eSQUARE: A Formal Methods Enhanced SQUARE Tool

Hadil Abukwaik¹, Cui Zhang
Computer Science Department, California State University Sacramento, CA 95819
hadeel.abukwaik@informatik.uni-kl.de, zhangc@ecs.csus.edu

Abstract - This paper presents a formal methods enhanced tool named eSQUARE that supports the SQUARE security requirements engineering methodology developed by CMU SEI [7]. Development of security requirements has been neglected for a long time in the software development industry [9]. This has caused many errors and failures in the delivered software products and increased the cost on defect correction and product maintenance. Therefore it is important to have precise and concise specification of security requirements and have early integration of security requirements in software development process. The tool presented is aiming at helping practitioners to integrate the specification of security requirements using formal language Z [3, 6, 11] with the security requirement engineering methodology SQUARE.

Keywords: Security Requirements, Requirements Engineering, Formal Methods

1 Introduction

There is no doubt that software products are affecting almost every aspect of our daily lives. Software is found in electronic medical systems, online calls, bank ATMs, controllers of spaceships and rockets, etc. In such products, security requirements, including safety, gets a special importance as failures in satisfying the requirements may lead to a serious harm that can affect lives and money. However, if we look at the implemented security in software projects, we find only general mechanisms describing password protection, firewalls, virus detection tools, etc. [8]. This indicates the improper elicitation and inadequate development of the system-specific security requirements.

Requirements engineering plays a vital role in developing quality in software applications and in reducing the cost for correcting the defects in the released products. Generally, in order to meet software requirements successfully, all requirements need to be specified at the very beginning of the software development process. Unfortunately, this is not the case for security requirements. According to Mead et al. [7], “Studies show that upfront attention to security saves the economy billions of dollars. Industry is thus in need of a model to examine security and quality requirements in the development stages of the production lifecycle.” Therefore, a number of methods and techniques have been developed to ensure meeting the security requirements. One of these methods is the Security Quality Requirements Engineering (SQUARE) [7] which aims at integrating the security requirements engineering with the requirements engineering phase.

In addition, the more complex the software systems become the more precise and complete requirements specification is needed. As pointed out by Clark et al. [2], “One way of achieving this goal is by using formal methods, which are mathematics-based languages, techniques, and tools for specifying and verifying such systems.” Using formal methods will support the consistence and the correctness of software systems. In other words, it solves the ambiguity problem that can be faced when informal natural languages are used by many requirements engineers in eliciting and documenting requirements from the users and stakeholders. Not only this, but formally specified requirements are also absolutely better testable than informal ones. Although, using formal methods will add a reasonable advantage in expressing and testing the requirements it is not a guarantee of correctness [9, 12].

This paper presents a web-based tool named eSQUARE that fully supports the nine steps of SQUARE and uniquely enhanced using the formal methods based language Z [3, 6, 11] for modeling and checking elicited security requirements. The rest of the paper is structured as follows. Section 2 is a review on SQUARE methodology, formal methods based languages, and the related work. Section 3 illustrates the tool eSQUARE. Section 4 concludes the paper and presents future work.

2 Background and related work

Security Quality Requirements Engineering (SQUARE) is a methodology developed at Software Engineering Institute (SEI) of Carnegie Mellon University (CMU) [7]. This methodology helps in engaging security requirements in the early stages of the software development process

¹ Currently in Computer Science Department, University of Kaiserslautern, Germany 67663.
where requirements engineering activities take place. It has been proved to be useful for documenting and analyzing the security aspects of already developed systems and can direct the enhancements and the changes that can be applied to these systems in the future [9]. SQUARE process is formed by nine discrete steps [7]: (1) agree on definitions, (2) identify security goals, (3) develop artifacts, (4) perform risk assessment, (5) select elicitation techniques, (6) elicit security requirements, (7) categorize requirements, (8) prioritize requirements and (9) requirements inspection.

There are many formal methods based languages that can be used to specify the requirements of software systems. Representative formal languages include Z, VDM, Larch, and LOTOS [12]. As pointed out by Mead et al. [9], “Some useful techniques are formal specification approaches to security requirements, such as REVEAL and Software Cost Reduction (SCR), and the higher levels of the Common Criteria.” Automated tools that combine both a security requirements engineering methodology and a formal methods based language will be of great help in the development of secure software systems. We believe that developing a tool that can support the usage of a formal methods based language in the SQUARE methodology can provide a high confidence in the security of the developed software. This is the motivation of the development of the eSQUARE tool.

Although there are many methodologies and tools that support the security requirements engineering, the interest of this paper is particularly on supporting the SQUARE methodology and enhancing it with the formal methods. There are two existing tools serving the same focus.

mySQUARE was the first tool developed to support some steps of the SQUARE methodology to ease the management and administration of the process [14]. It enables tracking the progress of the work for each of the nine steps of the methodology, generating reports for users with the least time and effort, and using the XML technology to provide users with the portability of their files from one place to another. In spite of the great benefits mySQUARE offers, it has the following limitations: it is a stand-alone application that cannot be accessed from anywhere; it does not provide a built-in support for common techniques related to requirements elicitation, categorization and prioritization; and it does not provide import/export features for the project documents which many users may need [14].

In 2008, in conjunction with CyLab, SEI CMU developed and released the SQUARE prototype tool, workshop, tutorial, and educational materials which are useful in understanding the methodology [7, 10]. A number of papers present a light version of SQUARE called SQUARE-Lite and case studies about projects used it [4]. SQUARE-Lite is a five-step process taken from SQUARE. The technical approach of introducing SQUARE as part of the standard software life-cycle models is described in [10].

The team in CyLab built the SQUARE tool that supports the nine steps of SQUARE with a managerial interest. It takes care of managing the contributions among multiple requirement engineers and stakeholders. The SQUARE tool can be used in large companies working on large projects with a team for requirements engineering, however, it does not provide any formal methods based support for the specification of security requirements. Until the research stage for eSQUARE in early 2012, there are no official documentations or published papers for this type of tools.

3 The eSQUARE tool

3.1 The eSQUARE functionality

One of the vital factors of successful software projects is the effective cooperation between the stakeholders and the developers in general and the requirements engineers in particular. However, these two groups can be based in miles apart or even in different continents. In addition, the larger the project requirements, the harder the management becomes and the more effort it needs to take. Many reports and documents are produced in following the SQUARE methodology which means more time, space and management are needed. In response, eSQUARE has been developed to support the nine steps of the SQUARE methodology to overcome these challenges with a unique enhancement that introduces the usage of the formal language Z [3, 6, 11] for the specification of elicited security requirements. Starting a project in eSQUARE, users benefit from interfaces that enable them to:

- Go through the nine steps of the SQUARE with the preferred order along with the online help which can be accessed at any time.
- Add, edit and delete the following units: project terms, business goals, security goals, risks, elicitation techniques, requirements categories and inspection techniques. These units are the base of the SQUARE methodology. As the units can be repeated in different projects, eSQUARE enables reusing them by users who created them to save their time and effort.
- Set up the project scale components’ value, each risk likelihood, consequences, source and impact. These entries will be used by eSQUARE to calculate the risk value automatically and present it to users.
- Upload project security artifacts like security use cases and Z files and enable users to download or preview them as needed.
- Select the preferred formal methods based language from the four options offered: the standard Z, the Object Z, Circus or Z Rules.
• Select the preferred markup from the four options offered: Latex, UTF8, UTF16 and XML.
• Parse, check and export Z files that formally specify their security requirements. By this feature, eSQUARE offer a high confidence in the consistency and correctness of secure software systems.
• Un-check the type checking option if they do not want it.
• Get the structure of the Z sections that is written in the parsed Z files.
• Rank requirements based on categories and risk assessment results that have been computed by eSQUARE.
• Produce a summary for users’ projects.

In addition to all the mentioned functionalities, eSQUARE improves users’ experience in practicing the SQUARE methodology as they do not need to be professional requirements engineers to follow the nine steps. Users have the online help at each step where they can learn how to make progress in the process. This can reduce the project cost dedicated to security requirements engineering experts.

eSQUARE is a web-based tool designed to be flexible and user friendly so that users are capable of performing the steps in the order they prefer without restrictions. Another major advantage of the eSQUARE is the ability to upload the project’s artifacts in a central database. In this way, users are able to view their uploaded artifacts from any computer connected to the internet.

3.2 The eSQUARE architecture

The web-based eSQUARE tool is a thin-client program that receives users’ requests, performs all the processing on the server side and sends results to users as HTML. As shown in Figure 1, the multi-tier architecture pattern is chosen for eSQUARE as it provides good quality attributes to the system that will be described in the rest of the section. As seen in Figure 1 there are three basic components for the system:

- **The client machine:** This is where the web browser is hosted to enable users to start using the web-based tool.
- **The application server:** This server hosts the entire application files including HTML files, JSP files, Java files and the Community Z Tools (CZT) components’ files [3, 6].
- **The database server:** This can be the same application server or any other server where the MySQL database of the application is hosted.

One of the advantages in a multi-tiered application is that users are able to work on the application data without knowing where the data is stored at the build time. Another advantage is the modularity of the application’s components with loose coupling and high cohesion characteristics. In other words, this architecture provides better modifiability and extensibility for the application in the future. Also, the code is easier to read, understand, and re-use. In addition, this architecture has advantages in the performance of the system. That is, hosting components of the layers on different machines decreases the work load and increase the speed of response compared to applications having all the components hosted on one machine dealing with requests for all components. This is noticed when the application usage grows up and its traffic increases. Furthermore, the multi-tier architecture results in a robust application. From one point, whenever a change needs to be made in a tier it does not affect the other tiers and is independent. This again helps in re-using the components of the software application.

![Figure 1. eSQUARE software architecture](image_url)

3.3 The eSQUARE implementation

eSQUARE is implemented using the iterative life cycle model. In each iteration, a part of the requirements was selected based on its importance to the project, then implemented and tested. This approach has helped in monitoring the progress and planning in a simple and reliable way. eSQUARE is built using JSP (Java Server Pages) technology as it is portable (can be hosted in any operating system), easy to write and modular.

Like all technologies, JSP has its disadvantages too. The one identified during the implementation of eSQUARE is that there is a noticeable delay when we access the JSP page.
for the first time. In fact, JSP files are compiled on the server when they are accessed for the first time. This compilation causes a delay.

It’s important to mention that the open source CZT package [3] consists of many classes that build the Z tool which users can launch in the security requirement elicitation step provided by the eSQUARE tool. Despite of the number of tools available for the Z specification language, most of them do not support the ISO standard for this language [6]. Andrew Martin found that the many tools developed by school and academic were not complete and were not robust enough [6]. Based on that in 2001, he proposed the idea of the Community Z Tools to be a useful open source that supports the ISO specification of Z language. Integrating part of the CZT in the eSQUARE tool is of great advantage that allows the usage of existing tools to enhance the SQUARE methodology.

3.4 Example

In this section, the eSQUARE web-based tool is used to examine its efficiency in supporting the SQUARE methodology and in enhancing it with the formal methods based language Z in specifying the security requirements. All the data used in this example are taken from an in-depth case study performed by graduate students in Carnegie Mellon University under the supervision of Nancy Mead where the SQUARE methodology was applied on a product called Asset Management System (ASM) [1, 5]. Data are entered into eSQUARE to show how the tool can be used for managing the nine step process and how to use the integrated Z tool for writing the security requirements specification.

First of all a new project “ASM” is created to be used in our example. Terms that have been used in the ASM project were entered into eSQUARE under the first step of the process. These terms are available for users to use in other projects as they are associated to their accounts not to the project. In the second step, the business goal of ASM and its related security goals are entered to be in the system database and displayed to users. Many artifacts like use cases, misuse cases, architectural diagrams and figures are uploaded to eSQUARE each in a file that the user can download and review from any computer with internet access and a browser.

With the designed risk assessment methodology in eSQUARE, only the risk impact level is displayed near each risk of the ASM project as it does not provide weights for the required factors of the risk assessment method. The agreed technique for eliciting the security requirements in ASM was the Interactive approach. With this approach, names and descriptions are entered into the system and can be viewed and edited as users need.

In the second phase of the SQUARE methodology case study, the security requirements were refined and summarized in nine security requirements [5]. In eSQUARE these requirements are entered and saved successfully in the system. To examine the CZT component integrated in eSQUARE, a file of the Z specification in latex format is written for each of the nine security requirements of ASM. Figure 2 shows the Z specification for one security requirement of ASM called R_07, where its English description is [5]: It is a requirement that both process-centric and logical means be in place to prevent the installation of any software or device without prior authorization.

![Figure 2. Z Specification of ASM security requirement R_07](image-url)
After launching the CZT component, the written Z specification file is opened for parsing and type checking. In this example Standard Z language and Latex markup are selected and the files are both parsed and checked as seen in Figure 3.

After determining the option the component starts reading the file and gives feedback on the correctness of the specification. The Z file is examined twice; the first with its original correct specification and the next with an intended syntactical error. eSQUARE has passed this examination and responds correctly. Figure 4 shows the positive feedback given by eSQUARE for the correct version of the Z file along with its structure. As the file has been parsed and checked successfully, users are able to export it to different formats. A test is made and an export is performed on asm.tex to the UTF16 format and the result is a perfect Z file in the specified format. After examining the formal methods based features of eSQUARE, we go back to step seven of the SQUARE methodology case study. There were eight defined categories used in categorizing the security requirements of ASM. These categories have been defined in eSQUARE and are ready to be used in categorizing the nine security requirements of ASM.

eSQUARE uses integer numbers to express the priority of projects’ security requirements. However in the ASM case study the ninth security requirements were prioritized under three levels (Essential, Conditional and Optional) [5]. As a result, each of the three level is assigned a number and is used in eSQUARE (Essential: 1, Conditional: 2 and optional: 3).

Finally, inspection technique selected for the ASM project was the Peer Review technique. This technique has been inserted and selected for the ASM project efficiently. A Project Summary Form is generated once by clicking the last link in the Nine Steps Form.

Table 1 presents a summary of similarities and differences among eSQUARE, mySQAURE and CyLab SQUARE tools. The three tools share multiple features like supporting the administration of SQUARE projects, providing users with help option at any time of the work flow and generating automatic summaries for projects. Though, eSQUARE and CyLab SQUARE are web-based and support the risk assessment step, but mySQUARE does not.

4 Conclusion

The eSQUARE tool is developed with a strong belief that tool support should be sufficiently provided for efficient security requirements engineering methodologies like SQUARE. eSQUARE has a promising future for its unique integration of the formal methods based language Z with the SQUARE process. It can grow to be a robust and a powerful tool that security requirement engineers can rely on to increase their confidence in the security requirements specifications. As shown by the example, eSQUARE is simple and easy to use where users are capable of practicing the nine steps of the SQUARE methodology in an effective way.
not. A noticeable difference is that eSQUARE provides the support for the formal methods based language Z while mySQUARE and CyLab SQUARE do not.

Table 1. SQAURE Tools Comparison

<table>
<thead>
<tr>
<th>Feature</th>
<th>eSQUARE</th>
<th>mySQUARE</th>
<th>CyLab SQUARE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web Access</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>SQUARE Administrative Support</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Risk Assessment Support</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>Z language Support</td>
<td>✓</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Automatic Project Summary Generation</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>Automatic Reports Generation</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>Tool Accessible Documentation</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>User Help</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Team Work Support</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
</tbody>
</table>

Although the presented example in the previous section is built upon real data taken from a real project called ASM [1, 5], it doesn’t provide the empirical evaluation of the tool. The usage of the ASM data helped in getting the feeling on how to use eSQUARE forms and in its sequence, but there is still a need for an experiment or case study where eSQUARE get used along the security requirements engineering activities. Also, metrics for eSQUARE advantages should be defined and used in evaluating the tool.

5 References


