Encrypt data
moderate
Health records
Compromise of integrity of health records [possible]
GP
Insufficient training
Health records copies stored on local computer [unlikely]
Patient is given wrong diagnosis [unlikely]
catastrophic
Prose-based health records
No input validation
major
Patients health
Wrong input in health records [possible]
Unable to set diagnosis due to slow system [likely]
Revise access lists
Apply input validation
IT personnel
Insufficient access control
Lack of competence
Misconfiguration of system [possible]
Slow system [possible]
moderate
Provisioning of telecardiology service

Figure 4-6 Telemedicine Treatment > Adapted from (Lund, et al., 2011)

Table 4-3 Requirements from CORAS tool

REQ ID	Requirement Name	Requirement fulfillment description	Means to fulfill requirement (Status)
SR001	Making awareness	CORAS tool includes the methodology and tutorials which in an instance give a glimpse that the method/tool is trying to achieve the first requirements. Also CORAS website http://coras.sourceforge.net/ provides seminars sometimes which will be helpful for the software development team. In this case CORAS fulfills this requirement	1. Resources available at http://coras.sourceforge.net/ 2. Methodology 3. Tutorial at: http://coras.sourceforge.net/newsarchive.html 4. Publications at: http://coras.sourceforge.net/online_documentation.html
SR002	Understanding context and assets	In correspondence to Figure 4-4 CORAS provides a way to demonstrate the assets along with the compliance in case of unwanted incident, so clearly CORAS also fulfills this requirement also	1. Asset diagram
SR003	Security requirements	In Figure 4-6 CORAS tries to fulfill the security requirements in the face of mitigations. The diagram itself shows the threats, vulnerabilities and unwanted incidents which are been addressed with the use of treatment diagram	1. Treatment diagram
SR004	Risk analysis	In Figure 4-5 CORAS Risk diagram shows the risks in the face of unwanted incidents, so it proves that CORAS also fulfill this requirement	1. Risk diagram 2. Threat diagram
SR005	Secure design practices	Tool does not address anything related to software architecture, and or implementation of secure design, which leads us to fail this requirement in CORAS tool	-NA
SR006	Justify design solution	CORAS method and tool provides some manual techniques to calculate the severity and likelihood of the incident. But it does not provide anything in the tool itself that can automate this calculation process. As there is some technique to calculate we make this requirements as fulfilled	1. Calculus for likelihood reasoning, available at http://coras.sourceforge.net/documents/tutorials/part1_CCS2011_CORAS.pdf
SR007	Response	CORAS does not provide any thing in correspondence to the response plan. However the tool has capability to demonstrate the response plan by using the available artifacts. For example Response diagram can be drawn using the artifacts used in Figure 4-6 treatment diagram. So we take this requirement as fulfilled	1. Means to create response plan diagram

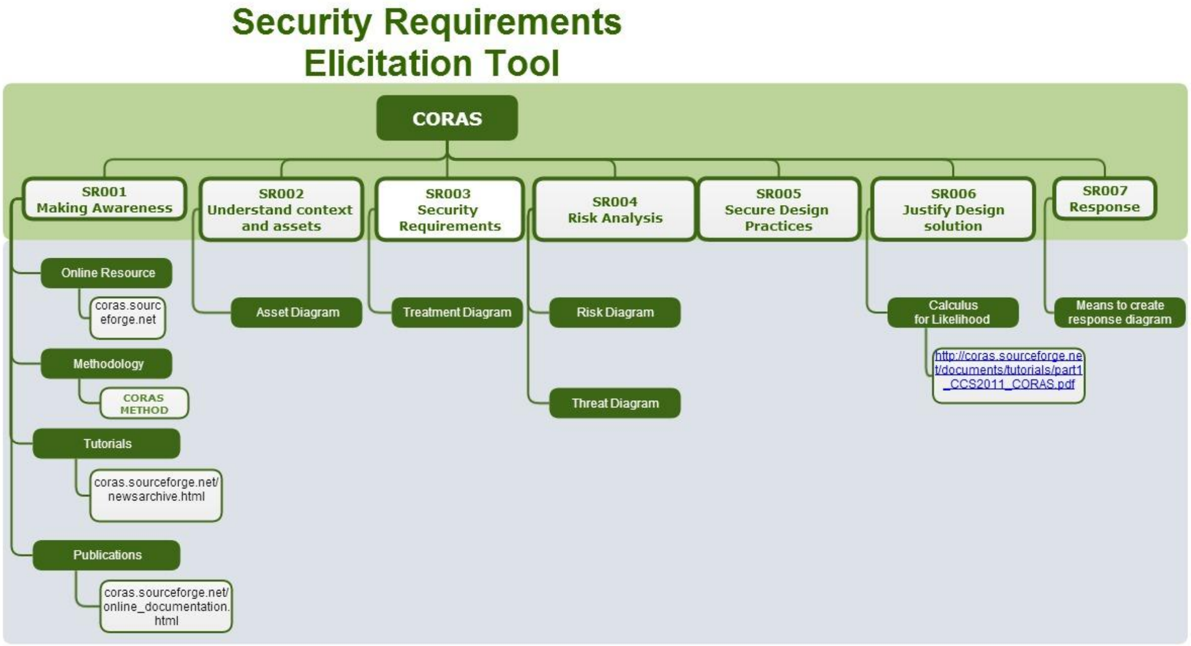

Security Requirements Elicitation Tool
CORAS
SR001 Making Awareness
SR002 Understand context and assets
SR003 Security Requirements
SR004 Risk Analysis
SR005 Secure Design Practices
SR006 Justify Design solution
SR007 Response
Online Resource
coras.sourceforge.net
Methodology
CORAS METHOD
Tutorials
coras.sourceforge.net/newsarchive.html
Publications
coras.sourceforge.net/online_documentation.html
Asset Diagram
Treatment Diagram
Risk Diagram
Threat Diagram
Calculus for Likelihood
http://coras.sourceforge.net/documents/tutorials/part1_CCS2011_CORAS.pdf
Means to create response diagram

Figure 4-7 Requirements fulfillment from CORAS

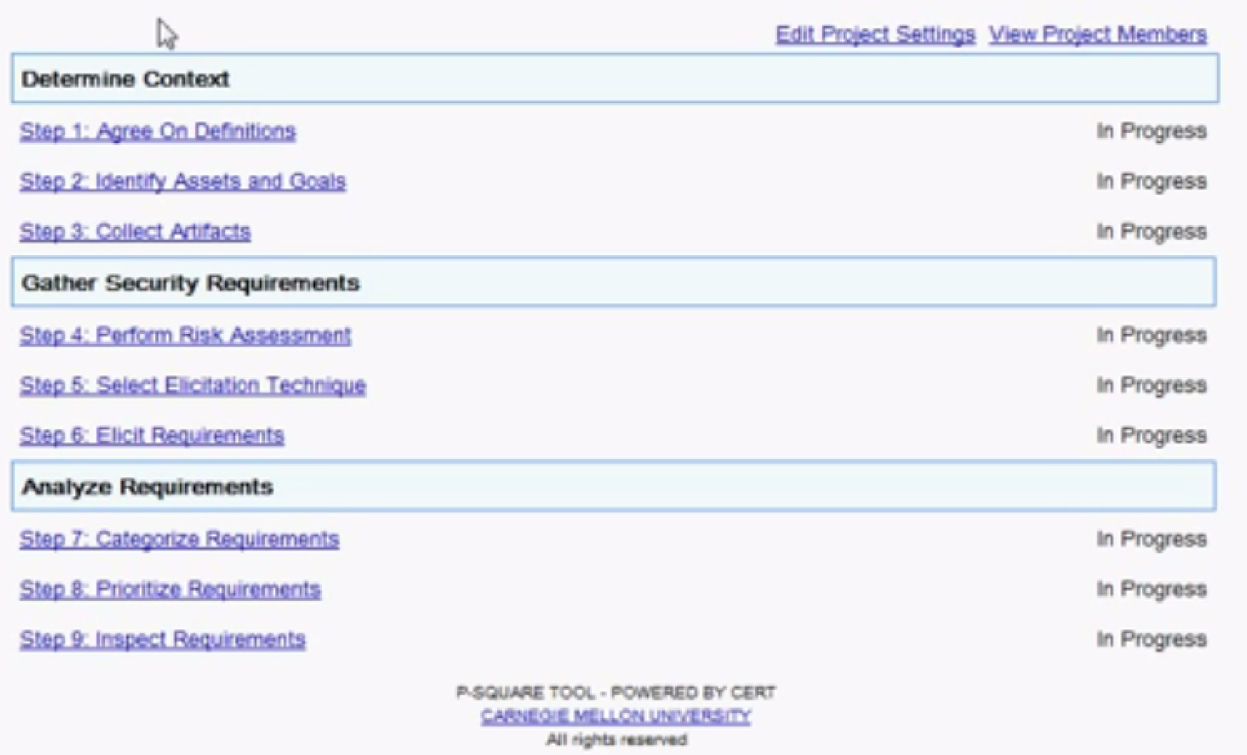
4.2.2. Capabilities of SQUARE Tool

The SQUARE tool is developed in Google web tool kit, which allows users to create projects from the perspective of security, and privacy or both. After creating a project in SQUARE tool user can see three categorize or stages: Determine context, Gather security requirements, and Analyze requirements. Table 4-4 shows these categories along with internal steps which cover all the aspects of software security requirements that can also be seen in Figure 4-8.

Table 4-4 Steps in SQUARE tool

	Step ID:	Category	Step
STEPS IN SQUARE TOOL	1	Determine context	Agree on definition
	2		Identify assets and goals
	3		Collect artifacts
	4	Gather security requirements	Perform risk assessment
	5		Select elicitation techniques
	6		Elicit requirements
	7	Analyze requirements	Categorize requirements
	8		Prioritize requirements
	9		Impact requirements

In order to simplify the tool analysis process we will take the same example that we have used in CORAS tool about Telemedicine. The example was of securing the health records in telemedicine system that includes two core actors i.e. IT personnel and general practitioner (technician). SQUARE tool however provides with a generic template for requirement elicitation, which can be useful for several projects but does not provide the artifacts or diagrams that we need for this example. So to fulfill the requirement we will try to generate the closest results possible to test this tool.



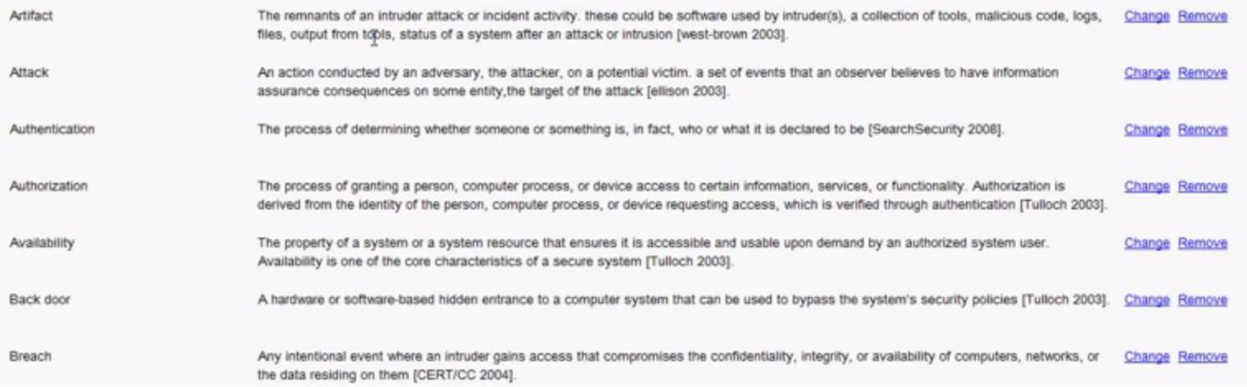
Edit Project Settings View Project Members

Determine Context	
Step 1: Agree On Definitions	In Progress
Step 2: Identify Assets and Goals	In Progress
Step 3: Collect Artifacts	In Progress
Gather Security Requirements	
Step 4: Perform Risk Assessment	In Progress
Step 5: Select Elicitation Technique	In Progress
Step 6: Elicit Requirements	In Progress
Analyze Requirements	
Step 7: Categorize Requirements	In Progress
Step 8: Prioritize Requirements	In Progress
Step 9: Inspect Requirements	In Progress

P-SQUARE TOOL - POWERED BY CERT
CARNEGIE MELLON UNIVERSITY
All rights reserved

Figure 4-8 Steps in SQUARE tool > Adapted from (Ganguly, 2011)

In step 1 of SQUARE tool we try to gather all the terms for the current project. For our example of telemedicine we use almost same terms as in example demonstrated in (Ganguly, 2011) video. In step 2 of SQUARE tool we gather assets and goals mentioned in Table 4-5



Artifact	The remnants of an intruder attack or incident activity. these could be software used by intruder(s), a collection of tools, malicious code, logs, files, output from tools, status of a system after an attack or intrusion [west-brown 2003].	Change Remove
Attack	An action conducted by an adversary, the attacker, on a potential victim. a set of events that an observer believes to have information assurance consequences on some entity,the target of the attack [ellison 2003].	Change Remove
Authentication	The process of determining whether someone or something is, in fact, who or what it is declared to be [SearchSecurity 2008].	Change Remove
Authorization	The process of granting a person, computer process, or device access to certain information, services, or functionality. Authorization is derived from the identity of the person, computer process, or device requesting access, which is verified through authentication [Tulloch 2003].	Change Remove
Availability	The property of a system or a system resource that ensures it is accessible and usable upon demand by an authorized system user. Availability is one of the core characteristics of a secure system [Tulloch 2003].	Change Remove
Back door	A hardware or software-based hidden entrance to a computer system that can be used to bypass the system's security policies [Tulloch 2003].	Change Remove
Breach	Any intentional event where an intruder gains access that compromises the confidentiality, integrity, or availability of computers, networks, or the data residing on them [CERT/CC 2004].	Change Remove

Figure 4-9 Step 1 SQUARE tool > Adapted from (Ganguly, 2011)

Table 4-5 Assets and goals SQUARE tool

	Priority	Goals	Assets
ASSETS AND GOALS	1	Apply input validation; Encrypt data; Improve training; Set code of conduct.	Health records
	2	Apply input validation to avoid wrong prescription	Patients health
	3	Revise access control list	Provision of tele-cardiology service

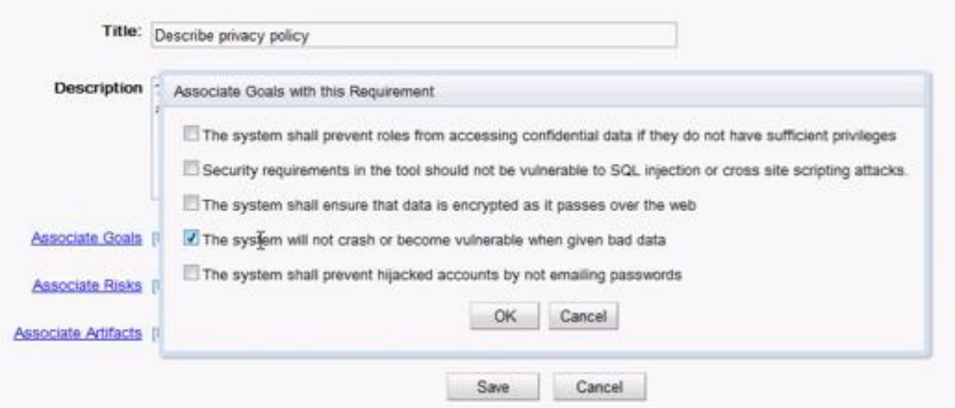
In step 3 of SQUARE tool we gather artifacts which in comparison with SSAP's is the awareness. The literature can be for report on telemedicine code of conduct, or case study for telemedicine current security specifications.

In step 4 of SQUARE tool perform risk assessment the risks are been associated with artifacts and goals. Table 4-6 adds in one additional column for risks associated with goals and assets mentioned in Table 4-5.

Table 4-6 Perform risk assessment SQUARE tool

	Priority	Goals	Assets	Risks
PERFORM RISK ASSESSMENT	1	Apply input validation; Encrypt data; Improve training; Set code of conduct.	Health records	Theft of health records
	2	Apply input validation to avoid wrong prescription	Patients health	Risk associated with patients life
	3	Revise access control list	Provision of tele-cardiology service	Tele-cardiology service unavailability

In step 5 of SQUARE tool elicitation techniques the tool provides a mechanism to choose elicitation technique by asking ten standardize questions to which answers provide privacy requirements generated. That then can be associated with risks and goals. The Figure 4-10 shows an example of associating requirement that has been generated by the SQUARE tool.



Title: Describe privacy policy

Description

Associate Goals with this Requirement

☐ The system shall prevent roles from accessing confidential data if they do not have sufficient privileges

☐ Security requirements in the tool should not be vulnerable to SQL injection or cross site scripting attacks.

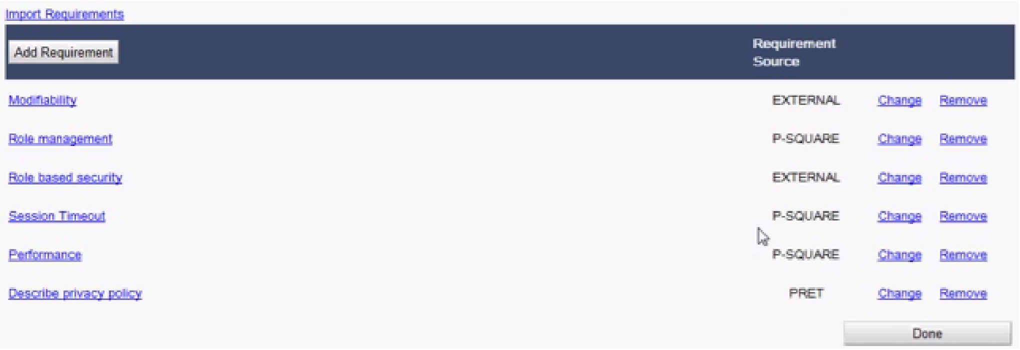
☐ The system shall ensure that data is encrypted as it passes over the web

☑ The system will not crash or become vulnerable when given bad data

☐ The system shall prevent hijacked accounts by not emailing passwords

Figure 4-10 Elicitation Techniques SQUARE tool > Adapted from (Ganguly, 2011)

In step 6 elicit requirements SQUARE tool uses the same generated requirements elicited in elicitation technique, for the further uses. See Figure 4-11 that depicts elicited security requirements. The step 7 is about categorizing these elicited requirements, in Figure 4-12 a requirement with an example can be seen needs to be categorized.

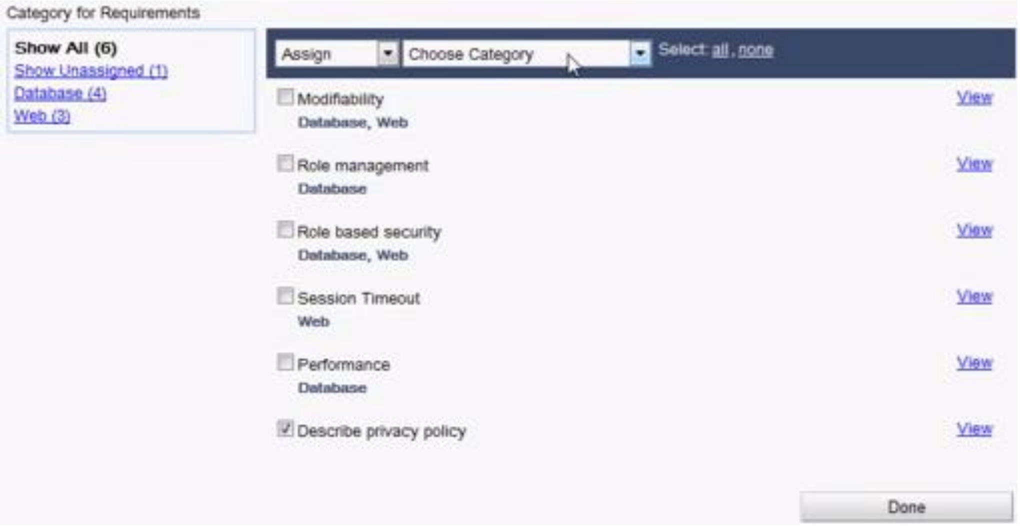


Import Requirements

Add Requirement	Requirement Source		
Modifiability	EXTERNAL	Change	Remove
Role management	P-SQUARE	Change	Remove
Role based security	EXTERNAL	Change	Remove
Session Timeout	P-SQUARE	Change	Remove
Performance	P-SQUARE	Change	Remove
Describe privacy policy	PRET	Change	Remove

Done

Figure 4-11 Elicit Requirements SQUARE tool > Adapted from (Ganguly, 2011)



Category for Requirements

Show All (6)
Show Unassigned (1)
Database (4)
Web (3)

Assign ▾ Choose Category ▾ Select: all , none

□ Modifiability Database, Web	View
□ Role management Database	View
□ Role based security Database, Web	View
□ Session Timeout Web	View
□ Performance Database	View
☑ Describe privacy policy	View

Done

Figure 4-12 Categorize Requirements SQUARE tool > Adapted from (Ganguly, 2011)

In step 8 prioritize requirements we use to prioritize these requirements, in terms of severity or criticality. And finally in step 9 inspect requirements, SQUARE implements a mechanism for inspection.

Getting close to the available results, we can assume that most the requirements were been fulfilled with some exceptions Table 4-7 shows the requirement ID, requirement name, fulfillment description and means to fulfill requirements from this tool. Requirements fulfillment from SQUARE tool can also be seen in Figure 4-13.

Table 4-7 Requirements from SQUARE tool

REQ ID	Requirement Name	Requirement fulfillment description	Means to fulfill requirement (Status)
SR001	Making awareness	SQUARE tool in step 3 provides with the "collect artifact" which can be used to store information for making awareness or available case-studies. So we make this requirement to be fulfilled.	1. Resources available at http://www.cert.org/cybersecurity-engineering/products-services/square.cfm? 2. Methodology (SQUARE) 3. Agree on definitions (gather terms) 4. Collect artifact (Check list)
SR002	Understanding context and assets	In step 2 and 4 SQUARE tool provides means to declare assets and goals that gives understanding that this tool fulfills this requirement.	1. Identify assets and goals (check list of assets & goals) 2. Template to document assets & goals
SR003	Security requirements	During step 5 to 8 SQUARE tool uses questions to analyze the security needs. Based on the answers tool provides with a standard template that can be customized according to the needs. We can make this requirement to be fulfilled also	1. Select elicitation technique 2. Template for security requirements
SR004	Risk analysis	In step 4 of SQUARE tool "perform risk assessment" we have defined few risks based on assets and goals. That leads us to fulfill this requirement also	1. Check list of vulnerabilities 2. Check list of security threats 3. Check list of risks 4. Template to define risks
SR005	Secure design practices	SQUARE tool does not provide any thing about design practices, which leads us to fail this requirement	-NA
SR006	Justify design solution	SQUARE tool in steps 7 & 8 provides with the concept of categorizing and prioritizing security requirements. That could be the part of justified design solution. In this case tool fulfills this requirement.	1. Security requirements categorization and prioritization
SR007	Response	In step 9 of SQUARE tool called inspect requirement, gives a possibility for inspection which can be used as a response plan. That makes us to fulfill this requirement.	1. Inspect requirement (can be used as a response plan also)

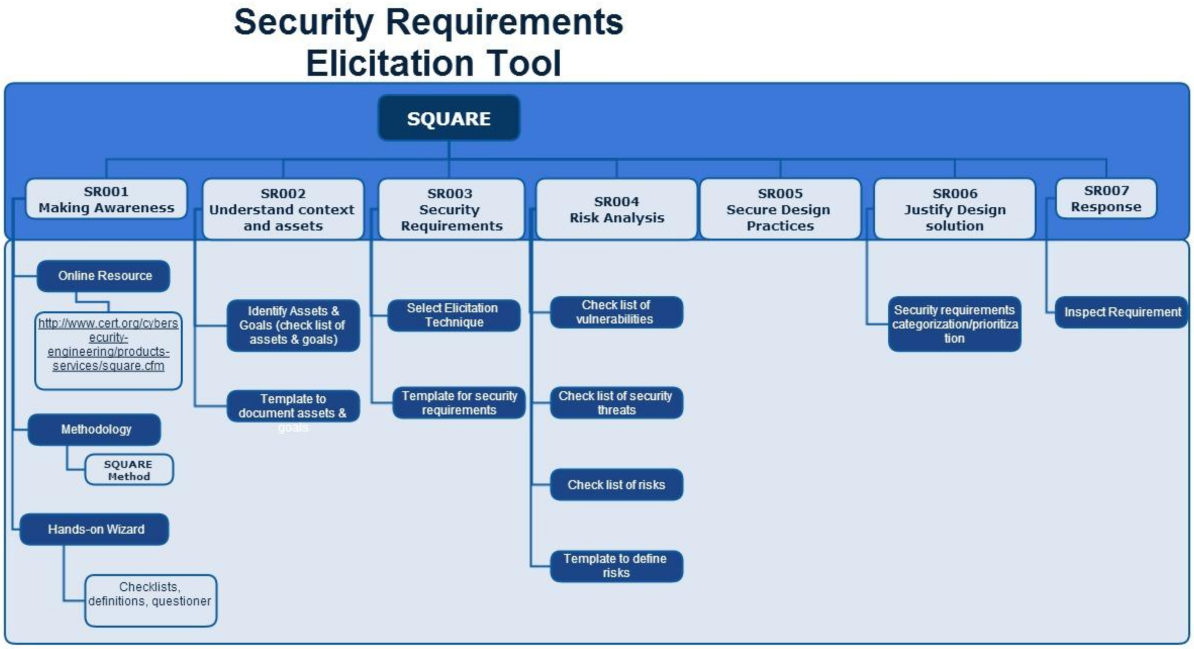

Security Requirements Elicitation Tool
SQUARE
SR001 Making Awareness
SR002 Understand context and assets
SR003 Security Requirements
SR004 Risk Analysis
SR005 Secure Design Practices
SR006 Justify Design solution
SR007 Response
Online Resource
http://www.cert.org/cybersecurity-engineering/products-services/square.cfm
Methodology
SQUARE Method
Hands-on Wizard
Checklists, definitions, questioner
Identify Assets & Goals (check list of assets & goals)
Template to document assets & goals
Select Elicitation Technique
Template for security requirements
Check list of vulnerabilities
Check list of security threats
Check list of risks
Template to define risks
Security requirements categorization/prioritization
Inspect Requirement

Figure 4-13 Requirements fulfillment from SQUARE tool

4.2.3. Capabilities of SecTro2 Tool

"Secure Tropos is a security-aware software systems development methodology, which combines requirements engineering concepts, such as actor, goal, plan together with security engineering concepts such as threat, security constraint and security mechanism, under a unified process to support the analysis and development of secure and trustworthy software systems" (Secure Tropos, 2014). SecTro2 tool is a security requirements elicitation tool that is based on Tropos methodology.

SecTro2 enables software analysts to gather requirements providing various artifacts available in tool. These artifacts are categorized under organizational view, security requirements view, security components view, security attacks view, and cloud analysis view. In appendix-A SecTro2's artifacts are depicted.

To make it simpler we will introduce the same example of telemedicine available in (Lund, et al., 2011) to demonstrate the capabilities of SecTro2 tool. As the tool adapts different approach, the results may vary from the original. At first we have tried to create the organizational view which can be seen in Figure 4-14 where in organization (telemedicine) a general practitioner is dedicated to store the health records and IT personnel is dedicated to keep these records safe.

In Figure 4-15 security requirements view is depicted in comparison to CORAS tool this diagram represents accidental threats. Where two actors' general practitioner and IT personnel are responsible for keeping health records, patient's health, and provision of tele-cardiology service safe. In regards to keep system running they need to implement some security constraints, which are "keep health records safe", "implement input validation" and "proper configuration of the system". Moreover both tools CORAS and SecTro2 have some differences so we only can depict the deliberate and accidental threat in one "security attacks view". However the security attacks view is only for defining the attack scenario.

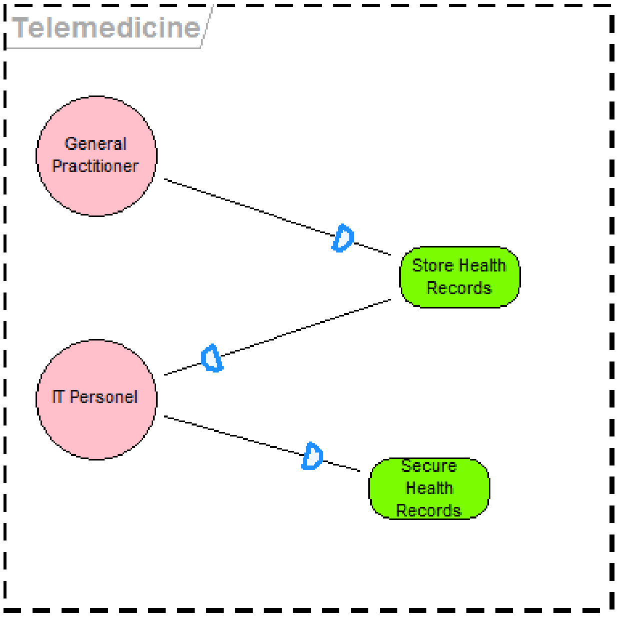


Figure 4-14 Organization View

When expending the threats mentioned in Figure 4-15 security attacks view can be seen, which then allows to introduce vulnerabilities, attack method, and attack scenario. We have security attacks view for all four threats mentioned in Figure 4-15 that can be seen from Figure 4-16 to Figure 4-19.

In Figure 4-16 scenario is depicted where improper handling of health records leads to health records been sent to unauthorized personnel. Two vulnerabilities mentioned "insufficient training" and "possibility of irregular handling of health records" are responsible for this incident. Figure 4-17 is similar to Figure 4-16 in regards with they both mention the improper handling of health record.

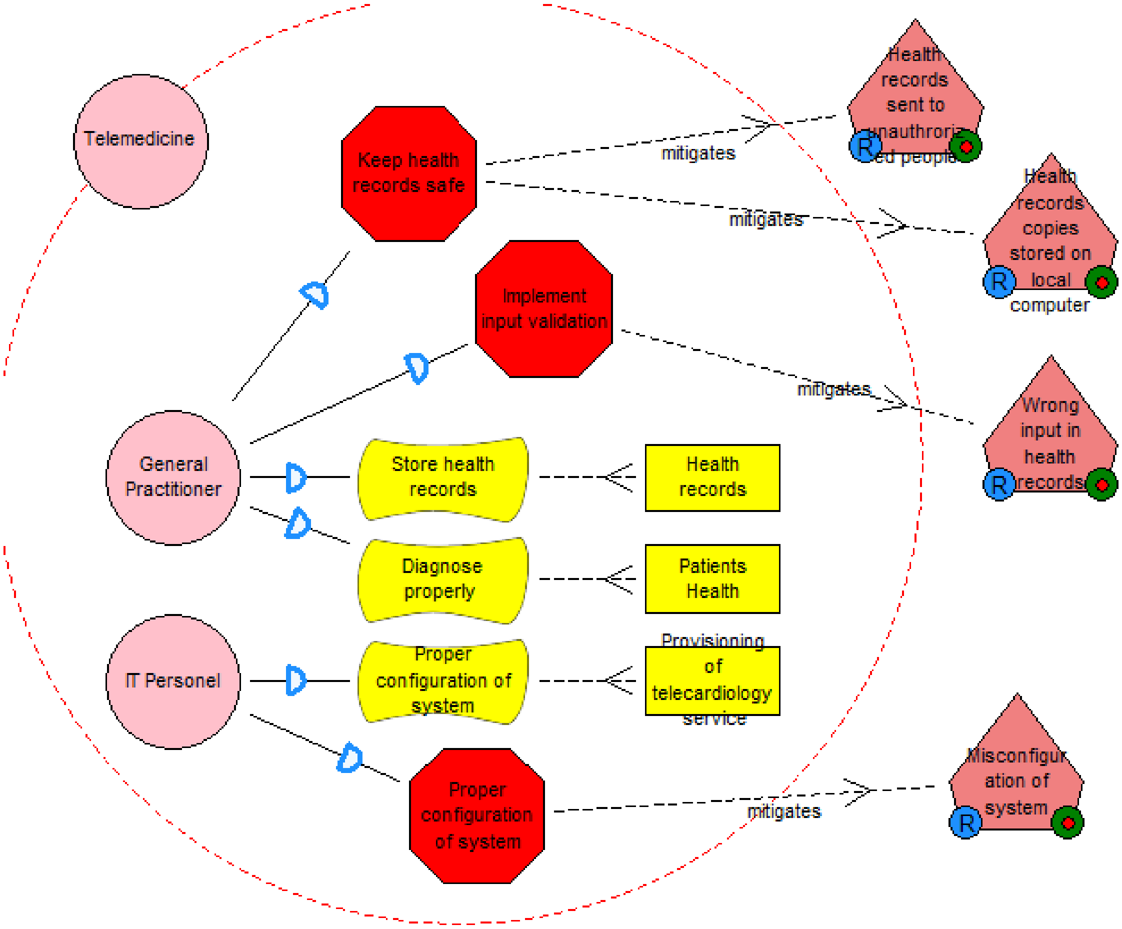

Telemedicine
Keep health records safe
mitigates
Health records sent to unauthrorized people
R
mitigates
Health records copies stored on local computer
R
Implement input validation
mitigates
Wrong input in health records
R
General Practitioner
Store health records
Health records
Diagnose properly
Patients Health
IT Personel
Proper configuration of system
Provisioning of telecardiology service
Proper configuration of system
mitigates
Misconfiguration of system
R

Figure 4-15 Security Requirements View – Accidental threat

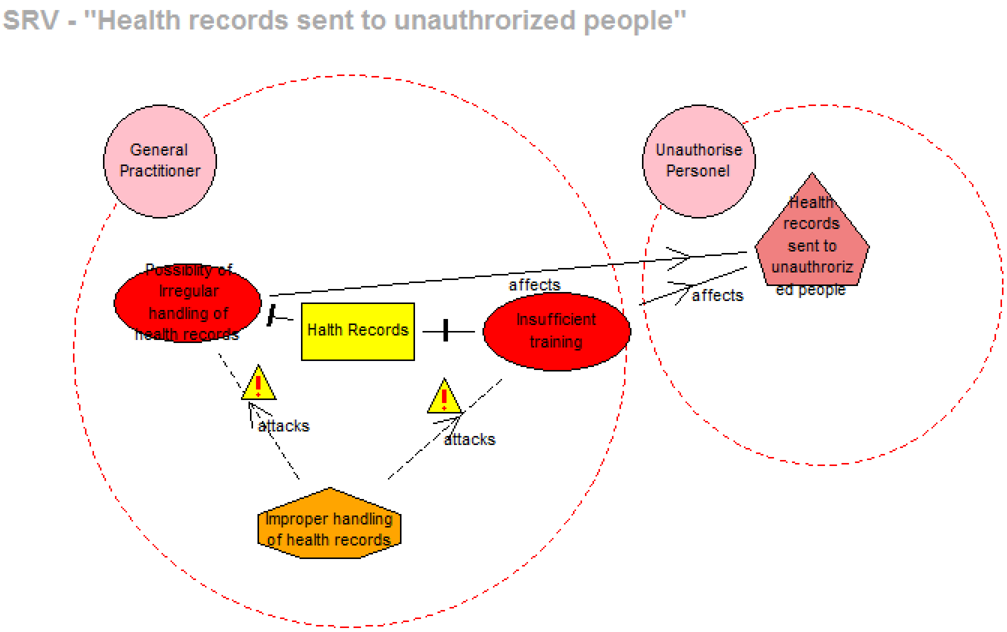

SRV - "Health records sent to unauthrorized people"
General Practitioner
Unauthorise Personel
Health records sent to unauthroriz ed people
affects
Possibility of Irregular handling of health records
affects
Halth Records
Insufficient training
attacks
attacks
Improper handling of health records

Figure 4-16 Security Attacks View - Health records sent to unauthorized people

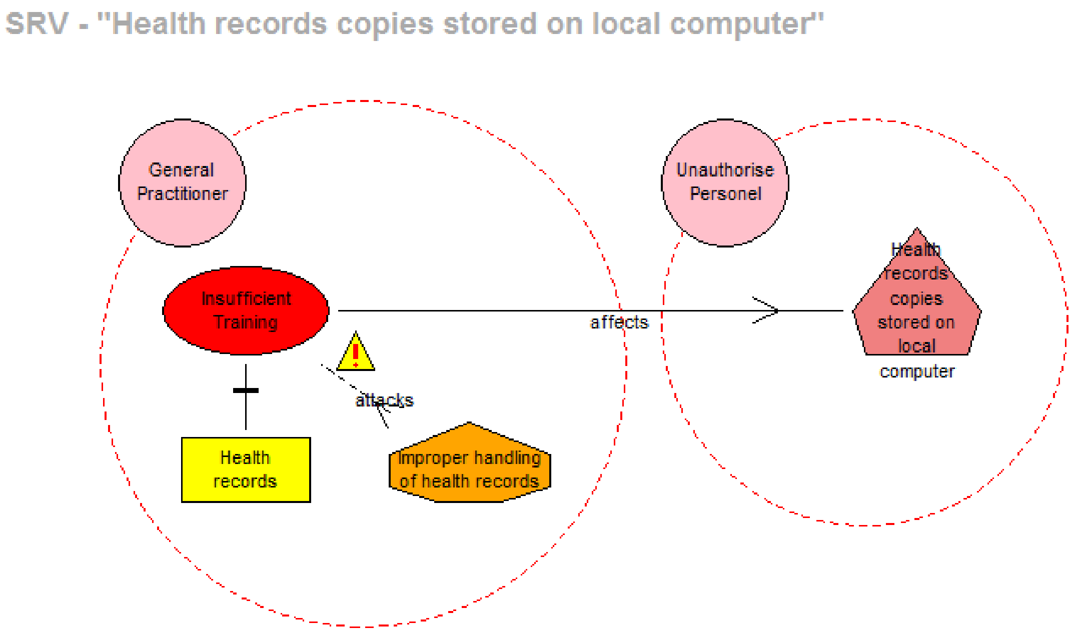

SRV - "Health records copies stored on local computer"
General Practitioner
Unauthorise Personel
Insufficient Training
affects
Health records copies stored on local computer
attacks
Health records
Improper handling of health records

Figure 4-17 Security Attacks View - Health records copies stored on local computer

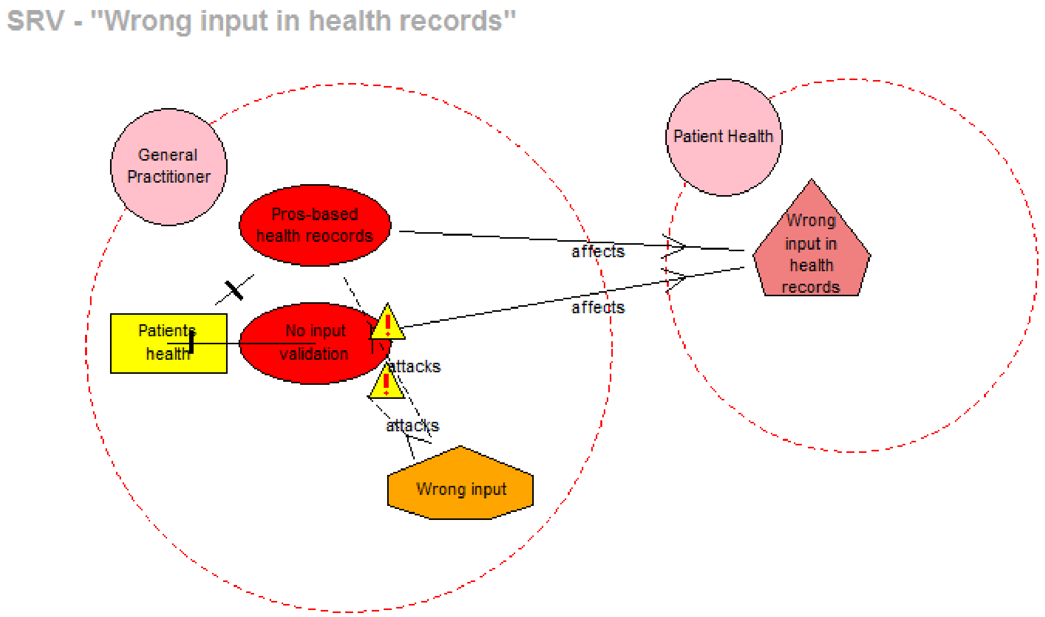


Figure 4-18 Security Attacks View - Wrong input in health records

In Figure 4-18 scenario is depicted where a practitioner inputs the wrong information in health records, making unreliable health records, two vulnerabilities are responsible for this a no input validation, and pros-based health records.

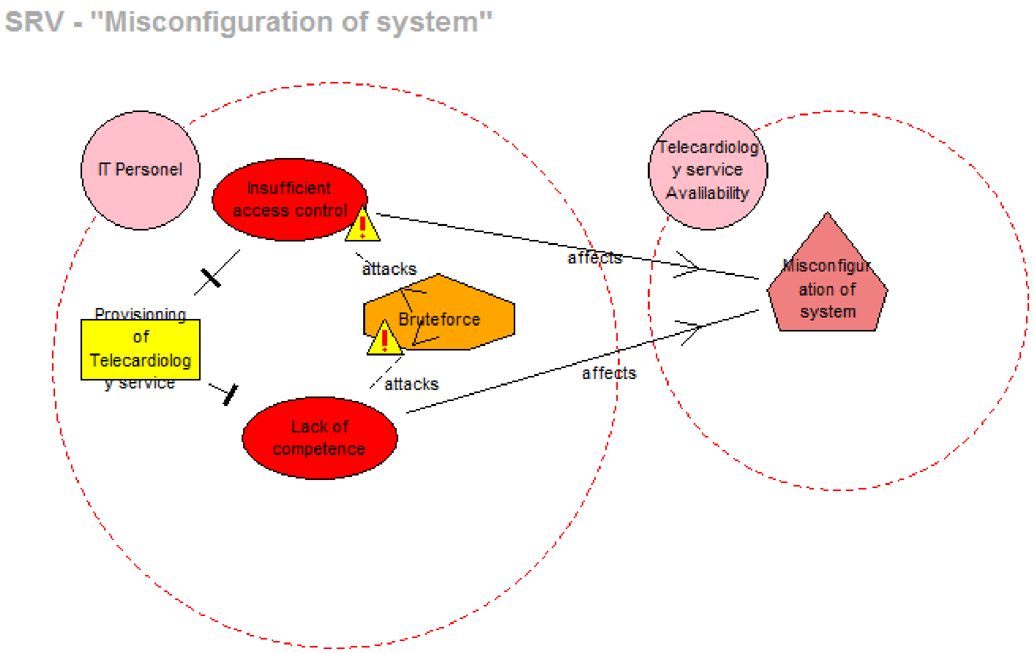

SRV - "Misconfiguration of system"
IT Personel
Insufficient access control
attacks
affects
Telecardiology service Availability
Misconfiguration of system
Provisioning of Telecardiology service
Bruteforce
attacks
affects
Lack of competence

Figure 4-19 Security Attacks View - Misconfiguration of system

Misconfiguration of system can make tele-cardiology service unavailable in Figure 4-19 scenario where a brute force attack and insufficient access control can lead to service unavailability.

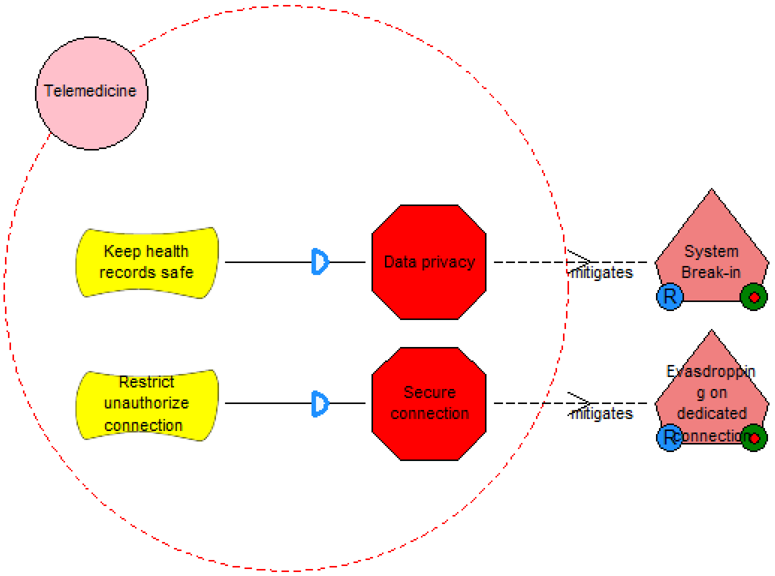


Figure 4-20 Security Requirements View – Deliberate Threat

In comparison to CORAS deliberate threat diagram, Figure 4-20 presents a view which includes an eavesdropper and hacker, that can be seen in Figure 4-21 and Figure 4-22. Telemedicine system has two soft goals, keep health records safe and restrict unauthorized connection. These are linked to security constraint “data privacy” and “secure connection”.

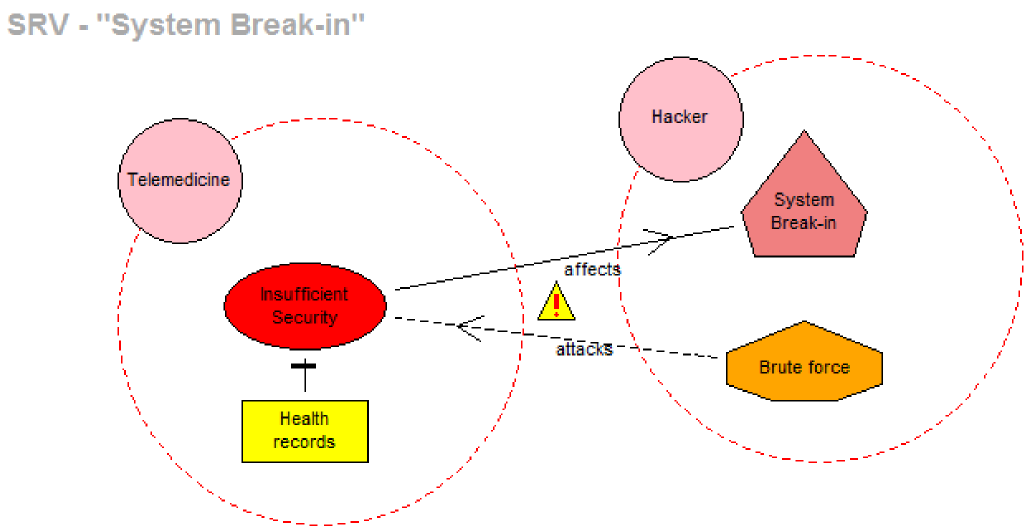


Figure 4-21 Security Attacks View - System Break-in

In Figure 4-21 system break-in is depicted which provides a view on how insufficient security can lead to loss of valuable information, where hacker tries to break-in to the system by using brute-force method.

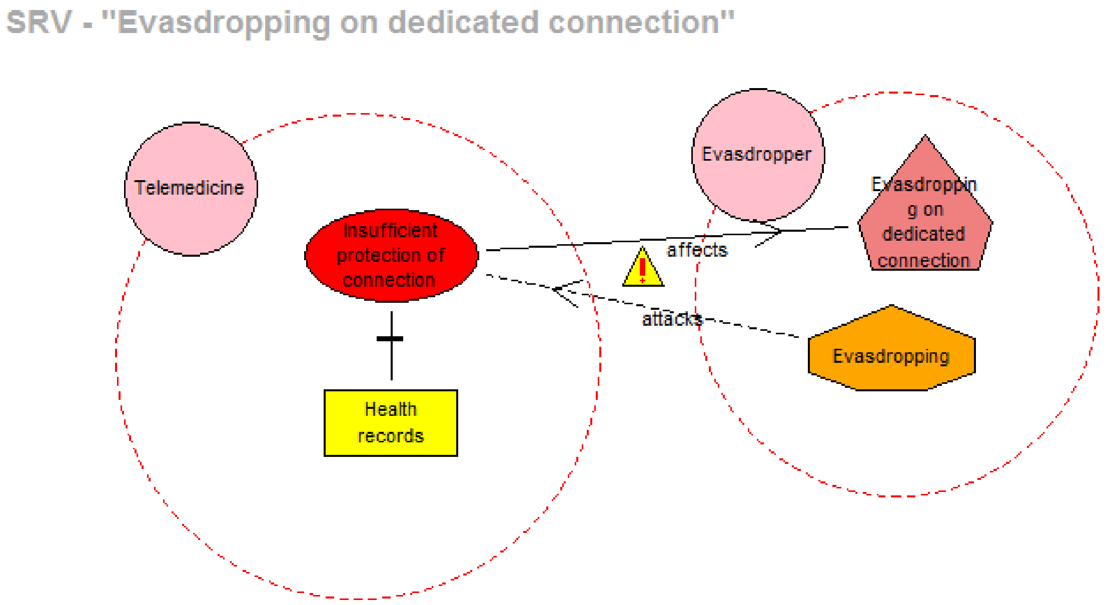

SRV - "Evasdropping on dedicated connection"
Evasdropper
Telemedicine
Evasdropping on dedicated connection
Insufficient protection of connection
affects
attacks
Evasdropping
Health records

Figure 4-22 Security Attacks View - Eavesdropping on dedicated connection

In Figure 4-22 an eavesdropper tries to listen to unprotected connection of telemedicine which can lead to loss of health records.

In example we have learned how SecTro2 provide users with different views that can be useful in demonstrating and eliciting security requirements. Our motive of “means to fulfill requirements" from SSAP's to security engineering tools can now be seen in Table 4-8 for SecTro2 tool. Requirements fulfillment from SecTro2 can also be seen in Figure 4-23.

Table 4-8 Requirements from SecTro2

REQ ID	Requirement Name	Requirement fulfillment description	Means to fulfill requirement (Status)
SR001	Making awareness	SecTro2 is adapting Tropos methodology, which means the tool follows security engineering methodology. Also offers online leaning resources at http://securetropos.org/	1. Resources available at http://securetropos.org/ 2. Methodology (Secure Tropos) 3. User manual 4. Publications http://securetropos.org/publications/
SR002	Understanding context and assets	In most diagrams the resource artifact is used which itself shows that tool provides the understanding of Assets. See Figure 4-15 for e.g. In this case SecTro2 fulfills this requirement.	1. Context diagram (security requirements view) 2. Asset diagram (security requirements view)
SR003	Security requirements	In Figure 4-15 and Figure 4-20 security constraint artifact is used which can represent the security requirements. So SecTro2 also fulfill this requirement.	1. Security requirements view
SR004	Risk analysis	Figure 4-21 and Figure 4-22 shows the threat and vulnerability artifact, which in Figure 4-15 is only depicts the risk associated "health records have been sent to unauthorized person". Mitigations can be seen when expended. SecTro2 fulfill this requirement also.	1. Threat diagram (security attacks view)
SR005	Secure design practices	SecTro2 does not include anything related to secure software architecture, but as it includes security components view that can be useful in terms of defining security mechanism which in return will suggest secure design mechanism. That leads us to consider this requirement as fulfilled.	1. Security components diagram (security components view)
SR006	Justify design solution	SecTro2 does not contain anything related to trade-off analysis, likelihood of certain event or anything related to justification of design. That means SecTro2 fails this requirement.	-NA
SR007	Response	If we consider artifact security constraint and using security mechanism artifact, one can create a response plan. We can consider this requirement as fulfilled.	1. Security response diagram (security requirements view)

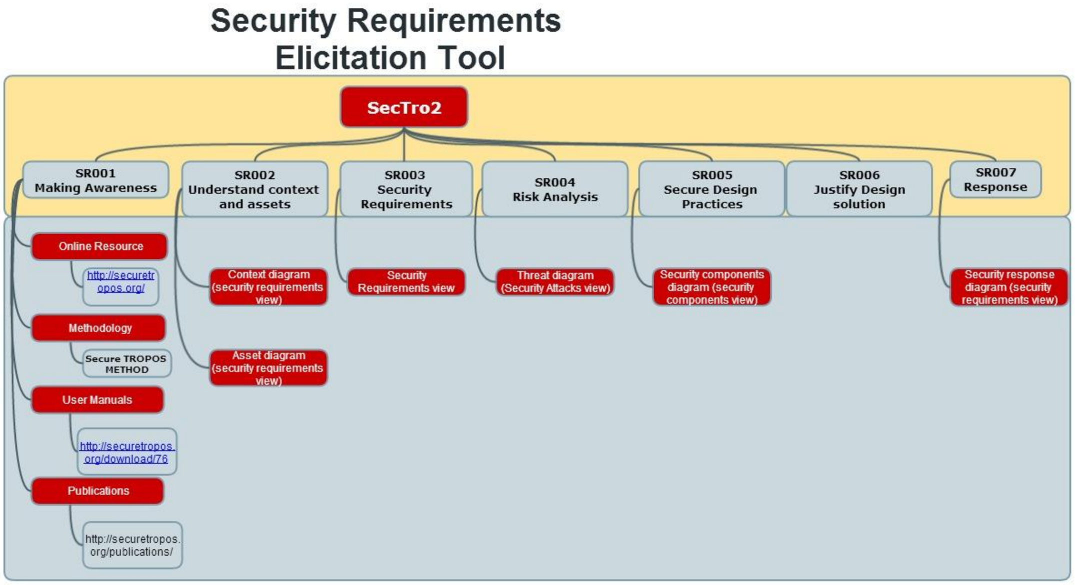


Figure 4-23 Requirements fulfillment from SECTRO2 tool

4.3. Chapter Summary

In this chapter we have introduced three software security requirements elicitation tools, and tested against requirements gathered from software security assurance processes (SSAP's), and came up with the means to fulfill those requirements. In conclusion we have gathered all the required instruments to construct a framework. That will help in choosing the right security requirements elicitation tool. Coming up next we will define a method to choose between security requirements elicitation tools.

5

Framework for security requirements engineering tool

In the previous chapter we have finalized analyzing the tools against SSAP's which gave us understanding on how the tools elicit security requirements. Based on these findings we can now develop a method which will enable us to construct a framework to choose security engineering tools based on these requirements. In Figure 5-2 the framework can be visited that is constructed with the help of tool analysis against requirements available in Table 5-1. In the last part of this chapter we have discussed about the usage of this framework that can be visited in Figure 5-2.

5.1. Derived Software Security Requirements

In chapter 3 we have analyzed SSAP's and gather the requirements for security engineering tools. In this chapter we have combined all these requirements to form one table. Requirements shown in Table 5-1 are based on SDL, 7 Touch points and OWASP, where original requirements are mentioned along with base artifact. Requirements for security engineering processes are divided into phases of software development, originally the phases in Microsoft SDL. This combined outcome from SSAP's has been enriched with means to fulfill these requirements from previous chapter of tools analysis. The table structure is same as tables in SSAP's, requirement ID with additional ID from SSAP, artifact name, and finally the actual requirement with precise description.

Table 5-1 Derived Requirements from SSAP

REQ ID	PREQ ID	Artifact	Requirement
PH1: Training			
SR001	*SRS001	Core Training	***(Making Awareness)*** Tool should provide basic training material that will help understand the implementation of security solution.
	#SRC001	Institute Awareness program	
PH2: Requirements and use cases			
SR002	*SRS002	Establish security and privacy	***(Understand context and assets)*** Tool should provide the means of establishing security

		requirements	requirements, defining minimum criteria for security in the developed application.
	+SRT005	Abuse cases	
	#SRC002	Perform Application Assessment	
SR003	*SRS004	Perform security and privacy risk assessment	**(*Security Requirements*)** Tool should be able to provide means of examining software design based on cost and regulatory requirements.
	+SRT002	Risk analysis,	
	#SRC003	Capture Security Requirements	
SR004	*SRS008	Use threat modeling	**(*Risk Analysis*)** Tool should be able to provide structured approach to threat scenarios, i.e. identification of security vulnerabilities, and determining risks from these threats, and establishing appropriate mitigation.
	+SRT002	Risk analysis	
	#SRC005	Build Vulnerability Remediation Procedures	
PH3: Architecture and design			
SR005	*SRS005	Establish design requirements	**(*Secure design practices*)** Tool should provide means of adopting and implementing secure design (architecture) techniques.
	#SRC004	Implement Secure Development Practices	
PH4: Test and test results (Verification)			
SR006	*SRS014	Conduct attack surface review	**(*Justify design solution*)** Tool should provide means for risk measurement and trade-off analysis.
	+SRT002	Risk analysis	
	#SRC006	Define and monitor metrics	
PH5: Release			
SR007	*SRS015	Create incident response plan	**(R*esponse*)** Tool should provide means of response plan to address new threats.
	#SRC007	Publish Operational Security Guidelines	

Legend: Requirement ID lookup

*: SDL

+: Seven Touch Points

#: OWASP

1. ***SR001: Making awareness***

   Security requirement "making awareness" plays very important role in security requirements elicitation. It is at the beginning of our means to fulfill requirements also the part of most SSAP's. This step not only gives involved actors, knowledge about security but also provides with basic information like terms, artifacts and methodology. One can learn from the fact that this requirement should be performed at all times, to reduce the chances of mistakes, while following standards and code of conduct will lead to development of secure software.

   In cases when this step is missing or not performed very well, it is possible that different mindset of a team can come-up with different solutions and may not be on the same mindset. That will lead to delays in development cycle and also adds up to the threat of implementing vulnerabilities into software unintentionally. Making awareness can provide team with an opportunity to learn about various forms of vulnerabilities, risks, threats, mitigations, and treatments. Moreover it will also provide an opportunity to update the team with newer risks etc. for example if a team is outdated with the threat of writing vulnerable code by using old technique of encrypted MD5 authentication, and not using salt technique of encryption can compromise privacy. As MD5 encrypted hash can easily be transformed into the static text with online engines available. For reference see (MD5ONLINE, 2014).

2. ***SR002: Understand context and assets***

   Security requirement "understand context and assets" is the basic part for security requirement elicitation. It provides with the understanding of assets involved or at stake during in a particular situation. Understanding context is important because from the context one can start analyzing the scenario, and will be able to perform actions accordingly. In this step goals for the security concerns are mainly jot down, that will be kept until the end of development cycle.

   Loosely implementation of this step can cause several problems, as this step will be the basis for the future steps in these requirements, where the basic concept of security will be defined. In simple words a solid abstract of complete system will be based on context and assets. The context actually is an idea that provides the basis for an event i.e. more the concrete form of idea is, the clear form of details can extracted from it. In some cases the assets are been overlooked because of not been considered as vulnerable to outside world. One example in this can be of dumpster diving as the cases related to this were more often overlooked and/or not considered as threat. A code of conduct should be followed to better understand assets as oppose to assumptions.

3. ***SR003: Security requirements***

   This step "security requirement" involves with the function of identifying elicitation technique, which is important because there are several elicitation techniques available. If in case elicitation technique has not been defined it will always confuse analyst and they will be using various techniques which in end will add up to efforts. Security requirements also include treatment for the particular security requirement. Treatment of these threats will provide with the basic understanding of how the system should be secured in terms of malice activities.

   First and foremost in order to elicit security requirements one needs an elicitation method. Among all available methods which one to choose from? This can be a difficult question. It is wise to ask several questions before choosing an elicitation method. Questions like: what are the actors involved? Who is the user? And what kind of security treatment stakeholders want to achieve? Answers to these questions can provide with the basic understanding of elicitation method. For example if stakeholders need is to achieve secure database, the method of eliciting security requirement could be a structured interviews with stakeholders: see following for reference and more (SQUARE Method, 2014). On the other hand not having an elicitation method can have several side effects like: recording unclear or incomplete security requirement.

4. ***SR004: Risk analysis***

   Security requirement "risk assessment" in this step risks are been specified in details. While risk is the combination of asset, threat, and vulnerability one missing component of vulnerabilities analysis can be performed to fulfill risk assessment. One risk can be composed of several threats and vulnerabilities. So the best way to analyze risk is to point out as much vulnerabilities as possible, and also threats. This will lead to a concrete risk that can be understood very well in the later stages.

   There are three components contributing to the construction of the risk: value of related assets, number of threats possible and number of vulnerabilities. The more detailed these three components are the more concrete risks can be defined. The basic concept of defining concrete risk is to provide with an opportunity to define appropriate treatment. A badly or wrongly stated risk can cause to construct inappropriate treatment or solution that in result will not be able to resolve the actual issue. Moreover it will also add up to time and efforts spent on treatment that actually didn't resolve the problem.

5. ***SR005: Secure design practices***

   In step "secure design practices" architecture identification should be performed according to the treatment needs. For example if the software requires a database then architectural suggestions will be to implement stored procedures. In this step coding standards can also be fixed, as different coding styles may distract developers working on the same project. Moreover security components like defining security mechanism are the part of secure design practices, because security mechanism actually informs in detail on how the security requirement shall be implemented.

There are several possible ways of building software architecture, i.e. secure software design architecture will offer the skeleton of software that will include secure database (implementation of stored procedures, encrypted data, etc.), inheritance/encapsulation of code, and possibility to implement input validation. Not having standard secure software design architecture will always create possibility of writing vulnerable code that could lead to an unwanted incident. Moreover secure architecture will always have the basic security mechanisms already implemented. A good example in this context is of Microsoft's SDL process template (Microsoft, 2014) that enables basic security requirements already implemented.

6. ***SR006: Justify design solution***

   In this step of "justify design solutions" one suggestion is to perform trade-off analysis which will give weight to the software features. The feature with highest weight will be developed first. This will always save time and efforts, and also give chance to decide what features are more important. In this step risk estimations should also be performed because this will weigh the highest risks. And the risks with the highest ranking should be addressed first. It also includes the prioritization of security requirements that will suggest what requirements should be taken first.

   Not having categorized or prioritized security requirements can create unwanted queue of incomplete work items, because work items are mostly related on each other i.e. not having authentication feature in initial stages will be time consuming to implement in later stages, as authentication is responsible to maintain user session. Another possibility of having justified design is to perform calculus of trade-off analysis that can create possibility to reduce unimportant features. This also creates an option of estimating risks that can help in making high risks at the highest priority.

7. ***SR007: Response***

   Step "response" provides with the plan that can be used at times of an incident. Response plan is important because during an incident there will be less time to react. In times of unwanted incident, unavailability of response plan can cause delays in the fix and in some situations can cause failure also.

   Response plan will always be handy when it comes to face unwanted incident. The reason behind is the nature of unwanted event being uncertain in most cases. For example an event that could cause the loss of data, can have a response plan stating steps to recover backed up data.

5.2. Means to fulfill the requirements

In previous chapter we analyzed three security requirements elicitation tools, from which we get to construct means to fulfill those requirements. Figure 5-1 shows all the possible means that have been fulfilled against requirements fulfillment from Table 5-1. The parent boxes shows the actual requirements that we have collected during analysis of SSAP's, under which several means are depicted, we have constructed the means as close to possible functionality available in the tools. However some additional suggestions are also mentioned for example use of secure software architecture, and trade-off analysis etc.

Moreover there could be several other means to fulfill requirements from Table 5-1 however our scope is limited to evaluate the tools that gave us most the requirements blocks fulfilled. The means to fulfill are based on experiments/example done in previous chapters that could also lead to more findings. But due to limited available time to do experiment we have collected as much means as possible from the security engineering tools. We will discuss about the means one by one in detail also, that will give us understanding on how the means/features actually work. Moreover we will determine the measurement scale for these means that will be helpful in rating the tools.

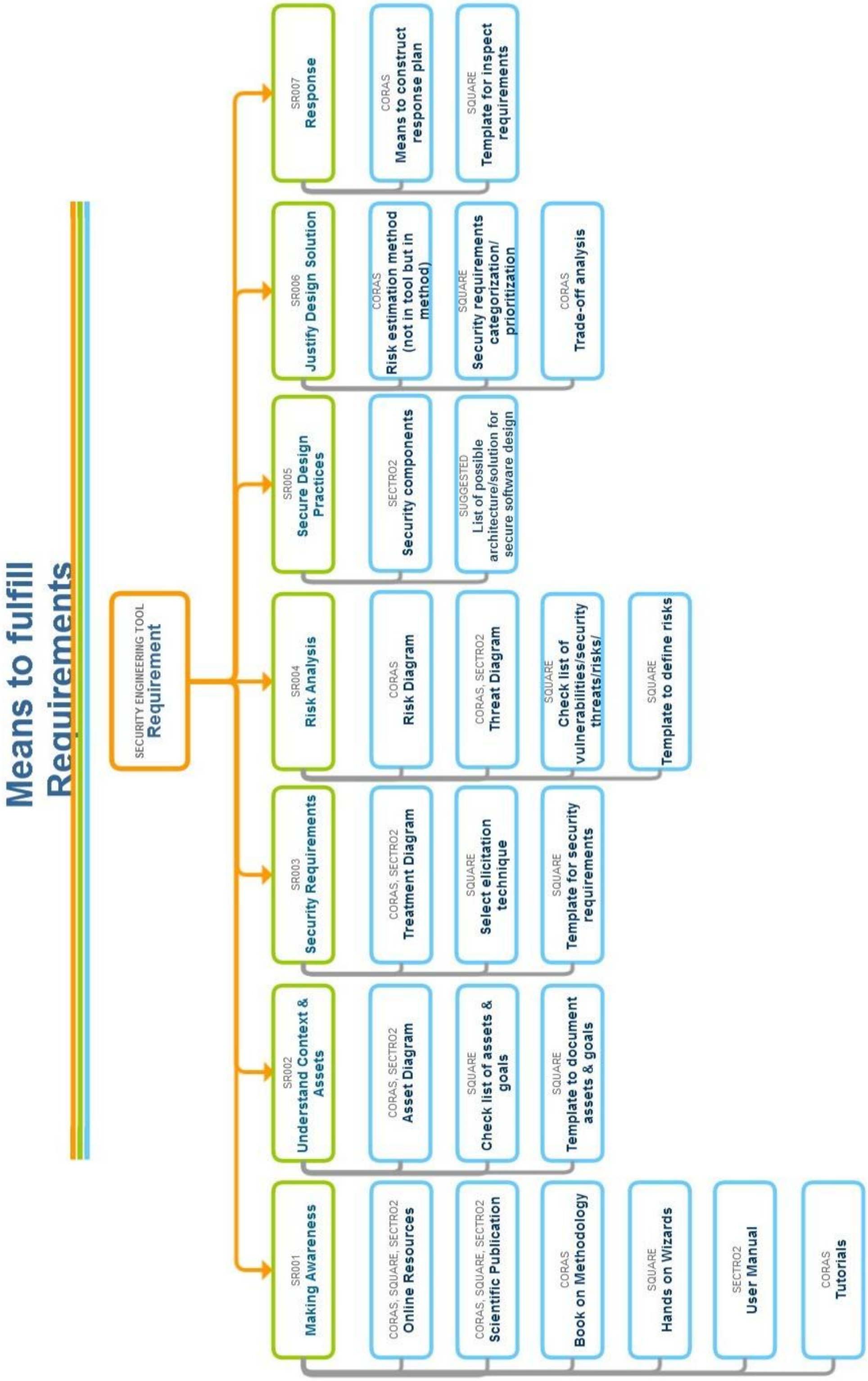


Figure 5-1 Evaluation Framework

➢ **SR001: Making awareness**

- ***Online Resource***

In all analyzed tools one common mean is of availability of online resources, which include the online presence of the tools, procedures related to tool, and documentations. All of these tools are available to download for free, along with documentations which could give security engineering tools an edge to be adapted very easily.

- ***Scientific Publications***

All three tools have adapted a methodology, and have several publications. More related to software security, cloud security, updates to methodology are the most common topic of publications among these tools. Scientific publications can add up to increase in awareness using various aspects of security engineering.

- ***Book on Methodology***

In all three tools only CORAS provide with the book on methodology, however other two have publications for methodology. This gives CORAS tool an edge over other tools, and provide with simplified explanation of methodology in book itself.

- ***Hands on Wizards***

SQUARE tool is built in a way that it provides with several guided wizards for information. However these wizards can also be customized to upload contents related to specific software. That can help in learning the aspects of tools as well as software build.

- ***User Manual***

A user manual is type of dictionary for specific tool, in this case SECTRO2 provides with the manual to support and guide users with artifacts available in tool. It includes abstract definition on how an artifact can be used in particular context. It can be handy in situations where one wants to experience all the available artifacts of the tool. Certainly SECTRO2 has edge over two other tools in this feature.

- ***Tutorials***

A tutorial is a guided plan to support a work process from beginning to the end. In most cases there is only tutorial which kick-start the learning curve really fast. In documentation of CORAS tool there are some tutorials available to teach users with the basics of the tool. This can be handy in situation where stakeholders want to start working on the tool as soon as possible. Unlike user manual tutorial in most cases can teach faster. CORAS tool has edge over other tools in this context.

➢ SR002: Understand context and assets

- ***Asset Diagram***

An asset diagram is to identify assets related to the software security. It is crucial to have clear definitions of assets that can help in defining the concrete goals for software security. Not having asset diagram can lead to ambiguity in valuable assets and in some cases can lead to overlooking of some valuable assets. CORAS & SECTRO2 provide with the asset diagram which gives these two tools edge over SQURE. But also SQUARE tool have means to define assets that can be taken as substitute for the diagram.

- ***Checklist of Assets & Goals***

In SQUARE tool there is way to define assets and goals, in tables. That can be handy in cases of definitions for these two aspects of software security. As this tool does not include any diagrams, the check lists and tables are the means to define assets and goals.

- ***Template to Document Assets & Goals***

SQUARE tool also provide with the predefined common assets and goals in common software that can save time via avoid writing additional assets and goals. The template however can be customized and molded according to the current software security needs.

➢ SR003: Security requirements

- ***Treatment Diagram***

A treatment diagram provide with the feature of defining treatments to the possible vulnerabilities, unwanted incident, and risks. CORAS and SECTRO2 provide with the functionality of depicting treatments in tools with the help of available artifacts for treatment. One can create security requirements as treatments to the possible vulnerabilities and risks. CORAS and SECTRO2 has an edge over SQURE tool in this aspect of security engineering.

- ***Select Elicitation Technique***

Choosing elicitation technique from several available techniques can sometimes be a difficult process. SQUARE tool provide with the mechanism to overcome this difficulty by adding a questioner that can suggest the elicitation according to the stakeholder needs. This gives SQUARE tool and edge over two other tools.

- ***Template for Security Requirements***

As there are no diagrams in SQUARE tool, again the way of capturing security requirements is done using the template. The template in SQUARE tool provides with the most common security requirements already built-in. That in return can save time and efforts to capture many additional requirements.

- **SR004: Risk analysis**
  - ***Risk Diagram***

A risk diagram is to depict the associated risk with the software. This process of depicting risks is handy because it provides with the basic understating of the common risks, which could lead to writing concrete threats and vulnerabilities. CORAS provides with the means to depict risks inside the tool using available artifacts. That gives this tool an advantage over other two.

- ***Threat Diagram***

A threat diagram is to define details of several threats that are associated with the particular risk. It provides the understanding of the malice actors also, that are outside the system. In case of uncertain or defining unclear threats could lead to bad description of the risk or also can make a risk that is not concrete enough. CORAS and SECTRO2 tools have this feature inbuilt to depict threats in diagrams, which gives these tools advantage over SQUARE tool.

- ***Checklist of Vulnerabilities/Security threats/Risks***

SQUARE tool also provide with the means of defining vulnerabilities, threats, and risks in terms of checklist. That can be in some cases an easy access because a risk is composed of vulnerabilities, threats and assets, and in SQUARE tool these all are associated also.

- ***Template to Define Risks***

SQUARE tool also provide with the means of pre-defined risks in expression of a template this contains most common types of risks. A template with already available risks can be handy because this can save time and efforts for the team.

- **SR005: Secure design practices**
  - ***Security Components***

A security component is a method to define security mechanism. This kind of approach can provide with the possibility to define how to deal with the particular security requirement. SECTRO2 provides with the possibility to create security components and depict them in details for better understanding.

- ***List of Possible Architecture/Solution for Secure Software Design***

A type of software architecture can be a crucial step in making secure software, because a design of software decides the security level of the particular software. Possibility to choose from different software architecture style can resolve several security related problems from the beginning. However this feature is not available in any of the tool, but the idea is to implement a questioner like in "choosing elicitation technique" feature, and answer to those questions will suggest which type of software design architecture should best suite with the current scenario. This will in end save time and efforts applied to choosing one software architecture style.

- **SR006: Justify design solution**
  - ***Risk Estimation Method***

A risk estimation method provides with the possibility to calculate the severity of particular risk. And based on this analysts can decide which risk should be addressed at foremost. This can also give an opportunity to decide whether the risk is relevant in current context or not. Having possibility to calculate risk estimation inside the tool can be very handy, but current tools does not provide any functionality that addresses this issue. Only CORAS methodology suggests some calculus for likelihood calculations that can rate risks accordingly.

- ***Security Requirements Categorization/Prioritization***

One aspect of justified design is to give clarifications for the particular security requirements. This can be achieved by categorizing and prioritizing security requirements based on severity or likelihood. SQAURE tool gives an opportunity to deal with this solution by providing the table that includes categorization and prioritization of security requirements. This gives an edge to SQUARE tool on other two.

- ***Trade-off Analysis***

The function of trade-off analysis is to decide whether a particular feature will add up a value to the software security. For example what features can be given up if the time duration for the software development is limited. Trade-off analysis can also be handy in situations where budget allotted to the project is limited and features that should be implemented are more than the budget. This feature is not available in any of the tools that we have analyzed, however having such functionality can help in decision making.

- **SR007: Response**
  - ***Means to Construct Response Plan***

A response plan is to provide solutions for unwanted incident. CORAS tool provide with this feature to depict a response plan also using the existing artifacts. However tool does not suggest such type of diagram but one can create a response plan using available artifacts. This will always provide analyst with the solution or steps to do in situation of an incident.

- ***Template for Inspect Requirements***

Inspection of security requirements can actually provide with an opportunity to improve future security requirements. As old security requirements will become obsolete in future, new security requirements can be improved by learning lessons from old ones. An inspection can also provide with the opportunity to verify if the requirement is actually fulfilled or not. SQUARE tool has an edge over other tools in this aspect.

5.3. Measurement Scale for tool Analysis

Based on the research done in (Matulevičius, et al., 2009) that discuss about QualOSS quality model that for developing a systematic quality model to assess robustness and scale of evolution of the OSS (open source software). In similar way we will try to give scale to means that we have discovered in order to provide an opportunity to measure the tool reliability. Below we have developed scaling. However in later discoveries, the scale will be based on the importance of the feature also.

In order to calculate the total measure of a particular tool one can answer to the sample questions given in Table 5-3, Table 5-4, Table 5-5, Table 5-6, Table 5-7, Table 5-8, and Table 5-9. For which we can then be able to calculate the rating of tool. The rating goes like from “no support” possesses 0 point and “full support” possesses 3 points and all others accordingly. The rating can also be seen in Table 5-2. The features priority is important also, as some tools support interactive methods to create diagrams, whereas some tools provides with the similar means but with no interactive diagrams. We prioritize features according to their importance in security engineering tools and also according to the ease of use.

The rating will then be multiplied with the feature priority to get the score for particular question. And then we can sum them to get the total score for the requirement. Finally sum of all the score’s obtained from requirements will be taken as the final score for that tool.

Table 5-2 Measurement Legend

Feature satisfaction	Description	Rating
Full Support	The feature is fully supported with no complications at all. And results gained by using particular feature are complete, concrete, and satisfactory.	3
Above Average Support	The feature is partially supported, indicating that there could be some additions made to enable full support.	2
Minimal Support	The feature is supported to the extent that it only fulfills the criteria of availability, which means it does not satisfies and simplifies the process.	1
No Support	The feature is not supported at all.	0

Table 5-3 Likert Scale for SR001-Making Awareness

Sample questions	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support
To which extent this tool provides online support, e.g. downloads and documentation?	I/6	☐	☐	☐	☐
To what extent this tools' book on methodology is explained and/or to what extent this tool adapts the methodology?	II/5	☐	☐	☐	☐
To what extent this tool provides support for the user manual?	III/4	☐	☐	☐	☐
To what extent this tool provides tutorials? That will support and provide learning opportunity.	IV/3	☐	☐	☐	☐
To what extent this tool has gained popularity to support and provide several publications?	V/2	☐	☐	☐	☐
To what extent this tool provides support for hands on wizards? These will guide engineers through several aspects of tool?	VI/1	☐	☐	☐	☐

Table 5-4 Likert Scale for SR002-Understand Context and Assets

Sample questions	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support
To which extent this tool supports the feature to create asset diagram, or at-least include a feature similar to declare assets in efficient way?	I/3	☐	☐	☐	☐
To what extent this tool supports the feature that includes a template to document assets and goals?	II/2	☐	☐	☐	☐
To which extent this tool supports the feature to create checklist of assets and goals?	III/1	☐	☐	☐	☐

Table 5-5 Likert Scale for SR003-Security Requirements

Sample questions	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support
To which extent this tool supports the feature to create treatment diagram, or at-least include a feature similar to declare treatments in efficient way?	I/3	☐	☐	☐	☐
To what extent this tool supports the feature that can help selecting security requirements elicitation techniques?	II/2	☐	☐	☐	☐

To what extent this tool supports template to document security requirements?	III/1	□	□	□	□

Table 5-6 Likert Scale for SR004-Risk Analysis

Sample questions	**Priority /Score**	**Full Support**	**Above Average Support**	**Minimal Support**	**No Support**
To which extent this tool supports feature to create risk diagram, or at-least include a feature similar to declare risks in efficient way?	I/4	□	□	□	□
To which extent this tool supports feature to create threat diagram, or at-least include a feature similar to declare threats in efficient way?	II/3	□	□	□	□
To what extent this tool supports template to document risks?	III/2	□	□	□	□
To what extent this tool supports feature to create checklist of vulnerabilities, security threats and risks?	IV/1	□	□	□	□

Table 5-7 Likert Scale for SR005-Secure Design Practices

Sample questions	**Priority /Score**	**Full Support**	**Above Average Support**	**Minimal Support**	**No Support**
To which extent this tool supports the feature to create security mechanism, or at-least include a feature similar to declare security mechanism in efficient way?	I/2	□	□	□	□
To what extent this tool supports the feature that can help selecting secure software architecture and secure software design?	II/1	□	□	□	□

Table 5-8 Likert Scale for SR006-Justify Design Solution

Sample questions	**Priority /Score**	**Full Support**	**Above Average Support**	**Minimal Support**	**No Support**
To which extent this tool supports feature that can help calculating risk estimations?	I/3	□	□	□	□
To what extent this tool supports feature that can help categorizing and prioritizing security requirements?	II/2	□	□	□	□
To what extent this tool supports mechanism to estimate trade-off analysis?	III/1	□	□	□	□

Table 5-9 Likert Scale for SR007-Response

Sample questions	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support
To what extent this tool supports feature that can help creating response plan?	I/2	□	□	□	□
To what extent this tool supports feature to create templates for security requirements inspection?	II/1	□	□	□	□

5.4. Use of Evaluation Framework

The means to fulfill requirements can be used as a framework if someone wants to evaluate security requirements engineering tool. This can help in specifying the best tool, on the basis of fulfillments. The audience of this framework shall be software architects and analysts, who are responsible for making requirements elicitation based decisions. There are four steps in choosing best SRE tool, step one is to refer to initialize requirements, step two is searching for the available tools for SRE. In step three analysts can evaluate tool one by one and specify available features in correspondence with the evaluation framework and rate them. In step four analysts can choose the tool with the highest ratings.

Use of this framework is depicted in Figure 5-2 in which we have made a simple process to evaluate security requirements engineering tool. When analyst gets instruction to decide on the SRE tool, he will first initiate the requirements, and list down all the fulfillments that should be available in a particular SRE tool. Then he can start searching for the available SRE tools mostly it comes from online source i.e. internet research. The next part can be time consuming and possess the actual value in use of the evaluation framework, in this step analyst has to use the tool one by one and rate them according to the features listed from step one. The next step is simple analyst only have to choose the best tool based on highest rating. Finally analyst has the tool and decision has been made.

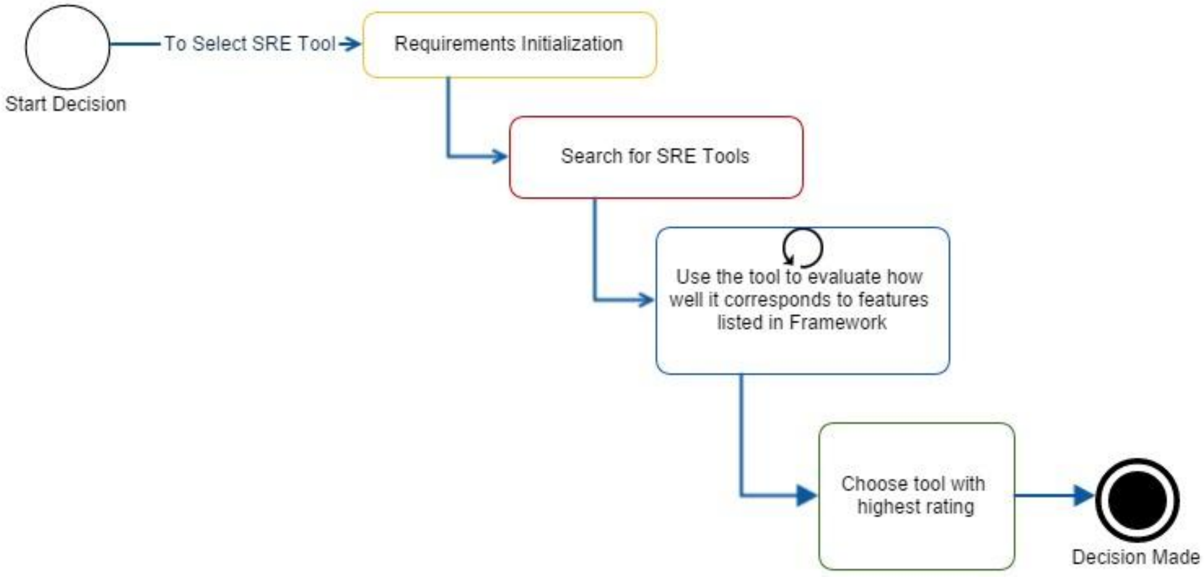

Start Decision
To Select SRE Tool
Requirements Initialization
Search for SRE Tools
Use the tool to evaluate how well it corresponds to features listed in Framework
Choose tool with highest rating
Decision Made

Figure 5-2 Use of Evaluation Framework

- ***STEP1: Requirements Initialization***

In this step user can analyze available steps in SRE tool evaluation framework, which will then be used as a reference for features to look for in a SRE tool. The importance of this step is of getting startup information about security engineering tools. For example: in order to find out the SRE tool the evaluation framework will help point out the major processes and features involved. This can make a clear distinction in the tool from others. One can use this step when decision to search for SRE tool has made, but properties of SRE tool are unknown.

- ***STEP2: Search for SRE tools***

This step is also important because searching for available tools can be a tedious process, however by finishing the first step user will be able to get some ideas about tool functionalities that can be useful to distinguish between SRE tools and RE tools. This step might take lot of time because of changing trends in requirement engineering technology. The major means for searching SRE tools include internet search, contact and references, and previous knowledge.

- ***STEP3: Use the tool to evaluate how well it corresponds to features listed in framework***

This step plays as an engine for the work flow in use of evaluation framework, because in this step user has to use tools and rate the features by answering questioners. The example questioners can be seen from Table 5-3 to Table 5-9. After rating the available features, it then will be multiplied by with the priority. Finally that particular tool's section score should be summed to get the total score. This step is regressive also because during analysis there should be more than one tool in the list to analyze.

- ***STEP4: Choose the tool with highest rating***

In this step user only have to choose the tool with highest ratings that were achieved from previous step. In order to determine which tool would correspond and fulfill the features efficiently the score should be made as precise as possible.

5.5. Chapter Summary

In this chapter we have constructed the means to fulfill SSAP's requirements using security engineering tools in other words evaluation framework for choosing security engineering tool that can be visited in Figure 5-1. Along with that we have created measurement for the tool, to evaluate the tool according to features. Finally we have discussed about the use of evaluation framework, and discusses on how to use this framework that can be visited in Figure 5-2.

Coming up next we will test this framework using one instrument, and conclude this research work.

6

Tool Assessment

In this chapter we will describe how tools were evaluated using two different frameworks, along with its application, and results. Rating the tool is an essential part of this research because it will allow users to choose between security engineering tools efficiently. The motive is to rate the tools according to their features reliability, completeness and ease of use. In this regards we have chosen five tools that include three previous tools that have been discussed in chapter 4. Additional two tool that we are going to use in this experiment are STS-Tool (Arcsin, 2014) and Magic Draw (No Magic, 2014). In this regards we will use RE-Tool evaluation approach (R-TEA) framework (Matulevičius, 2005) and framework created in previous chapter, security engineering tool evaluation framework (SETEF). Complete results of these tools and their score can be visited in appendix-B and appendix-C.

6.1. Design of Experiment

The experiment is designed in order to complete the tool evaluation using two different frameworks i.e. R-TEA (Matulevičius, 2005) and SETEF. There are four steps in this experiment, 1. Define requirements of experiment, 2. Assess tools using R-TEA (Matulevičius, 2005), 3. Assess tools using SETEF, and 4. Compare results. See Figure 6-1

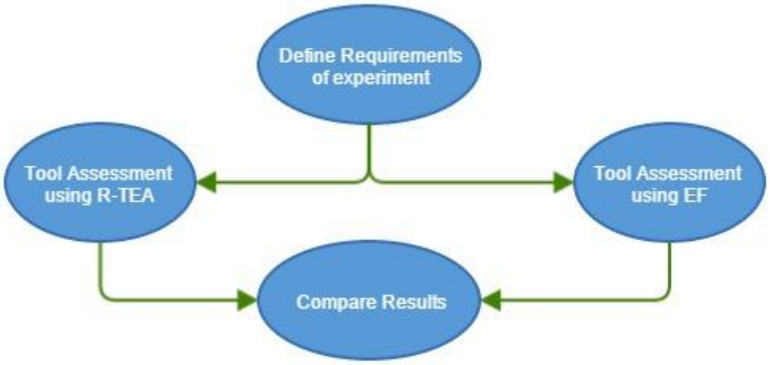

Define Requirements of experiment
Tool Assessment using R-TEA
Tool Assessment using EF
Compare Results

Figure 6-1 Design of Experiment

6.2. Assessment of the tools using R-TEA approach

In assessment of the tools using R-TEA approach, there are several steps however we have chosen the most relevant/modified steps to this research work. Below are the steps involved in R-TEA approach precisely discussed.

6.2.1. Preparation of requirement specification

This step involves gathering all the requirements according to the environmental needs, prioritizing these requirements according to importance of the feature, and finally based on elicitation and prioritization results team prepares the requirements specifications.

6.2.2. Selection of business parties

This step involves the investigation of the requirement engineering tool market according to external requirements. The evaluation team requests trail and demonstration RE-Tool version from the business parties (Matulevičius, 2005).

6.2.3. Investigation of the tools

This step involves analyzing the security engineering tools by using them. Rating is according to the features presence, ease of use, and completeness. These tools can be rated according to their features, and by summing all will give the total number for that particular tool.

6.2.4. Decision

Tool decision is based on their ranking, how good their score was in previous step of investigation, and will be chosen as the best tool.

6.3. Assessment of the tools using SETEF

6.3.1. Requirements Initialization

This step is primary in analyzing tools using evaluation framework that is to collect all the requirements for the tool, in our case using available requirements and prioritizing them according to the need for feature. The process of prioritizing requirement is pretty simple just using requirements backlog one can identify the needs and from that an analysts will be able to identify what requirements for the tool should be at the highest priority.

Priority given to features in our questioners is based on the ease of use, clarity, completeness and effectiveness of the feature. At first we start by giving score to individual feature comparing with other features in a particular requirement. And make the high priority feature a level up, which eventually sorted all the features according to their level of importance.

6.3.2. Search for Tools

Search for the tool is mostly based on internet search, in our case we have kept using Google search engine to find relevant security engineering tools. As security engineering tools are still not much popular, we were able to find 5 relevant tools. This process can be easier because of less available tools currently. And also because security engineering tools are not yet widely used, search makes it simple to find them.

6.3.3. Tool Evaluation

In this step an evaluator has to use the tools in order to rate the features available in the tool. We have prioritized the features in step one already, which allows us to move forward to rate the features according to their availability, ease of use, and completeness. The evaluation was based on following an example which was provided in a tutorial by (Lund, et al., 2011) where a case of telemedicine was provided. Based on the tutorial we have constructed diagrams to analyze the features. And finally we have rated these features accordingly. Below are the analyses of tools we have used to analyze different set of tools.

6.3.3.1. STS Tool

STS-tool in this list has the most importance because this tool is unique from the tools we have analyzed so far, moreover this tool is security requirements gathering specific also. There are few artifacts in this tool, which enables most of the diagrams, from security risk to asset diagrams. These artifacts are agent, role, goal, document, and event. However this tool does not have complete set of artifacts to support all the features of security engineering tools. As the tool follows standard language for security requirements elicitation, which suggests fewer artifacts, this also makes the tool complete according to its method. In our assessment of tools, this tool gained 113 by R-TEA method and 80 by our method. See appendix-B and C for full results.

6.3.3.2. CORAS Tool

CORAS tool so far have the best score calculated using evaluation framework. That’s because the tool have most of the security engineering features at its best. Risk, asset, and treatment diagrams can easily be constructed, along with the proper naming for the artifacts. Moreover this tool provides with the likelihood status also, which is helpful in prioritizing requirements. However the tool does not correspond to same score via R-TEA approach. This tool has the base score of 99 using R-TEA and 109 according to SETEF calculations. See Appendix-B and C for full results.

6.3.3.3. SQUARE Tool

SQUARE tool follows different approach than all other tools it has forms, templates, wizards etc. instead of diagram. The tool gained highest score of 167 using R-TEA approach, but got least score of 75 using SETEF. According to SETEF this tool gained fewer score because we have prioritized the interactive features at high, and SQUARE tool do not have diagrams.

6.3.3.4. SecTro2 Tool

SecTro2 tool analysis is composed of different views, and inside these views, diagrams are sub-grouped to distinguish between threats and their treatments. A view can be created for the organization and inside this organization, goals, assets, and threats are declared. And inside any of these threats, treatment can be viewed. Solely well-structured tool for security engineering, which gained score of 102 according to R-TEA and 80 based on evaluation framework calculations. See Appendix-B and C for full results.

6.3.3.5. Magic Draw Tool

The purpose of Magic Draw is not for security engineering, however as the tool support UML extensively, historically this tool have been in use for every other requirement elicitations. The base for this tool is it allows users to create class, use case, activity diagrams which can be transformed to cater security engineering artifacts. For example one can create misuse-case diagram to declare threats, just by using use case diagram. Moreover this tool supports extensive variety of diagrams which gives this tool score of 110 based on R-TEA approach and 99 based on evaluation framework calculations. See Appendix-B and C for full results.

6.3.4. Select the Best Tool

The selection of best tool is based on the score they acquired from the analysis done using evaluation framework. One can also change the priorities to maintain the required features for the particular project. However one can also use priorities available in this research work, because priorities for features in use of evaluation framework are calculated based on their ease of use. The criteria for the best tool will always be the one with the highest score.

6.4. Comparison of R-TEA and SETEF

These two methods have similar nature in analyzing the security engineering tools. Where R-TEA approach is addressed towards requirements engineering tools and SETEF focuses on sub-category security requirements engineering tools. However these two methods follow different approach to analyze the tools, which effects the results obtained in the previous evaluation. R-TEA approach is more focused towards language and complete specification of the requirements and SETEF declares interactive diagrams and clear specifications gathering as the primary objective. In previous evaluation of the tools, we these two methods followed some steps that can be seen in Table 6-1.

Table 6-1 R-TEA and SETEF Steps

Steps	R-TEA	SETEF
1	Preparation of the requirements specification	Requirement initialization
2	Selection of the business parties	Search for tools
3	Investigation of the tools	Tool Evaluation
4	Decision	Best tool selection

As oppose to SETEF requirements mentioned in Table 5-1 R-TEA suggests set of activities mentioned in Table 6-2. However these two cannot be compared because of differences, but few activities like (requirements must have a unique identifier) can be seen are common. While both approaches are addressing different aspects of requirements engineering, most of the activities in both are diverse. This gives us a strong reason that results obtained are oppose to each other, see Table 6-3.

Table 6-2 Activities of R-TEA > Adapted from (Matulevičius, 2005)

Dimensions	Activities in R-TEA
Representation dimension	
1.1	Specify uniquely identifiable description using informal language
1.2	Specify requirements using semi-formal language
1.3	Specify requirements using formal language
1.4	Define traceable associations between requirements and the different elements of requirements specification
1.5	Connect seamlessly with other tools and systems, by supporting interoperable protocols and standards
Agreement dimension	
2.1	Maintain an audit trail of changes, archive baseline version; and engage a mechanism to authenticate and approve change requests
2.2	Classify requirements into logical user defined groupings
2.3	Support secure, concurrent cooperative work between members of a multidisciplinary team, which may be geographically distributes
2.4	Maintain a comprehensive data dictionary of all project components and requirements in a shared repository
Specification dimension	
3.1	Collect and store a common system's and a product family's domain requirements
3.2	Generate predefined and ad hoc reports, documents that comply with standard industrial templates, with support for presentation-quality output and in-built document quality controls
3.3	Generate the complete specification, expressed using formal language (informal and semiformal languages might also be included), commonly agreed by all stakeholders

6.5. Threats to Validity

This experiment was conducted on the basis of personal experience gained using security engineering tool. The features were prioritized solely with the help of brainstorming and this may differ according to difference in needs or difference in experience. Most of the features in SETEF were derived using three security engineering tools, which compromises the future security engineering tools that may have different features.

Additionally the tools were compared against requirements collected from SSAP's in order to build the SETEF, where some of the requirements were out of scope because of limitations in capabilities in security engineering tools. For example requirements related to secure code, were neglected because security engineering tools do not support code related to requirements.

6.6. Summary

In Table 6-3 the comparison of both approaches is showing the result opposite from each other. The reason for this kind of behavior could be the difference in features. As R-TEA approach is more focused towards correct documentation of requirements and SETEF approach is more focused towards features instructiveness, the results will be opposite. Highest score is gained by SQUARE tool using R-TEA approach, and CORAS using SETEF approach.

In this chapter we have explained how the experiment was conducted using evaluation framework, and concluded that CORAS gained the highest points which make this tool to be chosen according to SETEF approach. See Table 6-4 for full tool evaluation results. In next chapter we will conclude this research work.

Table 6-3 Framework Results Comparison

Tool	R-TEA
CORAS	99
SecTro2	102
Magic Draw	110
STS-TOOL	113
SQUARE	167

Tool	SETEF
SQUARE	75
STS-TOOL	80
SecTro2	80
Magic Draw	99
CORAS	109

Table 6-4 Tool Evaluation Summary

Tool	R-TEA	SETEF
STS-TOOL	113	80
CORAS	99	109
SQUARE	167	75
SecTro2	102	80
Magic Draw	110	99

7

Conclusion and Future Work

This research work was done to create a method/framework, which helps in choosing best security engineering tool. In this regards SSAP's (SDL, 7 Touch points, and OWASP) were analyzed to gather common requirements secure software. These requirements were then tested against security engineering tools (CORAS, SecTro2 and SQUARE) to collect means to fulfill these requirements. These means however can be many but our focus was to collect most of which from the well-developed security engineering tools. These means were taken as the base for Security engineering tool evaluation framework (SETEF). SETEF can be used to evaluate the security engineering tools, and one can choose between tools and select the best one based on highest score.

In conclusion if someone wants to evaluate security engineering tools using SETEF approach they might get different result based on the priorities they give to the features. But most appropriate results can only be obtained based on features an individual wants utmost. Moreover the features that we have elicited using SSAP's have a tendency to change with the changing needs of the software security requirements. This also leaves the room for future development in this field.

In this research work we also concluded that results using R-TEA approach were opposite to SETEF because R-TEA framework is more focused towards documentation part of requirements engineering. And in this research SQUARE tool is more focused towards documentation, which gives this tool the highest score using R-TEA approach. However using SETEF approach CORAS tool gets the highest score. Because using SETEF we have scored the features according to their interactive diagram and visualization. One can also score the features according to their priorities, but experiment conducted in previous chapter we have prioritized the features based on visualization.

7.1. Limitations

In this research we have evaluated the tools on the basis of priorities given by personal opinion and judgment, however these values can differ according to different analysts. These priorities are not concrete enough to provide actual results of the tool evaluation, which also leaves the room for future development in this perspective. Our conclusion was based on scores obtained by analyzing tools and calculating the base score from each activity, for which this also can be done using different statistical methods like: calculating the mean of priority features.

One more addition to this work can be to make priorities more solid at the requirements level. As we have prioritized these features, researchers also can priorities requirements itself. For example how much value does the requirement SR001 contains over SR002.

7.2. Future Work

This framework contain requirements analyzed from SSAP's, however these requirements can also be gathered by market research and following the current trend in security requirements gathering techniques, currently used within most of the companies. Asking questions like "what an analyst look for if he has to identify the assets" the answer to this question can be variant and diverse based on the expertise the market researcher has. And can provide various ides about techniques used in identifying assets. As oppose to task done in this thesis work, one can also try to analyze the development processes to analyze security constraints available. That can provide with various other terms to cater in regards to analyzing security engineering tools.

BIBLIOGRAPHY

Addison-Wesley Software Security Series. 2006. Seven Touch Points for Software Security. *buildingsecurityin.com.* [Online] Addison-Wesley, 2006. [Cited: 03 26, 2014.] http://buildingsecurityin.com/concepts/touchpoints/.

Altuhhova, Olga, Ahmed, Naveed and Matulevičius, Raimundas. 2013. *An Extension of Business Process Model and Notation for Security Risk Management.* Tartu : Institute of Computer Science, University of Tarut, 2013.

Anderson, J. Ross. 2001. *Security Engieering.* Cambridge : Wiley, 2001. 978-0470068526.

Arcsin. 2014. *STS-TOOL.* [Internet] Italy : Arcsin, 2014.

Basin, David, et al. 2009. *Model-Driven Development of Security-Aware GUI's for Data-Centric Applications.* Spain : ETH Zurich, Switzerland; IMDEA Software Institute, Madrid, Spain; Universidad Computense, Madrid, Spain, 2009.

Braz, Fabricio A., Fernandez, Eduardo B. and VanHilst, Michael. 2008. *Eliciting Security Requirements through Misuse Activities.* Florida : IEEE, 2008.

Bresciani, Paolo, et al. 2004. *Tropos: An Agent-Oriented Software Development.* Italy : Kluwer Academic Publishers, 2004.

CORAS. 2014. The CORAS Method. *coras.sourceforge.net.* [Online] CORAS, 2014. [Cited: 05 17, 2014.] http://coras.sourceforge.net/.

Crook, Robert, et al. 2002. *Security Requirements Engineering: When Anti-requirements Hit the Fan.* Milton Keynes : IEEE, 2002. 1090-705X.

Dictionary.com. 2014. Definition. *Dictionary.com.* [Online] 2014. [Cited: 04 08, 2014.] http://dictionary.reference.com.

Ganguly, Aroop. 2011. P-SQUARE Tool - Quick Demo. [Online] 2011. [Cited: 9 16, 2014.] http://www.youtube.com/watch?v=zobdTHjcDjc.

Howard, Michael and Lipner, Steve. 2006. *The Security Development Life Cycle.* Redmond : Microsoft Press, 2006. 978-0735622142.

Institute of Electrical and Electronic Engineers. 1990. *IEEE Standard Glossary of Software Engineering Terminology.* New York : IEEE, 1990. 610.12-1990.

ISO. 2014. ISO/IEC 27001 - Information security management. *iso.org.* [Online] International Organization for Standardization, 2014. [Cited: 05 23, 2014.] http://www.iso.org/iso/home/standards/management-standards/iso27001.htm.

Jürjens, Jan. 2001. *Modelling Audit Security For Smart-Card Payment Schemes with UML-SEC.* London : Springer US, 2001. 978-0-306-46998-5.

Lund, Mass Soldal, Solhaug, Bjornar and Stolen, Ketil. 2011. *Model Driven Risk Analysis The CORAS Approach.* Heidelberg : Springer Heidelberg Dordrecht, 2011.

Matulevičius, Raimundas. 2005. *Process Support for Requirements Engineering.* Trondheim : NTNU-trykk, 2005. 82-471-7171-6.

Matulevičius, Raimundas, Kamseu, Flora and Habra, Naji. 2009. *Measuring Open Source Documentation Availablity.* Namur : University of Namur, 2009.

Mayer, Nicolas. 2009. *Model-based Maagement of Information System Security Risk .* Namur : Presses uniersitaires de Namur, 2009. 978-2-87037-640-9.

McGraw, Gary. 2003. *Software Security.* Dulles : Cigital, 2003.

MD5ONLINE. 2014. MD5 Decrypter. *md5online.org.* [Online] Md5online, 2014. [Cited: 08 06, 2014.] http://www.md5online.org/.

Mead, Nancy R, Hough, Eric D and Stehney II, Theodore R. 2005. *Security Quality Requiremets Engineerig (SQUARE) Methodology.* Pittsburgh : Carnegie Mellon Uniersity, 2005. PA, 15213-3890.

Merriam-Webster. 2014. Life Cycle. *Merriam-Webster.com.* [Online] 2014. [Cited: 04 12, 2014.] http://www.merriam-webster.com/dictionary/life%20cycle.

Microsoft. 2014. SDL Process Template. *microsoft.com.* [Online] 2014. [Cited: 08 06, 2014.] http://www.microsoft.com/security/sdl/adopt/processtemplate.aspx.

Microsoft. 2014. SDL-Agile Example. *msdn.microsoftcom.* [Online] Microsoft, 2014. [Cited: 04 12, 2014.] http://msdn.microsoft.com/en-us/library/windows/desktop/ee790619.aspx.

Microsoft.. 2010. Simplified Implementation of the Microsoft SDL. *microsoft.com.* [Online] 11 04, 2010. [Cited: 03 26, 2014.] http://www.microsoft.com/en-us/download/details.aspx?id=12379.

Mochal, Tom. 2008. 10 techniques for gathering requirements. *techrepublic.com.* [Online] CBS Interactive, 01 02, 2008. [Cited: 06 01, 2014.] http://www.techrepublic.com/blog/10-things/10-techniques-for-gathering-requirements/.

Mouratidis, Haralambos, et al. 2008. *Adapting Secure Tropos for Security Risk Management during Early Pahses of the Information System Development.* London : Springer, 2008. 10.1007/978-3-540-69534-9_40.

No Magic. 2014. *Magic Draw.* [Internet] Texas : No Magic, 2014.

Open software security community. 2014. CLASP Best Practice. *owasp.org.* [Online] open software security community, 2014. [Cited: 04 15, 2014.] https://www.owasp.org/index.php/Category:CLASP_Best_Practice.

OWASP. 2014. OWASP CLASP Project. *owasp.org.* [Online] OWASP, 2014. [Cited: 05 11, 2014.] https://www.owasp.org/index.php/Category:OWASP_CLASP_Project.

Oxford Dictionaries. 2014. Artifact. *oxforddictionaries.com.* [Online] 2014. [Cited: 04 12, 2014.] http://www.oxforddictionaries.com/us/definition/american_english/artifact.

Ragunath, P.K, et al. 2010. *Evolving A New Model (SDLC Model-2010) For Software Development Life Cycle (SDLC).* Chennai : IJCSNS International Journal of Computer Science and etwork Securit, 2010.

Refsdal, Atle. 2014. Analysing Risk in Practice: The CORAS Approach to Model-Driven Risk Analysis. *http://coras.sourceforge.net/.* [Online] 2014. [Cited: 06 21, 2014.] http://coras.sourceforge.net/newsarchive.html.

Science Daily. 2014. Security Engineering. *sciencedaily.com.* [Online] Science Daily, 2014. [Cited: 05 17, 2014.]

Secure Tropos. 2014. Secure Tropos. *securetopos.org.* [Online] Secure Tropos, 2014. [Cited: 06 22, 2014.] http://securetropos.org/.

Sindre, Guttorm and L.Opdahl, Andreas. 2005. *Eliciting Security Requirements with Misuse cases.* London : Springer-Verlag London Limited, 2005. DOI 10.1007/s00766-004-0194-4.

Sindre, Guttorm. 2006. *Mal-Activity Diagram for Capturing Attacks on Business Processes.* Trondheim : Norwegian University of Science and Technology, 2006.

Sommerville, Ian. 2004. *Software Engieering 7th Edition.* Massachusetts : Addison Wesley, 2004. 0321210263.

SQUARE Method. 2014. SQUARE Method. *sei.cmu.edu.* [Online] 2014. [Cited: 08 06, 2014.] http://www.sei.cmu.edu/reports/05tr009.pdf.

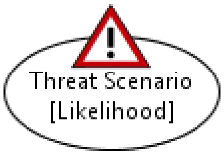
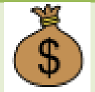
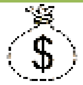
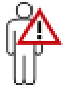
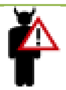
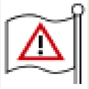
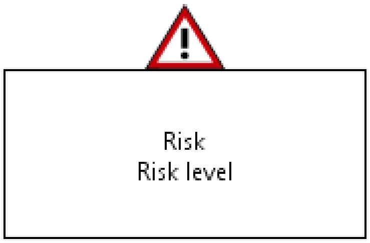
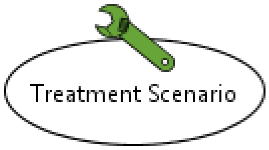
T. Devanbu, Premkumar and Stubblebine, Stuart. 2000. *Software Engineering For Security: A Roadmap.* New York : ICSE 2000, 2000.

T. Ross, Douglas and E. Schoman, Kenneth. 1977. *Structured Analysis for Requirements Definition.* Massachusetts : IEEE, 1977.

TechTarget. 2014. Software Requirements Specificaiton . *TechTarget.com.* [Online] 2014. [Cited: 05 18, 2014.] http://searchsoftwarequality.techtarget.com/definition/software-requirements-specification.

APPENDIX-A

Table 9-1 CORAS Tool Artifacts

	Category	Name	Image
CORAS TOOL ARTIFACTS	Connections	Harm	⟶
		Impact	⟶
		Initiates	⟶
		Leads To	⟶
		Threats	⟶
		Vulnerability Target	——
		Legal Norm Target	——
	Basic CORAS	Threat Scenario	
		Direct Asset	
		Indirect Asset	
		Human Threat Accidental	
		Human Threat Deliberate	
		Non-Human Threat	
		Risk	
		Treatment Scenario	

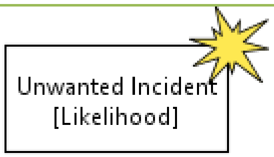
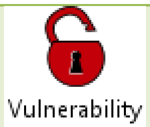
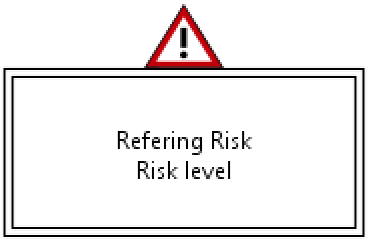
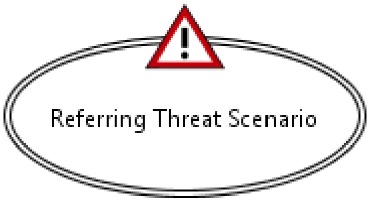
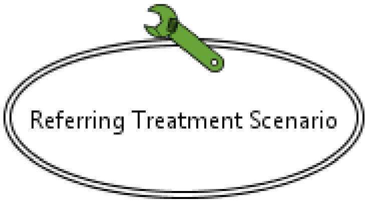
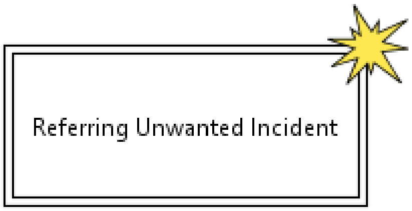
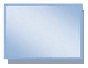
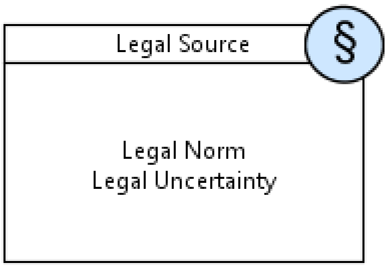
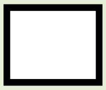
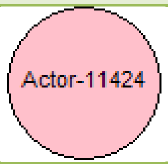
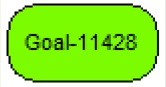
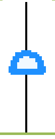
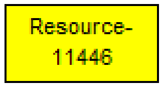
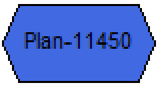
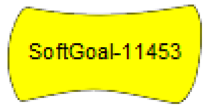
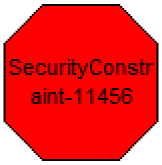
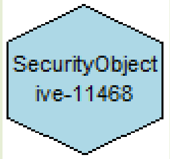
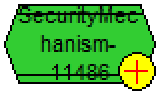
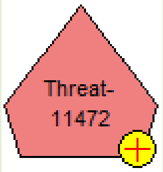
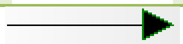
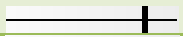
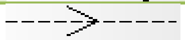
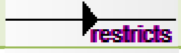
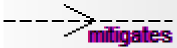
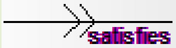
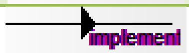
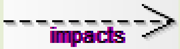
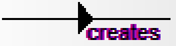
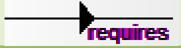
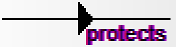
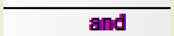
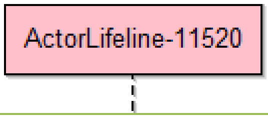
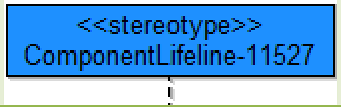
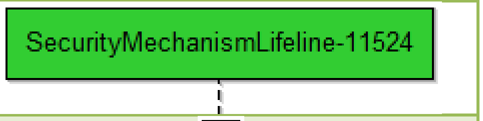
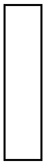
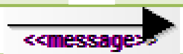
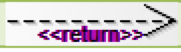
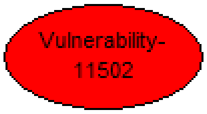
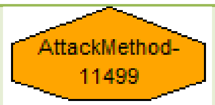
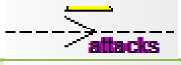
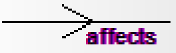
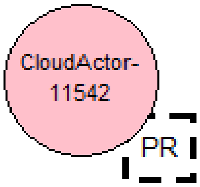
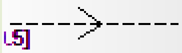
		Unwanted Incident	
		Vulnerability	
	High Level CORAS	Referring Risk	
		Referring Threat Scenario	
		Referring Treatment Scenario	
		Referring Unwanted Incident	
	Dependent CORAS	Border	
	Legal CORAS	Legal Norm	

Table 9-2 SecTro2 Tool Artifacts

	Category	Name	Image
SECTRO2 TOOL ARTIFACTS	Organizational view	Organization	
		Actor	
		Goal	
		Dependency link	
	Security Requirements view ad	Resource	
		Plan	
		Soft Goal	
		Security Constraint	
		Security Objective	
		Security Mechanism	
		Threat	
		Means End	
		Decomposition	
		Contribution	

		Restricts	
		Mitigates	
		Satisfies	
		Implement	
		Impacts	
		Creates	
		Requires	
		Protects	
		And	
	Security Components view	Actor Lifeline	
		Security Mechanism Lifeline	
		Component Lifeline	
		Activation Bar	
		Message Link	
		Return Message Link	
	Security Attacks view	Vulnerability	
		Attack Method	
		Attacks Link	
		Affects Link	
	Cloud Analysis view	Cloud Actor	
		Satisfiability Link	

APPENDIX-B

Results using R-TEA method

Table 10-1 Calculated Results from STS-TOOL

Activities of the Representation Dimension

FEF1.1 Specify uniquely identifiable description using informal language	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	Score	Total Score
							113.00
FEF1.1.1 To what extent this tool provides natural language description for security?	II/2	☐	☒	☐	☐	4	
FEF 1.1.2 To what extent this tool allows to specify unique identification (ID) for each separate requirement?	I/3	☒	☐	☐	☐	9	
FEF 1.1.3 To what extent this tool allows importing of requirements and their description from text document?	III/1	☐	☐	☐	☒	0	
						13	

FEF1.2 Specify requirements using semi-formal language	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
FEF1.2.1 To what extent this tool provides support for semi-formal language description?	I/2	☒	☐	☐	☐	6
FEF 1.2.2 To what extent this tool provides forward/backward traceability between semi-formal informal and formal description?	II/1	☐	☒	☐	☐	2
						8

FEF1.3 Specify requirements using formal languages	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
FEF1.3.1 To what extent this tool provides support for formal language description?	I/2	☐	☐	☒	☐	2
FEF1.3.2 To what extent this tool provides forward/backward traceability between formal and informal, semiformal description?	II/1	☐	☐	☐	☒	0
						2

FEF1.4 Define traceable associations between requirements and the different elements of requirements specification	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
FEF1.4.1 To what extent this tool provide functions for testing traceability between informal, semiformal, and formal requirement description?	V/1	☐	☐	☐	☒	0
FEF1.4.2 To what extent this tool allows to create parent Child traceable relations between requirements?	I/5	☐	☒	☐	☐	10
FEF1.4.3 To what extent this tool allows to maintain peer-to-peer traceable relations between requirements?	II/4	☐	☒	☐	☐	8
FEF1.4.4 To what extent this tool allows to maintain traceable relations between various related information?	III/3	☐	☒	☐	☐	6

FEF1.4.5 To what extent this tool allows to maintain forward/backward traceability between a source of requirements, the requirements and design?	IV/2	☐	☐	☐	☒	0
						24

FEF1.5 Connect seamlessly with other tools and systems, by supporting interoperable protocols and standards	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
FEF1.5.1 To what extent this tool allows importing/exporting requirements description from text document?	I/2	☐	☒	☐	☐	4
FEF1.5.2 To what extent this tool allows importing/exporting of requirements from graphical documents?	II/1	☐	☐	☐	☒	0
						4

Activities of Agreement Dimension

FEF2.1 Maintain an audit trail of changes, archive baseline versions; and engage a mechanism to authenticate and approve change requests.	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
FEF2.1.1 To what extent this tool provides maintainability of user authentication for the system?	I/5	☐	☒	☐	☐	10
FEF2.1.2 To what extent this tool allows grouping of different users?	II/4	☐	☐	☐	☒	0
FEF2.1.3 To what extent this tool provides different views i.e. (documents, requirements, attributes) for different stakeholders?	III/3	☒	☐	☐	☐	9
FEF2.1.4 To what extent this tool allows changes/ history of requirements/ negotiation?	IV/2	☐	☐	☐	☒	0

FEF2.1.5 To what extent this tool allows to call earlier requirement description/versions and register them into history context?	V/1	☐	☐	☐	☒	0
						19.00

FEF2.2 Classify requirements into logical user defined groupings.	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
FEF2.2.1 To what extent this tool allows specifying attributes/properties of the requirement?	I/3	☒	☐	☐	☐	9
FEF2.2.2 To what extent this tool provides sorting according to different attributes/properties?	II/2	☐	☐	☐	☒	0
FEF2.2.3 To what extent this tool provides filtering according to different attributes/properties?	III/1	☐	☐	☐	☒	0
						9

FEF2.3 Support secure, concurrent cooperative work between members of a multidisciplinary team, which may be geographically distributed.	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
FEF2.3.1 To what extent this tool provides independent interface for geographically distributed users?	I/3	☐	☐	☒	☐	3
FEF2.3.2 To what extent this tool allow making a copy for modification of an already approved version of requirements description in different abstract levels (document, requirement)?	II/2	☐	☐	☐	☒	0
FEF2.3.3 To what extent this tool provides change approval cycle for multiple change negotiation and approval before posting into common repository?	III/1	☐	☐	☐	☒	0
						3.00

FEF2.4 Maintain a comprehensive data dictionary of all project components and requirements in a shared repository	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
FEF2.4.1 To what extent this tool provides the single repository or data and concept dictionary?	II/2	☐	☐	☒	☐	2
FEF2.4.2 To what extent this tool provide separate data dictionaries for non-technical users and technical users?	III/1	☐	☐	☐	☒	0
FEF2.4.3 To what extent this tool provides the help system to the user?	I/3	☒	☐	☐	☐	9
						11.00

Activities of the specification Dimension

FEF3.1 Collect and store a common system's and a product family's domain requirements.	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
FEF3.1.1 To what extent this tool enable selection and extraction of common domain requirements and requirements which differentiate systems in product line?	I/4	☐	☐	☐	☒	0
FEF3.1.2 To what extent this tool incorporate requirements to a concrete project?	II/3	☐	☐	☐	☒	0
FEF3.1.3 To what extent this tool adapt/spread changes in domain requirements to concrete projects within domain?	III/2	☐	☐	☐	☒	0
FEF3.1.4 To what extent this tool provides comparison of domain requirements feasibility?	V/1	☐	☐	☐	☒	0
						0.00

FEF3.2 Generate predefined and ad hoc reports, documents that comply with standard industrial templates, with support for presentation-quality output and in-built document quality controls.	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
FEF3.2.1 To what extent this tool provides wizards for report generation?	I/4	☒	☐	☐	☐	12
FEF3.2.2 To what extent this tool provides possibility to print report according to views and sorting?	II/3	☐	☐	☐	☒	0
FEF3.2.3 To what extent this tool provide possibility to print results of rationale, brainstorm and etc.?	III/2	☐	☐	☐	☒	0
FEF3.2.4 To what extent this tool provides techniques for error checking?	IV/1	☐	☐	☐	☒	0
						12.00

FEF3.3 Generate the complete specification, expressed using formal language (informal and semiformal languages might also be included), commonly agreed by all stakeholders.	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
FEF3.3.1 To what extent this tool correspond to standards of software documentation?	I/3	☐	☒	☐	☐	6
FEF3.3.2 To what extent this tool correspond to standards defined by an organization?	II/2	☐	☐	☐	☒	0
FEF3.3.3 To what extent this tool support formal languages for complete, commonly agreed requirements specification?	III/1	☐	☒	☐	☐	2
						8.00

Table 10-2 Calculated Results from CORAS Tool

Activities of the Representation Dimension

FEF1.1 Specify uniquely identifiable description using informal language	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	Score	**Total Score**
							99.00
FEF1.1.1 To what extent this tool provides natural language description for security?	II/2	☒	☐	☐	☐	6	
FEF 1.1.2 To what extent this tool allows to specify unique identification (ID) for each separate requirement?	I/3	☒	☐	☐	☐	9	
FEF 1.1.3 To what extent this tool allows importing of requirements and their description from text document?	III/1	☐	☐	☐	☒	0	
						15	

FEF1.2 Specify requirements using semi-formal language	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
FEF1.2.1 To what extent this tool provides support for semi-formal language description?	I/2	☒	☐	☐	☐	6
FEF 1.2.2 To what extent this tool provides forward/backward traceability between semi-formal informal and formal description?	II/1	☐	☒	☐	☐	2
						8

FEF1.3 Specify requirements using formal languages	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
FEF1.3.1 To what extent this tool provides support for formal language description?	I/2	☐	☐	☒	☐	2
FEF1.3.2 To what extent this tool provides forward/backward traceability between formal and informal, semiformal description?	II/1	☐	☐	☐	☒	0
						2

FEF1.4 Define traceable associations between requirements and the different elements of requirements specification	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
FEF1.4.1 To what extent this tool provide functions for testing traceability between informal, semiformal, and formal requirement description?	V/1	☐	☐	☐	☒	0
FEF1.4.2 To what extent this tool allows to create parent Child traceable relations between requirements?	I/5	☐	☒	☐	☐	10
FEF1.4.3 To what extent this tool allows to maintain peer-to-peer traceable relations between requirements?	II/4	☐	☒	☐	☐	8
FEF1.4.4 To what extent this tool allows to maintain traceable relations between various related information?	III/3	☐	☒	☐	☐	6
FEF1.4.5 To what extent this tool allows to maintain forward/backward traceability between a source of requirements, the requirements and design?	IV/2	☐	☒	☐	☒	4
						28

FEF1.5 Connect seamlessly with other tools and systems, by supporting interoperable protocols and standards	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
FEF1.5.1 To what extent this tool allows importing/exporting requirements description from text document?	I/2	☐	☐	☐	☒	0
FEF1.5.2 To what extent this tool allows importing/exporting of requirements from graphical documents?	II/1	☐	☐	☐	☒	0
						0

Activities of Agreement Dimension

FEF2.1 Maintain an audit trail of changes, archive	Priority	Full	Above	Minimal	No

baseline versions; and engage a mechanism to authenticate and approve change requests.	/Score	Support	Average Support	Support	Support	
FEF2.1.1 To what extent this tool provides maintainability of user authentication for the system?	I/5	☒	☐	☐	☐	15
FEF2.1.2 To what extent this tool allows grouping of different users?	II/4	☐	☐	☐	☒	0
FEF2.1.3 To what extent this tool provides different views i.e. (documents, requirements, attributes) for different stakeholders?	III/3	☐	☒	☐	☐	6
FEF2.1.4 To what extent this tool allows changes/ history of requirements/ negotiation?	IV/2	☐	☐	☐	☒	0
FEF2.1.5 To what extent this tool allows to call earlier requirement description/versions and register them into history context?	V/1	☐	☐	☐	☒	0
						21.00

FEF2.2 Classify requirements into logical user defined groupings.	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
FEF2.2.1 To what extent this tool allows specifying attributes/properties of the requirement?	I/3	☒	☐	☐	☐	9
FEF2.2.2 To what extent this tool provides sorting according to different attributes/properties?	II/2	☐	☐	☐	☒	0
FEF2.2.3 To what extent this tool provides filtering according to different attributes/properties?	III/1	☐	☐	☐	☒	0
						9

FEF2.3 Support secure, concurrent cooperative work between members of a multidisciplinary team, which may be geographically distributed.	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
FEF2.3.1 To what extent this tool provides independent interface for geographically distributed users?	I/3	☐	☐	☒	☐	3
FEF2.3.2 To what extent this tool allow	II/2					0

making a copy for modification of an already approved version of requirements description in different abstract levels (document, requirement)?		☐	☐	☐	☒	
FEF2.3.3 To what extent this tool provides change approval cycle for multiple change negotiation and approval before posting into common repository?	III/1	☐	☐	☐	☒	0
						3.00

FEF2.4 Maintain a comprehensive data dictionary of all project components and requirements in a shared repository	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
FEF2.4.1 To what extent this tool provides the single repository or data and concept dictionary?	II/2	☐	☐	☒	☐	2
FEF2.4.2 To what extent this tool provide separate data dictionaries for non-technical users and technical users?	III/1	☐	☐	☐	☒	0
FEF2.4.3 To what extent this tool provides the help system to the user?	I/3	☐	☐	☐	☒	0
						2.00

Activities of the specification Dimension

FEF3.1 Collect and store a common system's and a product family's domain requirements.	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
FEF3.1.1 To what extent this tool enable selection and extraction of common domain requirements and requirements which differentiate systems in product line?	I/4	☐	☐	☐	☒	0
FEF3.1.2 To what extent this tool incorporate requirements to a concrete project?	II/3	☐	☐	☐	☒	0
FEF3.1.3 To what extent this tool	III/2					0

adapt/spread changes in domain requirements to concrete projects within domain?		☐	☐	☐	☒	
FEF3.1.4 To what extent this tool provides comparison of domain requirements feasibility?	V/1	☐	☐	☐	☒	0
						0.00

FEF3.2 Generate predefined and ad hoc reports, documents that comply with standard industrial templates, with support for presentation-quality output and in-built document quality controls.	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
FEF3.2.1 To what extent this tool provides wizards for report generation?	I/4	☐	☐	☐	☒	0
FEF3.2.2 To what extent this tool provides possibility to print report according to views and sorting?	II/3	☐	☐	☐	☒	0
FEF3.2.3 To what extent this tool provide possibility to print results of rationale, brainstorm and etc.?	III/2	☐	☐	☐	☒	0
FEF3.2.4 To what extent this tool provides techniques for error checking?	IV/1	☐	☐	☐	☒	0
						0.00

FEF3.3 Generate the complete specification, expressed using formal language (informal and semiformal languages might also be included), commonly agreed by all stakeholders.	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
FEF3.3.1 To what extent this tool correspond to standards of software documentation?	I/3	☒	☐	☐	☐	9
FEF3.3.2 To what extent this tool correspond to standards defined by an organization?	II/2	☐	☐	☐	☒	0
FEF3.3.3 To what extent this tool support formal languages for complete, commonly agreed requirements specification?	III/1	☐	☒	☐	☐	2
						11.00

Table 10-3 Calculated Results from SQUARE Tool

Activities of the Representation Dimension

FEF1.1 Specify uniquely identifiable description using informal language	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	Score	Total Score 167.00
FEF1.1.1 To what extent this tool provides natural language description for security?	II/2	☒	☐	☐	☐	6	
FEF 1.1.2 To what extent this tool allows to specify unique identification (ID) for each separate requirement?	I/3	☒	☐	☐	☐	9	
FEF 1.1.3 To what extent this tool allows importing of requirements and their description from text document?	III/1	☒	☐	☐	☐	3	
						18	

FEF1.2 Specify requirements using semi-formal language	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
FEF1.2.1 To what extent this tool provides support for semi-formal language description?	I/2	☐	☐	☐	☒	0
FEF 1.2.2 To what extent this tool provides forward/backward traceability between semi-formal informal and formal description?	II/1	☐	☐	☐	☒	0
						0

FEF1.3 Specify requirements using formal languages	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
FEF1.3.1 To what extent this tool provides support for formal language description?	I/2	☒	☐	☐	☐	6
FEF1.3.2 To what extent this tool provides	II/1	☐	☒	☐	☐	2

forward/backward traceability between formal and informal, semiformal description?						
						8

FEF1.4 Define traceable associations between requirements and the different elements of requirements specification	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
FEF1.4.1 To what extent this tool provide functions for testing traceability between informal, semiformal, and formal requirement description?	V/1	☐	☐	☐	☒	0
FEF1.4.2 To what extent this tool allows to create parent Child traceable relations between requirements?	I/5	☐	☒	☐	☐	10
FEF1.4.3 To what extent this tool allows to maintain peer-to-peer traceable relations between requirements?	II/4	☐	☒	☐	☐	8
FEF1.4.4 To what extent this tool allows to maintain traceable relations between various related information?	III/3	☐	☒	☐	☐	6
FEF1.4.5 To what extent this tool allows to maintain forward/backward traceability between a source of requirements, the requirements and design?	IV/2	☐	☒	☐	☐	4
						28

FEF1.5 Connect seamlessly with other tools and systems, by supporting interoperable protocols and standards	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
FEF1.5.1 To what extent this tool allows importing/exporting requirements description from text document?	I/2	☒	☐	☐	☐	6
FEF1.5.2 To what extent this tool allows	II/1					0

importing/exporting of requirements from graphical documents?		☐	☐	☐	☒	
						6

Activities of Agreement Dimension

FEF2.1 Maintain an audit trail of changes, archive baseline versions; and engage a mechanism to authenticate and approve change requests.	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
FEF2.1.1 To what extent this tool provides maintainability of user authentication for the system?	I/5	☐	☒	☐	☐	10
FEF2.1.2 To what extent this tool allows grouping of different users?	II/4	☐	☐	☐	☒	0
FEF2.1.3 To what extent this tool provides different views i.e. (documents, requirements, attributes) for different stakeholders?	III/3	☐	☐	☐	☒	0
FEF2.1.4 To what extent this tool allows changes/ history of requirements/ negotiation?	IV/2	☐	☒	☐	☐	4
FEF2.1.5 To what extent this tool allows to call earlier requirement description/versions and register them into history context?	V/1	☐	☒	☐	☐	2
						16.00

FEF2.2 Classify requirements into logical user defined groupings.	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
FEF2.2.1 To what extent this tool allows specifying attributes/properties of the requirement?	I/3	☐	☒	☐	☐	6
FEF2.2.2 To what extent this tool provides sorting according to different attributes/properties?	II/2	☒	☐	☐	☐	6
FEF2.2.3 To what extent this tool provides filtering according to different attributes/properties?	III/1	☒	☐	☐	☐	3
						15

FEF2.3 Support secure, concurrent cooperative work between members of a multidisciplinary team, which may be geographically distributed.	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
FEF2.3.1 To what extent this tool provides independent interface for geographically distributed users?	I/3	☐	☐	☒	☐	3
FEF2.3.2 To what extent this tool allow making a copy for modification of an already approved version of requirements description in different abstract levels (document, requirement)?	II/2	☐	☐	☐	☒	0
FEF2.3.3 To what extent this tool provides change approval cycle for multiple change negotiation and approval before posting into common repository?	III/1	☐	☐	☐	☒	0
						3.00

FEF2.4 Maintain a comprehensive data dictionary of all project components and requirements in a shared repository	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
FEF2.4.1 To what extent this tool provides the single repository or data and concept dictionary?	II/2	☐	☐	☒	☐	2
FEF2.4.2 To what extent this tool provide separate data dictionaries for non-technical users and technical users?	III/1	☐	☐	☐	☒	0
FEF2.4.3 To what extent this tool provides the help system to the user?	I/3	☐	☐	☒	☐	3
						5.00

Activities of the specification Dimension

FEF3.1 Collect and store a common system's and a product family's domain requirements.	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
FEF3.1.1 To what extent this tool enable	I/4					8

selection and extraction of common domain requirements and requirements which differentiate systems in product line?		☐	☒	☐	☐	
FEF3.1.2 To what extent this tool incorporate requirements to a concrete project?	II/3	☐	☐	☒	☐	3
FEF3.1.3 To what extent this tool adapt/spread changes in domain requirements to concrete projects within domain?	III/2	☐	☐	☒	☐	2
FEF3.1.4 To what extent this tool provides comparison of domain requirements feasibility?	V/1	☐	☐	☐	☒	0
						13.00
FEF3.2 Generate predefined and ad hoc reports, documents that comply with standard industrial templates, with support for presentation-quality output and in-built document quality controls.	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
FEF3.2.1 To what extent this tool provides wizards for report generation?	I/4	☒	☐	☐	☐	12
FEF3.2.2 To what extent this tool provides possibility to print report according to views and sorting?	II/3	☒	☐	☐	☒	9
FEF3.2.3 To what extent this tool provide possibility to print results of rationale, brainstorm and etc.?	III/2	☐	☐	☒	☐	26
FEF3.2.4 To what extent this tool provides techniques for error checking?	IV/1	☐	☐	☐	☒	0
						47.00
FEF3.3 Generate the complete specification, expressed using formal language (informal and semiformal languages might also be included), commonly agreed by all stakeholders.	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
FEF3.3.1 To what extent this tool correspond to standards of software documentation?	I/3	☐	☒	☐	☐	6
FEF3.3.2 To what extent this tool correspond	II/2					0

to standards defined by an organization?		☐	☐	☐	☒	
FEF3.3.3 To what extent this tool support formal languages for complete, commonly agreed requirements specification?	III/1	☐	☒	☐	☐	2
						8.00

Table 10-4 Calculated Results from SecTro2 Tool

Activities of the Representation Dimension

FEF1.1 Specify uniquely identifiable description using informal language	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	Score	Total Score 102.00
FEF1.1.1 To what extent this tool provides natural language description for security?	II/2	☒	☐	☐	☐	6	
FEF 1.1.2 To what extent this tool allows to specify unique identification (ID) for each separate requirement?	I/3	☒	☐	☐	☐	9	
FEF 1.1.3 To what extent this tool allows importing of requirements and their description from text document?	III/1	☐	☐	☐	☒	0	
						15	

FEF1.2 Specify requirements using semi-formal language	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
FEF1.2.1 To what extent this tool provides support for semi-formal language description?	I/2	☒	☐	☐	☐	6
FEF 1.2.2 To what extent this tool provides forward/backward traceability between semi-formal informal and formal description?	II/1	☐	☒	☐	☐	2
						8

FEF1.3 Specify requirements using formal languages	Priority /Score	Full Support	Above Average	Minimal Support	No Support

			Support			
FEF1.3.1 To what extent this tool provides support for formal language description?	I/2	☐	☐	☒	☐	2
FEF1.3.2 To what extent this tool provides forward/backward traceability between formal and informal, semiformal description?	II/1	☐	☐	☐	☒	0
						2

FEF1.4 Define traceable associations between requirements and the different elements of requirements specification	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
FEF1.4.1 To what extent this tool provide functions for testing traceability between informal, semiformal, and formal requirement description?	V/1	☐	☐	☐	☒	0
FEF1.4.2 To what extent this tool allows to create parent Child traceable relations between requirements?	I/5	☐	☒	☐	☐	10
FEF1.4.3 To what extent this tool allows to maintain peer-to-peer traceable relations between requirements?	II/4	☐	☒	☐	☐	8
FEF1.4.4 To what extent this tool allows to maintain traceable relations between various related information?	III/3	☐	☒	☐	☐	6
FEF1.4.5 To what extent this tool allows to maintain forward/backward traceability between a source of requirements, the requirements and design?	IV/2	☐	☒	☐	☐	4
						28

FEF1.5 Connect seamlessly with other tools and systems, by supporting interoperable protocols and standards	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
FEF1.5.1 To what extent this tool allows	I/2					0

importing/exporting requirements description from text document?		☐	☐	☐	☒	
FEF1.5.2 To what extent this tool allows importing/exporting of requirements from graphical documents?	II/1	☐	☐	☐	☒	0
						0

Activities of Agreement Dimension

FEF2.1 Maintain an audit trail of changes, archive baseline versions; and engage a mechanism to authenticate and approve change requests.	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
FEF2.1.1 To what extent this tool provides maintainability of user authentication for the system?	I/5	☒	☐	☐	☐	15
FEF2.1.2 To what extent this tool allows grouping of different users?	II/4	☐	☐	☐	☒	0
FEF2.1.3 To what extent this tool provides different views i.e. (documents, requirements, attributes) for different stakeholders?	III/3	☐	☒	☐	☐	6
FEF2.1.4 To what extent this tool allows changes/ history of requirements/ negotiation?	IV/2	☐	☐	☐	☒	0
FEF2.1.5 To what extent this tool allows to call earlier requirement description/versions and register them into history context?	V/1	☐	☐	☐	☒	0
						21.00

FEF2.2 Classify requirements into logical user defined groupings.	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
FEF2.2.1 To what extent this tool allows specifying attributes/properties of the requirement?	I/3	☒	☐	☐	☐	9
FEF2.2.2 To what extent this tool provides	II/2	☐	☐	☐	☒	0

sorting according to different attributes/properties?						
FEF2.2.3 To what extent this tool provides filtering according to different attributes/properties?	III/1	☐	☐	☐	☒	0
						9

FEF2.3 Support secure, concurrent cooperative work between members of a multidisciplinary team, which may be geographically distributed.	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
FEF2.3.1 To what extent this tool provides independent interface for geographically distributed users?	I/3	☐	☐	☒	☐	3
FEF2.3.2 To what extent this tool allow making a copy for modification of an already approved version of requirements description in different abstract levels (document, requirement)?	II/2	☐	☐	☐	☒	0
FEF2.3.3 To what extent this tool provides change approval cycle for multiple change negotiation and approval before posting into common repository?	III/1	☐	☐	☐	☒	0
						3.00

FEF2.4 Maintain a comprehensive data dictionary of all project components and requirements in a shared repository	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
FEF2.4.1 To what extent this tool provides the single repository or data and concept dictionary?	II/2	☐	☐	☒	☐	2
FEF2.4.2 To what extent this tool provide separate data dictionaries for non-technical users and technical users?	III/1	☐	☐	☐	☒	0
FEF2.4.3 To what extent this tool provides the help system to the user?	I/3	☐	☒	☐	☐	6
						8.00

Activities of the specification Dimension

FEF3.1 Collect and store a common system's and a product family's domain requirements.	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
FEF3.1.1 To what extent this tool enable selection and extraction of common domain requirements and requirements which differentiate systems in product line?	I/4	☐	☐	☐	☒	0
FEF3.1.2 To what extent this tool incorporate requirements to a concrete project?	II/3	☐	☐	☐	☒	0
FEF3.1.3 To what extent this tool adapt/spread changes in domain requirements to concrete projects within domain?	III/2	☐	☐	☐	☒	0
FEF3.1.4 To what extent this tool provides comparison of domain requirements feasibility?	V/1	☐	☐	☐	☒	0
						0.00

FEF3.2 Generate predefined and ad hoc reports, documents that comply with standard industrial templates, with support for presentation-quality output and in-built document quality controls.	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
FEF3.2.1 To what extent this tool provides wizards for report generation?	I/4	☐	☐	☐	☒	0
FEF3.2.2 To what extent this tool provides possibility to print report according to views and sorting?	II/3	☐	☐	☐	☒	0
FEF3.2.3 To what extent this tool provide possibility to print results of rationale, brainstorm and etc.?	III/2	☐	☐	☐	☒	0
FEF3.2.4 To what extent this tool provides techniques for error checking?	IV/1	☐	☐	☐	☒	0
						0.00

FEF3.3 Generate the complete specification, expressed using formal language (informal and semiformal languages might also be included), commonly agreed by all stakeholders.	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support

FEF3.3.1 To what extent this tool correspond to standards of software documentation?	I/3	☐	☒	☐	☐	6
FEF3.3.2 To what extent this tool correspond to standards defined by an organization?	II/2	☐	☐	☐	☒	0
FEF3.3.3 To what extent this tool support formal languages for complete, commonly agreed requirements specification?	III/1	☐	☒	☐	☐	2
						8.00

Table 10-5 Calculated Results from Magic Draw Tool

Activities of the Representation Dimension

FEF1.1 Specify uniquely identifiable description using informal language	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	Score	**Total Score** 110.00
FEF1.1.1 To what extent this tool provides natural language description for security?	II/2	☒	☐	☐	☐	6	
FEF 1.1.2 To what extent this tool allows to specify unique identification (ID) for each separate requirement?	I/3	☒	☐	☐	☐	9	
FEF 1.1.3 To what extent this tool allows importing of requirements and their description from text document?	III/1	☐	☐	☐	☒	0	
						15	

FEF1.2 Specify requirements using semi-formal language	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
FEF1.2.1 To what extent this tool provides support for semi-formal language description?	I/2	☒	☐	☐	☐	6
FEF 1.2.2 To what extent this tool provides forward/backward traceability between semi-formal informal and formal description?	II/1	☒	☒	☐	☐	3

9

FEF1.3 Specify requirements using formal languages	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
FEF1.3.1 To what extent this tool provides support for formal language description?	I/2	☐	☒	☒	☐	4
FEF1.3.2 To what extent this tool provides forward/backward traceability between formal and informal, semiformal description?	II/1	☐	☒	☐	☐	2

6

FEF1.4 Define traceable associations between requirements and the different elements of requirements specification	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
FEF1.4.1 To what extent this tool provide functions for testing traceability between informal, semiformal, and formal requirement description?	V/1	☐	☐	☐	☒	0
FEF1.4.2 To what extent this tool allows to create parent Child traceable relations between requirements?	I/5	☐	☒	☐	☐	10
FEF1.4.3 To what extent this tool allows to maintain peer-to-peer traceable relations between requirements?	II/4	☐	☒	☐	☐	8
FEF1.4.4 To what extent this tool allows to maintain traceable relations between various related information?	III/3	☒	☐	☐	☐	9
FEF1.4.5 To what extent this tool allows to maintain forward/backward traceability between a source of requirements, the requirements and design?	IV/2	☐	☒	☐	☐	4

31

FEF1.5 Connect seamlessly with other tools and systems, by supporting interoperable protocols and standards	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
FEF1.5.1 To what extent this tool allows importing/exporting requirements description from text document?	I/2	☐	☐	☐	☒	0
FEF1.5.2 To what extent this tool allows importing/exporting of requirements from graphical documents?	II/1	☐	☐	☐	☒	0
						0

Activities of Agreement Dimension

FEF2.1 Maintain an audit trail of changes, archive baseline versions; and engage a mechanism to authenticate and approve change requests.	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
FEF2.1.1 To what extent this tool provides maintainability of user authentication for the system?	I/5	☐	☒	☐	☐	10
FEF2.1.2 To what extent this tool allows grouping of different users?	II/4	☐	☐	☐	☒	0
FEF2.1.3 To what extent this tool provides different views i.e. (documents, requirements, attributes) for different stakeholders?	III/3	☐	☒	☐	☐	6
FEF2.1.4 To what extent this tool allows changes/ history of requirements/ negotiation?	IV/2	☐	☐	☐	☒	0
FEF2.1.5 To what extent this tool allows to call earlier requirement description/versions and register them into history context?	V/1	☐	☐	☐	☒	0
						16.00

FEF2.2 Classify requirements into logical user defined groupings.	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
FEF2.2.1 To what extent this tool allows specifying attributes/properties of the requirement?	I/3	☐	☒	☐	☐	6

FEF2.2.2 To what extent this tool provides sorting according to different attributes/properties?	II/2	☐	☐	☐	☒	0
FEF2.2.3 To what extent this tool provides filtering according to different attributes/properties?	III/1	☐	☐	☐	☒	0
						6

FEF2.3 Support secure, concurrent cooperative work between members of a multidisciplinary team, which may be geographically distributed.	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
FEF2.3.1 To what extent this tool provides independent interface for geographically distributed users?	I/3	☐	☒	☐	☐	6
FEF2.3.2 To what extent this tool allow making a copy for modification of an already approved version of requirements description in different abstract levels (document, requirement)?	II/2	☐	☐	☐	☒	0
FEF2.3.3 To what extent this tool provides change approval cycle for multiple change negotiation and approval before posting into common repository?	III/1	☐	☐	☐	☒	0
						6.00

FEF2.4 Maintain a comprehensive data dictionary of all project components and requirements in a shared repository	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
FEF2.4.1 To what extent this tool provides the single repository or data and concept dictionary?	II/2	☐	☐	☒	☐	2
FEF2.4.2 To what extent this tool provide separate data dictionaries for non-technical users and technical users?	III/1	☐	☒	☐	☐	2
FEF2.4.3 To what extent this tool provides the help system to the user?	I/3	☒	☐	☐	☐	9
						13.00

Activities of the specification Dimension

FEF3.1 Collect and store a common system's and a product family's domain requirements.	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
FEF3.1.1 To what extent this tool enable selection and extraction of common domain requirements and requirements which differentiate systems in product line?	I/4	☐	☐	☐	☒	0
FEF3.1.2 To what extent this tool incorporate requirements to a concrete project?	II/3	☐	☐	☐	☒	0
FEF3.1.3 To what extent this tool adapt/spread changes in domain requirements to concrete projects within domain?	III/2	☐	☐	☐	☒	0
FEF3.1.4 To what extent this tool provides comparison of domain requirements feasibility?	V/1	☐	☐	☐	☒	0
						0.00

FEF3.2 Generate predefined and ad hoc reports, documents that comply with standard industrial templates, with support for presentation-quality output and in-built document quality controls.	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
FEF3.2.1 To what extent this tool provides wizards for report generation?	I/4	☐	☐	☐	☒	0
FEF3.2.2 To what extent this tool provides possibility to print report according to views and sorting?	II/3	☐	☐	☐	☒	0
FEF3.2.3 To what extent this tool provide possibility to print results of rationale, brainstorm and etc.?	III/2	☐	☐	☐	☒	0
FEF3.2.4 To what extent this tool provides techniques for error checking?	IV/1	☐	☐	☐	☒	0
						0.00

FEF3.3 Generate the complete specification, expressed using formal language (informal and semiformal languages might also be included), commonly agreed by all stakeholders.	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support

FEF3.3.1 To what extent this tool correspond to standards of software documentation?	I/3	☐	☒	☐	☐	6
FEF3.3.2 To what extent this tool correspond to standards defined by an organization?	II/2	☐	☐	☐	☒	0
FEF3.3.3 To what extent this tool support formal languages for complete, commonly agreed requirements specification?	III/1	☐	☒	☐	☐	2
						8.00

APPENDIX-C

Below are the results and the score of security engineering tools evaluation using method constructed in this research.

Table 11-1 Calculated Results for STS-TOOL

Table 0-1 Likert Scale for SR001-Making Awareness

Sample questions	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	Score	Total Score 80
To which extent this tool provides online support, e.g. downloads and documentation?	I/6	☒	☐	☐	☐	18	
To what extent this tools' book on methodology is explained and/or to what extent this tool adapts the methodology?	II/5	☐	☐	☒	☐	5	
To what extent this tool provides support for the user manual?	III/4	☒	☐	☐	☐	12	
To what extent this tool provides tutorials? That will support and provide learning opportunity.	IV/3	☒	☐	☐	☐	9	
To what extent this tool has gained popularity to support and provide several publications?	V/2	☐	☒	☐	☐	4	
To what extent this tool provides support for hands on wizards? These will guide engineers through several aspects of tool?	VI/1	☐	☐	☐	☒	0	
						48	

Table 0-2 Likert Scale for SR002-Understand Context and Assets

Sample questions	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
To which extent this tool supports the feature to create asset diagram, or at-least include a feature similar to declare assets in efficient way?	I/3	☒	☐	☐	☐	9

To what extent this tool supports the feature that includes a template to document assets and goals?	II/2	☐	☐	☐	☒	0
To which extent this tool supports the feature to create checklist of assets and goals?	III/1	☐	☐	☐	☒	0
						9

Table 0-3 Likert Scale for SR003-Security Requirements

Sample questions	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
To which extent this tool supports the feature to create treatment diagram, or at-least include a feature similar to declare treatments in efficient way?	I/3	☐	☐	☐	☒	0
To what extent this tool supports the feature that can help selecting security requirements elicitation techniques?	II/2	☐	☐	☐	☒	0
To what extent this tool supports template to document security requirements?	III/1	☐	☐	☐	☒	0
						0

Table 0-4 Likert Scale for SR004-Risk Analysis

Sample questions	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
To which extent this tool supports feature to create risk diagram, or at-least include a feature similar to declare risks in efficient way?	I/4	☒	☐	☐	☐	12
To which extent this tool supports feature to create threat diagram, or at-least include a feature similar to declare threats in efficient way?	II/3	☒	☐	☐	☐	9
To what extent this tool supports template to document risks?	III/2	☐	☐	☐	☒	0
To what extent this tool supports feature to create checklist of vulnerabilities, security threats and risks?	IV/1					0

		☐	☐	☐	☒	

21

Table 0-5 Likert Scale for SR005-Secure Design Practices

Sample questions	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
To which extent this tool supports the feature to create security mechanism, or at-least include a feature similar to declare security mechanism in efficient way?	I/2	☐	☐	☐	☒	0
To what extent this tool supports the feature that can help selecting secure software architecture and secure software design?	II/1	☐	☐	☐	☒	0

0

Table 0-6 Likert Scale for SR006-Justify Design Solution

Sample questions	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
To which extent this tool supports feature that can help calculating risk estimations?	I/3	☐	☐	☐	☒	0
To what extent this tool supports feature that can help categorizing and prioritizing security requirements?	II/2	☐	☐	☐	☒	0
To what extent this tool supports mechanism to estimate trade-off analysis?	III/1	☐	☐	☐	☒	0

0

Table 0-7 Likert Scale for SR007-Response

Sample questions	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
To what extent this tool supports feature that can help creating response plan?	I/2					2

		□	□	☒	□	
To what extent this tool supports feature to create templates for security requirements inspection?	II/1	□	□	□	☒	0
						2

Table 11-2 Calculated Results for CORAS TOOL

Table 0-1 Likert Scale for SR001-Making Awareness

Total Score

109.00

Sample questions	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	Score
To which extent this tool provides online support, e.g. downloads and documentation?	I/6	☒	□	□	□	18
To what extent this tools' book on methodology is explained and/or to what extent this tool adapts the methodology?	II/5	☒	□	□	□	15
To what extent this tool provides support for the user manual?	III/4	□	□	☒	□	4
To what extent this tool provides tutorials? That will support and provide learning opportunity.	IV/3	☒	□	□	□	9
To what extent this tool has gained popularity to support and provide several publications?	V/2	□	☒	□	□	4
To what extent this tool provides support for hands on wizards? These will guide engineers through several aspects of tool?	VI/1	□	□	□	☒	0
						50.00

Table 0-2 Likert Scale for SR002-Understand Context and Assets

Sample questions	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
To which extent this tool supports the feature	I/3					9

to create asset diagram, or at-least include a feature similar to declare assets in efficient way?		☒	☐	☐	☐	
To what extent this tool supports the feature that includes a template to document assets and goals?	II/2	☐	☐	☐	☒	0
To which extent this tool supports the feature to create checklist of assets and goals?	III/1	☐	☐	☐	☒	0
						9

Table 0-3 Likert Scale for SR003-Security Requirements

Sample questions	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
To which extent this tool supports the feature to create treatment diagram, or at-least include a feature similar to declare treatments in efficient way?	I/3	☒	☐	☐	☐	9
To what extent this tool supports the feature that can help selecting security requirements elicitation techniques?	II/2	☐	☐	☐	☒	0
To what extent this tool supports template to document security requirements?	III/1	☐	☐	☐	☒	0
						9

Table 0-4 Likert Scale for SR004-Risk Analysis

Sample questions	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
To which extent this tool supports feature to create risk diagram, or at-least include a feature similar to declare risks in efficient way?	I/4	☒	☐	☐	☐	12
To which extent this tool supports feature to create threat diagram, or at-least include a feature similar to declare threats in efficient way?	II/3	☒	☐	☐	☐	9
To what extent this tool supports templates to	III/2					0

document risks?		☐	☐	☐	☒	
To what extent this tool supports feature to create checklist of vulnerabilities, security threats and risks?	IV/1	☐	☐	☐	☒	0
						21

Table 0-5 Likert Scale for SR005-Secure Design Practices

Sample questions	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
To which extent this tool supports the feature to create security mechanism, or at-least include a feature similar to declare security mechanism in efficient way?	I/2	☒	☐	☐	☐	6
To what extent this tool supports the feature that can help selecting secure software architecture and secure software design?	II/1	☐	☐	☐	☒	0
						6

Table 0-6 Likert Scale for SR006-Justify Design Solution

Sample questions	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
To which extent this tool supports feature that can help calculating risk estimations?	I/3	☐	☒	☐	☐	6
To what extent this tool supports feature that can help categorizing and prioritizing security requirements?	II/2	☐	☐	☐	☒	0
To what extent this tool supports mechanism to estimate trade-off analysis?	III/1	☐	☒	☐	☐	2
						8.00

Table 0-7 Likert Scale for SR007-Response

Sample questions	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
To what extent this tool supports feature that	I/2					6

can help creating response plan?		☒	□	□	□	
To what extent this tool supports feature to create templates for security requirements inspection?	II/1	□	□	□	☒	0
						6

Table 11-3 Calculated Results for SQUARE TOOL

Table 0-1 Likert Scale for SR001-Making Awareness

Total Score

75.00

Sample questions	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	Score
To which extent this tool provides online support, e.g. downloads and documentation?	I/6	□	☒	□	□	12
To what extent this tools' book on methodology is explained and/or to what extent this tool adapts the methodology?	II/5	□	□	☒	□	5
To what extent this tool provides support for the user manual?	III/4	□	□	□	☒	0
To what extent this tool provides tutorials? That will support and provide learning opportunity.	IV/3	□	□	□	☒	9
To what extent this tool has gained popularity to support and provide several publications?	V/2	□	☒	□	□	4
To what extent this tool provides support for hands on wizards? These will guide engineers through several aspects of tool?	VI/1	☒	□	□	□	3
						33.00

Table 0-2 Likert Scale for SR002-Understand Context and Assets

Sample questions	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
To which extent this tool supports the feature	I/3					3

to create asset diagram, or at-least include a feature similar to declare assets in efficient way?		☐	☐	☒	☐	
To what extent this tool supports the feature that includes a template to document assets and goals?	II/2	☒	☐	☐	☐	6
To which extent this tool supports the feature to create checklist of assets and goals?	III/1	☒	☐	☐	☐	3
						12

Table 0-3 Likert Scale for SR003-Security Requirements

Sample questions	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
To which extent this tool supports the feature to create treatment diagram, or at-least include a feature similar to declare treatments in efficient way?	I/3	☐	☐	☒	☐	3
To what extent this tool supports the feature that can help selecting security requirements elicitation techniques?	II/2	☒	☐	☐	☐	6
To what extent this tool supports template to document security requirements?	III/1	☒	☐	☐	☐	3
						12

Table 0-4 Likert Scale for SR004-Risk Analysis

Sample questions	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
To which extent this tool supports feature to create risk diagram, or at-least include a feature similar to declare risks in efficient way?	I/4	☐	☐	☐	☒	0
To which extent this tool supports feature to create threat diagram, or at-least include a feature similar to declare threats in efficient way?	II/3	☐	☐	☐	☒	0
To what extent this tool supports template to	III/2					6

document risks?		☒	☐	☐	☐	
To what extent this tool supports feature to create checklist of vulnerabilities, security threats and risks?	IV/1	☒	☐	☐	☐	3
						9

Table 0-5 Likert Scale for SR005-Secure Design Practices

Sample questions	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
To which extent this tool supports the feature to create security mechanism, or at-least include a feature similar to declare security mechanism in efficient way?	I/2	☐	☐	☐	☒	0
To what extent this tool supports the feature that can help selecting secure software architecture and secure software design?	II/1	☒	☐	☐	☐	0
						0

Table 0-6 Likert Scale for SR006-Justify Design Solution

Sample questions	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
To which extent this tool supports feature that can help calculating risk estimations?	I/3	☐	☐	☐	☒	0
To what extent this tool supports feature that can help categorizing and prioritizing security requirements?	II/2	☒	☐	☐	☐	6
To what extent this tool supports mechanism to estimate trade-off analysis?	III/1	☐	☐	☐	☒	0
						6.00

Table 0-7 Likert Scale for SR007-Response

Sample questions	Priority	Full	Above	Minimal	No

	/Score	Support	Average Support	Support	Support	
To what extent this tool supports feature that can help creating response plan?	I/2	☐	☐	☐	☒	0
To what extent this tool supports feature to create templates for security requirements inspection?	II/1	☒	☐	☐	☐	3
						3

Table 11-4 Calculated Results for SecTro2 TOOL

Table 0-1 Likert Scale for SR001-Making Awareness

Sample questions	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	Score	Total Score 80.00
To which extent this tool provides online support, e.g. downloads and documentation?	I/6	☐	☒	☐	☐	12	
To what extent this tools' book on methodology is explained and/or to what extent this tool adapts the methodology?	II/5	☐	☐	☒	☐	5	
To what extent this tool provides support for the user manual?	III/4	☒	☐	☐	☐	12	
To what extent this tool provides tutorials? That will support and provide learning opportunity.	IV/3	☐	☐	☐	☒	0	
To what extent this tool has gained popularity to support and provide several publications?	V/2	☐	☒	☐	☐	4	
To what extent this tool provides support for hands on wizards? These will guide engineers through several aspects of tool?	VI/1	☐	☐	☐	☒	0	
						33.00	

Table 0-2 Likert Scale for SR002-Understand Context and Assets

Sample questions	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
To which extent this tool supports the feature to create asset diagram, or at-least include a feature similar to declare assets in efficient way?	I/3	☒	☐	☐	☐	9
To what extent this tool supports the feature that includes a template to document assets and goals?	II/2	☐	☐	☐	☒	0
To which extent this tool supports the feature to create checklist of assets and goals?	III/1	☐	☐	☐	☒	0
						9

Table 0-3 Likert Scale for SR003-Security Requirements

Sample questions	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
To which extent this tool supports the feature to create treatment diagram, or at-least include a feature similar to declare treatments in efficient way?	I/3	☒	☐	☐	☐	9
To what extent this tool supports the feature that can help selecting security requirements elicitation techniques?	II/2	☐	☐	☐	☒	0
To what extent this tool supports template to document security requirements?	III/1	☐	☐	☐	☒	0
						9

Table 0-4 Likert Scale for SR004-Risk Analysis

Sample questions	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
To which extent this tool supports feature to	I/4					12

create risk diagram, or at-least include a feature similar to declare risks in efficient way?		☒	☐	☐	☐	
To which extent this tool supports feature to create threat diagram, or at-least include a feature similar to declare threats in efficient way?	II/3	☒	☐	☐	☐	9
To what extent this tool supports template to document risks?	III/2	☐	☐	☐	☒	0
To what extent this tool supports feature to create checklist of vulnerabilities, security threats and risks?	IV/1	☐	☐	☐	☒	0
						21

Table 0-5 Likert Scale for SR005-Secure Design Practices

Sample questions	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
To which extent this tool supports the feature to create security mechanism, or at-least include a feature similar to declare security mechanism in efficient way?	I/2	☒	☐	☐	☐	6
To what extent this tool supports the feature that can help selecting secure software architecture and secure software design?	II/1	☐	☐	☐	☒	0
						6

Table 0-6 Likert Scale for SR006-Justify Design Solution

Sample questions	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
To which extent this tool supports feature that can help calculating risk estimations?	I/3	☐	☐	☐	☒	0
To what extent this tool supports feature that can help categorizing and prioritizing security requirements?	II/2	☐	☐	☐	☒	0

To what extent this tool supports mechanism to estimate trade-off analysis?	III/1	☐	☐	☐	☒	0
						0

Table 0-7 Likert Scale for SR007-Response

Sample questions	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
To what extent this tool supports feature that can help creating response plan?	I/2	☐	☐	☒	☐	2
To what extent this tool supports feature to create templates for security requirements inspection?	II/1	☐	☐	☐	☒	0
						2

Table 11-5 Calculated Results for Magic Draw TOOL

Table 0-1 Likert Scale for SR001-Making Awareness

Total Score

99.00

Sample questions	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	Score
To which extent this tool provides online support, e.g. downloads and documentation?	I/6	☒	☐	☐	☐	18
To what extent this tools' book on methodology is explained and/or to what extent this tool adapts the methodology?	II/5	☐	☐	☒	☐	5
To what extent this tool provides support for the user manual?	III/4	☒	☐	☐	☐	12
To what extent this tool provides tutorials? That will support and provide learning opportunity.	IV/3	☒	☐	☐	☐	9
To what extent this tool has gained popularity	V/2					2

to support and provide several publications?		☐	☐	☒	☐	
To what extent this tool provides support for hands on wizards? These will guide engineers through several aspects of tool?	VI/1	☐	☒	☐	☐	2

48.00

Table 0-2 Likert Scale for SR002-Understand Context and Assets

Sample questions	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
To which extent this tool supports the feature to create asset diagram, or at-least include a feature similar to declare assets in efficient way?	I/3	☒	☐	☐	☐	9
To what extent this tool supports the feature that includes a template to document assets and goals?	II/2	☐	☐	☐	☒	0
To which extent this tool supports the feature to create checklist of assets and goals?	III/1	☐	☐	☐	☒	0

9

Table 0-3 Likert Scale for SR003-Security Requirements

Sample questions	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
To which extent this tool supports the feature to create treatment diagram, or at-least include a feature similar to declare treatments in efficient way?	I/3	☒	☐	☐	☐	9
To what extent this tool supports the feature that can help selecting security requirements elicitation techniques?	II/2	☐	☐	☐	☒	0
To what extent this tool supports template to document security requirements?	III/1	☐	☐	☐	☒	0

9

Table 0-4 Likert Scale for SR004-Risk Analysis

Sample questions	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
To which extent this tool supports feature to	I/4					12

create risk diagram, or at-least include a feature similar to declare risks in efficient way?		☒	☐	☐	☐	
To which extent this tool supports feature to create threat diagram, or at-least include a feature similar to declare threats in efficient way?	II/3	☒	☐	☒	☐	9
To what extent this tool supports template to document risks?	III/2	☐	☐	☐	☒	0
To what extent this tool supports feature to create checklist of vulnerabilities, security threats and risks?	IV/1	☐	☐	☐	☒	0
						21

Table 0-5 Likert Scale for SR005-Secure Design Practices

Sample questions	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
To which extent this tool supports the feature to create security mechanism, or at-least include a feature similar to declare security mechanism in efficient way?	I/2	☒	☐	☐	☐	6
To what extent this tool supports the feature that can help selecting secure software architecture and secure software design?	II/1	☐	☐	☐	☒	0
						6

Table 0-6 Likert Scale for SR006-Justify Design Solution

Sample questions	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
To which extent this tool supports feature that can help calculating risk estimations?	I/3	☐	☐	☐	☒	0
To what extent this tool supports feature that	II/2					0

can help categorizing and prioritizing security requirements?		☐	☐	☐	☒	
To what extent this tool supports mechanism to estimate trade-off analysis?	III/1	☐	☐	☐	☒	0
						0

Table 0-7 Likert Scale for SR007-Response

Sample questions	Priority /Score	Full Support	Above Average Support	Minimal Support	No Support	
To what extent this tool supports feature that can help creating response plan?	I/2	☒	☐	☒	☐	6
To what extent this tool supports feature to create templates for security requirements inspection?	II/1	☐	☐	☐	☒	0
						6

LICENSE

Non-exclusive license to reproduce thesis and make thesis public

1. I, Wajid Ali Khilji (Date of birth: 23-10-1985), herewith grant the University of Tartu a free permit (non-exclusive license) to:
    a. Reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and
    b. Make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

    Of my thesis.

Title,

(Evaluation framework for software security requirements engineering tools)

Supervised by,

(Dr. Raimundas Matulevičius)

2. I am aware of the fact that the author retains these rights.
3. I certify that granting the non-exclusive license does not infringe the intellectual property rights and rights arising from the Personal Data Protection Act.

Tartu, 06.11.2014