eSQUARE: A WEB-BASED & FORMAL-METHODS-ENHANCED SQUARE TOOL

Hadil K.M. Abukwaik
B.E., Al-Azhar University, Gaza, Palestine, 2007

PROJECT

Submitted in partial satisfaction of
the requirements for the degree of

MASTER OF SCIENCE

in

SOFTWARE ENGINEERING

at

CALIFORNIA STATE UNIVERSITY, SACRAMENTO

SPRING
2010
eSQUARE: A WEB-BASED & FORMAL-METHODS-ENHANCED SQUARE TOOL

A Project

by

Hadil K.M. Abukwaik

Approved by:

___________________________, Committee Chair
Dr. Cui Zhang

___________________________, Second Reader
Dr. Ahmed Salem

___________________________
Date
Student: Hadil K.M. Abukwaik

I certify that this student has met the requirements for format contained in the University format manual, and that this project is suitable for shelving in the Library and credit is to be awarded for the Project.

__________________________, Graduate Coordinator    ________________________
Dr. Cui Zhang                                                                      Date

Department of Computer Science
Abstract

of

eSQUARE: A WEB-BASED & FORMAL-METHODS-ENHANCED SQUARE TOOL

by

Hadil K.M. Abukwaik

Development of security requirements has been neglected for a long time in the software development industry [1]. This has caused many errors and failures in the delivered products and increased the cost spent on correcting the defects and maintaining the products. It is important to have early integration of security requirements in software development life cycle (SDLC) and get the benefits of using formal methods in specifying security requirements. This report presents the SQUARE methodology and the development of the eSQUARE tool that automates the nine steps of the methodology. The eSQUARE is a web-based and formal-methods-enhanced tool that integrates the specification of security requirements using the formal methods based language Z with the security requirement engineering methodology SQUARE.

_________________________, Committee Chair
Dr. Cui Zhang

_________________________
Date
DEDICATION

This project is dedicated to my wonderful parents, Kamal and Hedaya, who have raised me to be the person I am today. The beauty is not in reaching the destination, but in the journey we go through toward it with our beloved ones. Thank you for all the unconditional love and support which provide me with the confidence that I am capable of doing anything I put my mind to. Thank you for everything. I love you!
ACKNOWLEDGEMENT

The success of this project depends largely on the encouragement and guidance I got from many people. I take this opportunity to express my gratitude to them in my humble acknowledgment.

In the first place I would like to record my gratitude to Dr. Cui Zhang for her supervision, advice, and guidance with her extraordinary experience. Her wonderful classes are a constant oasis of ideas and passions in software engineering world. Above all and the most needed, she provided me unflinching encouragement and support in various ways. I am indebted to her more than she knows.

I gratefully acknowledge Dr. Ahmed Salem for his interactive classes which exceptionally inspire and enrich my growth as a student, a researcher and a software engineer. I want to thank him for giving from his precious time to read this project and give his comments about it.

Many thanks go in particular to the Fulbright Program for giving me this valuable scholarship that gave me the opportunity to pursue my graduate studies at California State University.

All my professors, fellow graduate students and friends in Sac State, thank you! The last two years have been quite an experience and you have all made it a memorable time of my life. All the best to each of you in your future endeavors.

Where would I be without you Dad and Mom? My parents deserve special mention for their never-ending love, inseparable encouragement and prayers. Thank you for supporting me all the way, and for giving me the foundation to be who I am.

My brothers, Mahmoud and Mohammed, and my sisters, Yasmeen and Shahd, thank you for the continual support and love through my journey. Thank you for the laughing and the fighting, and everything in between.

Words fail me to express my appreciation to Mohammed Abufouda whose dedication, care and persistent confidence in me, has taken the load off my shoulder. I owe him for keeping my spirits up when this project seemed interminable.

Finally, I would like to thank everybody who was important to the successful realization of the project, as well as expressing my apology that I could not mention personally one by one.
# TABLE OF CONTENTS

Dedication .......................................................................................................................... vi

Acknowledgement ............................................................................................................. vii

List of tables ....................................................................................................................... x

List of figures ...................................................................................................................... xi

Chapter

1. INTRODUCTION ......................................................................................................... 1

2. BACKGROUND AND RELATED WORK ........................................................................ 5
   2.1 Need of Security Requirements Engineering ............................................................ 5
   2.2 SQUARE Methodology .............................................................................................. 5
   2.3 mySQUARE Tool ....................................................................................................... 7
   2.4 CyLab SQUARE ......................................................................................................... 8
   2.5 Formal Methods ......................................................................................................... 9
   2.6 eSQUARE ................................................................................................................ 10

3. ESQUARE ANALYSIS, ARCHITECTURE AND DESIGN ........................................... 12
   3.1 eSQUARE Functional and Non-Functional Requirements ........................................... 12
   3.2 eSQUARE Architecture .......................................................................................... 15
   3.3 eSQUARE Software Design ................................................................................... 19
   3.4 eSQUARE Interface Design ..................................................................................... 24

4. eSQUARE IMPLEMENTATION .................................................................................. 77

5. EXAMPLE ...................................................................................................................... 88
6. CONCLUSION........................................................................................................... 109

References..................................................................................................................113
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Table 1. SQAURE Automation Tools Comparison</td>
</tr>
</tbody>
</table>

# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Figure 1. eSQUARE Architecture</td>
<td>16</td>
</tr>
<tr>
<td>2.</td>
<td>Figure 2. eSQUARE Design</td>
<td>20</td>
</tr>
<tr>
<td>3.</td>
<td>Figure 3. eSQUARE Home Page</td>
<td>25</td>
</tr>
<tr>
<td>4.</td>
<td>Figure 4. eSQUARE Registration Form</td>
<td>26</td>
</tr>
<tr>
<td>5.</td>
<td>Figure 5. eSQUARE Main Services Form</td>
<td>27</td>
</tr>
<tr>
<td>6.</td>
<td>Figure 6. Create eSQUARE Project Form</td>
<td>28</td>
</tr>
<tr>
<td>7.</td>
<td>Figure 7. Open Project Form</td>
<td>29</td>
</tr>
<tr>
<td>8.</td>
<td>Figure 8. Project Nine Steps Form</td>
<td>30</td>
</tr>
<tr>
<td>9.</td>
<td>Figure 9. Terms Main Menu Form</td>
<td>31</td>
</tr>
<tr>
<td>10.</td>
<td>Figure 10. Add Term Form</td>
<td>32</td>
</tr>
<tr>
<td>11.</td>
<td>Figure 11. Select Term Form</td>
<td>33</td>
</tr>
<tr>
<td>12.</td>
<td>Figure 12. Edit Term Form</td>
<td>34</td>
</tr>
<tr>
<td>13.</td>
<td>Figure 13. Delete Term Form</td>
<td>35</td>
</tr>
<tr>
<td>14.</td>
<td>Figure 14. Goals Main Menu</td>
<td>36</td>
</tr>
<tr>
<td>15.</td>
<td>Figure 15. Add Goal Form</td>
<td>37</td>
</tr>
<tr>
<td>16.</td>
<td>Figure 16. Select Goal Form</td>
<td>38</td>
</tr>
<tr>
<td>17.</td>
<td>Figure 17. Edit Goal Form</td>
<td>39</td>
</tr>
<tr>
<td>18.</td>
<td>Figure 18. Delete Goal Form</td>
<td>40</td>
</tr>
<tr>
<td>19.</td>
<td>Figure 19. Artifacts Main Menu Form</td>
<td>41</td>
</tr>
</tbody>
</table>
20. Figure 20. Add Artifact Form ................................................................. 42
21. Figure 21. Select Artifact Form ............................................................. 43
22. Figure 22. Delete Artifact Form .............................................................. 44
23. Figure 23. Risk Main Menu Form ............................................................ 45
24. Figure 24. Risk Scale Form ................................................................. 46
25. Figure 25. Add Risk Form ................................................................. 47
26. Figure 26. Select Risk Form ................................................................. 48
27. Figure 27. Edit Risk Form ................................................................. 49
28. Figure 28. Delete Risk Form ................................................................. 50
29. Figure 29. Elicitation Techniques Main Menu Form ................................ 51
30. Figure 30. Add Elicitation Technique Form ............................................ 52
31. Figure 31. Select Elicitation Technique Form ......................................... 53
32. Figure 32. Edit Elicitation Technique Form ............................................ 54
33. Figure 33. Delete Elicitation Technique Form ......................................... 55
34. Figure 34. Requirements Main Menu Form ............................................ 56
35. Figure 35. Add Requirement Form ....................................................... 57
36. Figure 36. Z Tool Form ...................................................................... 58
37. Figure 37. Z Community Tool ............................................................... 59
38. Figure 38. Select Requirement Form .................................................... 60
39. Figure 39. Edit Requirement Form ....................................................... 61
40. Figure 40. Delete Requirement Form .................................................... 62
62. Figure 62. Creating ASM Project ................................................................. 89
63. Figure 63. ASM Security Definitions ........................................................... 90
64. Figure 64. ASM Business and Security Goals .............................................. 91
65. Figure 65. ASM Artifacts ........................................................................... 92
66. Figure 66. ASM Risk Assessment ............................................................... 93
67. Figure 67. ASM Selected Elicitation Technique ......................................... 94
68. Figure 68. ASM Elicited Security Requirements ....................................... 95
69. Figure 69. Z Specification For ASM Security Requirement R_07 .............. 96
70. Figure 70. Opening a Z File in CZT component .......................................... 97
71. Figure 71. Selecting a Language ................................................................. 98
72. Figure 72. Selecting a Markup ................................................................ 98
73. Figure 73. CZT Feedback on Incorrect Z Specifications .......................... 99
74. Figure 74. CZT Feedback on Correct Z Specification .............................. 100
75. Figure 75. Export Z File to Different Formats ........................................... 101
76. Figure 76. U16 Export Result ................................................................ 102
77. Figure 77. ASM Eight Security Requirements Categories ...................... 103
78. Figure 78. Categorizing ASM Security Requirements ............................. 104
79. Figure 79. ASM’s Security Requirements Prioritization ............................ 105
80. Figure 80. Defining the Peer Review Technique in eSQUARE ............... 106
81. Figure 81. Selecting Peer Review Technique for ASM Project .............. 107
82. Figure 82. Project Summary .................................................................. 108

xiv
Chapter 1

INTRODUCTION

No one can deny the vital role that requirements engineering plays in developing quality software applications and in reducing the cost of correcting the defects in the released products in terms of time, effort and money. However, if we look at security requirements included in projects that need security, we find them describing password protection, firewalls, virus detection tools,… etc [3]. This indicates the improper elicitation for the security requirements which also may be independent and not integrated in the software development life cycle activities. According to Dr. Mead [2], “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.”

In addition, the more complex the software systems become day after day, the more precise and correct requirements specification is needed especially in systems where security requirements can affect lives, money and time. As pointed out by Mr. Clark [4], “One way of achieving this goal is by using formal methods, which are mathematically based languages, techniques, and tools for specifying and verifying such systems.” Using formal methods will support the consistency, correctness and clear deep understanding of the software system. In other words, it solves the ambiguity problem
that can be faced using the informal natural language used by many requirements engineers in eliciting and documenting the requirements from the users and stakeholders. Although, using formal methods will add a reasonable advantage in expressing the requirements it is not a guarantee of correctness [1].

Security Quality Requirements Engineering (SQUARE) is a methodology developed at Carnegie Mellon University by Professor Nancy Mead with Professor Donald Firesmith and Professor Carol Woody in a research project in the Software Engineering Institute (SEI) [2]. This methodology helps in engaging security requirements in the early stages of the software development life cycle. This methodology includes nine steps through which requirements engineers work with stakeholders on eliciting, categorizing, and prioritizing security requirements for software systems and applications [2]. SQUARE also 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 [1]. Although using the SQUARE methodology will be of a great benefit for integrating the security to software systems as a part of process rather than adding it after the development finishes which will be almost not useful, the amount of work that needs to be done through this process will add more cost to the projects that may not be affordable, e.g. time limitation. Also, it needs expertise in requirements engineering and communication with the stakeholders which adds another layer of difficulty.
In response to these difficulties, an application called mySQUARE has been developed to semi-automate some steps of the SQUARE methodology to ease the management and administration of the process [5]. 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 feature for the project documents as the user may need them [5].

After reading many published papers and case studies, I reached a belief that implementing an automated tool that covers the limitations mentioned previously in the mySQUARE tool and includes the usage of formal methods for security requirements specification is of invaluable advantage for secure software systems development industry. From this belief I, under the supervision of Dr. Cui Zhang, started developing a tool called eSQUARE. Indeed, eSQUARE is the first web-based tool that automates the nine steps of SQUARE methodology and supports writing the security requirements specification in the formal methods based language Z. In this way, eSQUARE is uniquely providing a higher confidence in the security of the developed software. In addition, this provides a high confidence in the consistency and correctness of software systems. Also, the tool improves users’ experience in following the methodology. Moreover, users are
able to access the tool from any place with an internet connection which makes it easier and more flexible for them.

Chapter 2 discusses the background and related work of this project. We will have a closer look on how eSQUARE helps improving the security requirement engineering process. Chapter 3 includes the description of the requirements, system architecture and design of eSQUARE. Chapter 4 documents the implementation details used in building eSQUARE. Chapter 5 presents an example that shows the benefits of using eSQUARE compared to the manual approach. Finally, we conclude the report with a summary and future work for eSQUARE.
Chapter 2
BACKGROUND AND RELATED WORK

2.1 Need of Security Requirements Engineering

There is no doubt that software products are playing a vital role in our daily lives and affects almost every aspect. We find software in electronic medical machines, online calls, bank ATMs, controllers of spaceships and rockets, …etc. It is obvious that security is becoming more and more important in the developed software products. As we know, in order for the software to meet its requirements successfully, these requirements need to be specified early in the software engineering process. Unfortunately, this is not the case in security requirements!

Therefore, a number of methods and techniques have been developed to ensure meeting the security requirements, for example: Comprehensive, Lightweight Application Security Process (CLAPS), System Quality Requirements Engineering (SQUARE), and Core Security requirements artifacts [1].

2.2 SQUARE Methodology

The SQUARE methodology is developed to help in engaging security requirements in the early stages of the software development life cycle. This methodology includes nine steps through which requirements engineers work with
stakeholders on eliciting, categorizing, and prioritizing security requirements for software systems and applications [2]. SQUARE has also 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 [1].

SQUARE can be decomposed into nine discrete steps, each step has necessary inputs, major participants, suggested techniques, and final output. These steps can be summarized as the following:

**Step 1:** Agree on definitions: meeting the stakeholders to get consistent definitions for the security requirements definitions.

**Step 2:** Identify security goals: determining security goals from the different area stakeholders and prioritizing them.

**Step 3:** Develop artifacts: supporting all the subsequent activities.

**Step 4:** Perform risk assessment: using a method to assess the risks and identifying the high-priority security exposures.

**Step 5:** Select elicitation techniques: selecting what technique to use for eliciting security requirements either from one stakeholder or different-class of stakeholders.

**Step 6:** Elicit security requirements: using selected technique in actual eliciting of security requirements.

**Step 7:** Categorize requirements: determining whether they are requirements, goals, or constraints.
**Step 8:** Prioritize requirements: giving priority number for the security requirements based on Step 8 and the cost analysis of these requirements.

**Step 9:** Requirements inspection: inspecting the resulted documentation may be done with different level of formality.

This methodology is of a great benefit for integrating the security to software systems as a part of process rather than adding it after the development finishes which will be almost not useful. However, the amount of work that needs to be done through this process will add more cost to the projects that may not be affordable.

### 2.3 mySQUARE Tool

mySQUARE [5] is the first tool that semi-automates some steps of the SQUARE methodology to ease managing and administrating it. Using mySQUARE has many advantages. Some of them are the following [5]:

- It enables tracking the progress of the work for each of the nine steps of SQUARE methodology.
- It gives users the ability to perform the steps in the order they decided to follow.
- It generates reports for users with the least time and effort.
- It gives users the option to save all the related information of a project in one central location.
- It offers a helping hand for users by posting notes and comments through the project life.
- It uses the XML technology to provide users with the portability of their files from one place to another.

Although mySQUARE has many advantages, it can be enhanced in some areas if [5]:
- It provides a built-in support for common techniques related to requirements elicitation, categorization and prioritization.
- It provides a built-in support for building popular security artifacts like misuse cases.
- It enables the online access to the application using the internet so users can use it from anywhere on the network.
- It provides import/export features for the project documents as users may need them.

2.4 CyLab SQUARE

In 2008, Carnegie Mellon University (CMU) in conjunction with CyLab developed and released the SQUARE prototype tool, workshop, tutorial, and educational materials which are useful in understanding the methodology [2]. A number of papers present a light version of SQUARE called SQUARE-Lite and case studies about projects used it [9]. SQUARE-Lite is a five-step process taken from SQUARE. Another important paper describes the technical approach of introducing SQUARE as part of the standard software life-cycle models [12].
At the time I started searching and studying the work completed in the area of security requirements engineering, CyLab website had a plan for 2009 to implement a tool that automates SQUARE without any description of this tool. As I was working on my web-based tool eSQUARE, at the beginning of 2010 CyLab tool was completed but not available to the public. I requested Dr. Nancy Mead to provide me with an authorized account to have a look on what they developed. Thanks to her collaboration I was able to log in and I was not surprised when I found that the team in CyLab have automated the nine steps of SQUARE with a managerial interest. Also, the web-based tool takes care of managing the contributions among multiple requirement engineers and stakeholders. Obviously, the CyLab tool can be used in large companies working on large projects with a team for requirements engineering, however, it does not support engaging any of the formal methods based languages in specifying the security requirements. Until now there is not any official documentation or published paper for this tool.

2.5 Formal Methods

There are many formal methods based languages that can be used to specify the requirements of software systems with some differences. But they all use mathematical concepts of abstraction and composition. For example Z, VDM, and Larch focus on specifying the behavior of sequential systems. Described states are in terms of rich mathematical structures such as sets, relations, and functions; state transitions are given in terms of both pre and post-conditions. However, RAISE and LOTOS are two different
methods, the first for handling rich state spaces and the second for handling complexity due to concurrency [7].

As Dr. Mead said [1], “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 a great achievement in the Secure Software Systems Development. In addition such tools will improve the experience of the software developers in this field using the automated guidance and directions. Based on what we have seen so far, I believe that developing a tool that can support the usage of a formal methods based language in the SQUARE methodology will be unique, strong and provide a higher confidence in the security of the developed software. This is the basis of my web-based tool eSQUARE that is presented in the next chapter.

2.6 eSQUARE

Despite of the benefits that can be obtained through the nine steps of the SQUARE methodology that integrates defining the security requirements early in the software development life cycle, it still can be viewed as a long and difficult to manage process especially in large and complex projects. Not only it needs management efforts, but it also requires expertise in requirements engineering. In addition, precise and correct security requirements specification is needed especially for safety-critical and/or
financial-critical systems. One way of achieving this goal is by using formal methods based languages, techniques, and tools for specifying and verifying such systems.

In order to cover all these concerns the web-based tool eSQUARE has been developed to automate the nine steps of SQAURE with an enhancement that introduces the usage of the formal methods based language Z in specifying the elicited security requirements. Within eSQUARE users do not need to be professional requirements engineers to follow the automated nine steps. They have the online help at each step where they can learn how to progress and carry on in the process. eSQUARE is 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 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 no matter where it is. Moreover, eSQUARE enables users to parse and check their Z files that formally specifies their security requirements and upload them to the database.
Chapter 3

ESQUARE ANALYSIS, ARCHITECTURE AND DESIGN

3.1 eSQUARE Functional and Non-Functional Requirements

It is undeniable that one of the vital factors of successful software projects is the effective cooperation between the stakeholders and the developers generally and with the requirements engineers specifically. 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. Moreover, many reports and documents are produced in following the SQUARE methodology which means more time, space and management are needed. In response, eSQUARE automates the SQUARE methodology to overcome these challenges. eSQUARE provides users with a set of helpful features which are defined bellow.

**eSQUARE Functional Requirements:**

- Create user account and edit account.
- Login to the tool.
- Add, edit and delete projects.
- SQUARE steps for each project:
  - **STEP 1:**
    - Add, edit and delete terms.
- Select terms in project.

  o STEP 2:
    - Add, edit and delete business goals.
    - Add, edit and delete security goals.

  o STEP 3:
    - Add (upload files) and delete artifacts.
    - Download and view uploaded artifacts.

  o STEP 4:
    - Define project scale components’ values.
    - Add, edit and delete risks.
    - Assign each risk its likelihood, consequences, source, impact and edit them.
    - Calculate each risk value based on the scale defined.

  o STEP 5:
    - Add, edit and delete elicitation techniques.
    - Select wanted requirements elicitation techniques and describe the reason.

  o STEP 6:
    - Document security requirements (name and description).
    - Optional usage of the formal method language Z features that parse, check and export the Z files.
- Upload Z files to database.

  o STEP 7:
    - Add, edit and delete requirements categories.
    - Categorize security requirements.
    - Edit categorization.

  o STEP 8:
    - Rank requirements based on categories and risk assessment results.

  o STEP 9:
    - Add, edit and delete inspection techniques.
    - Select inspection techniques needed by the project.

- Produce a summary for users’ projects.

- Provide online help at each point of the process.

  eSQUARE also addresses a couple of non-functional requirements that aims at producing a software of good quality.

**eSQUARE Non-Functional Requirements:**

- *Usability*: tool is friendly, easy to use and has consistent forms.

- *Re-usability*: Java Object-Oriented classes can be used in other projects.

- *Modifiability*: it is easy to make changes by abstracting and by separating layers.

- *Portability*: system can be hosted on servers with different platforms by implementing it with the JSP technology.
• Availability: wherever there is internet connection and web browser users are able to access the tool.

3.2 eSQUARE Architecture

Software applications started with the standalone applications which also known as desktop applications. Years after, web-based applications started to take their place in software development world. Many papers and studies compare them versus each other and try to explain advantages and disadvantages based on the need of the applications. They also discuss different quality attributes in both types of applications, but the main reason that made me choose to build a web-based tool is to enable users to access their project from any computer connected to the internet and has a web browser on it. In this way, there is a freedom to move without being bound to a specific computer where user initiate and work on their project through eSQUARE. Add to this that all the processing is done on the server and sent to users as HTML results, in this way eSQUARE is called a thin-client program.

Like most software systems no matter standalone or web-based systems, eSQUARE is a collection of components that have relationships that interact with each other to accomplish a specific functionality. In order to organize these components and model their communication, a software system architecture should be selected. Based on my study in Csc 235 “Software System Architecture” with Dr. Cui Zhang at California State University and my further reading in software patterns, I found that the multi-tier
architecture is the best for eSQUARE (see Figure 1) as it provides good quality attributes to the system as described next.

![eSQUARE Architecture](image_url)

Figure 1. eSQUARE Architecture

As we can see in Figure 1 there are three basic components for the system:

- **The client machine**: this is where the web browser is hosted and enables users to start using the web-based tool.
• **The application server:** this server is hosting the entire application files including HTML files, JSP files, Java files and the CZT files.

• **The database server:** this can be the same application server or any other server where the MySQL database of the application is hosted.

Having a closer look inside the tiers of the system, we find that its components in the application tier and the database tier are using the multi-tier architecture in the following manner:

• **Presentation Layer:** this layer includes all the HTML and JSP pages displayed for users who enter inputs and/or send commands and receive HTML results through this layer.

• **Application Layer:** this layer includes the core logic of the application that binds the representation layer with the data layer and controls the communication between them. In eSQUARE, this layer is divided into two sub layers as following:

  o **Business Logic Tier:** this tier receives inputs and commands from the presentation layer, invokes the needed functionality from the data access tier and/or the Community Z Tools (CZT) components, and finally arranges the html output sent to users. In eSQUARE, this tier is represented by the JSP pages hosted on the web server.
- **Data Access Tier**: this tier receives requests from the business logic tier for accessing data from the data source, retrieves the data, processes if needed and sends it back to the business logic tier. In eSQUARE, this is represented in the JavaBeans hosted on the web server.

- **CZT external components**: this represents the community of Z tools’ open source Java packages that are invoked by the business logic tier components.

- **Data Tier**: this tier includes the data source, usually the database itself and the database management system. In eSQUARE the database is hosted on the MySQL server.

One of the advantages in a multi-tiered application is that users are able to work on the application data without knowing at the build time where the data is stored [6]. To make this level of transparency possible, the underlying JSP pages in a multi-tiered architecture use commands to create instances of Java classes (Java Beans) that interact with the database and move data from and to the client all through the JSP pages [6].

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 understand, read 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 [7].

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 effect the other tiers and is independent. For example, if a change is made in the presentation tier neither the application tier nor the data tier need to be changed. In the same way if a change happens in the data tier neither application tier nor presentation is effected [7]. This again helps in re-using the components of the software application.

3.3 eSQUARE Software Design

In this section we describe eSQUARE design in detail. As seen in Figure 2, eSQUARE design is built from three primary component:

- **JSP and HTML Forms:**
  In this component, a set of presented forms serve as an initial point of interaction between users and eSQUARE. In other words, users issue commands and request functionalities from eSQUARE through these forms that contains HTML code with JSP code embedded in it. Some of the forms that belongs to this component
are: LoginForm, SubscribeForm, AddProjectForm, DeleteProjectForm, AddGoalForm, EditRequirementForm, UploadArtifactForm, and more.

Figure 2. eSQUARE Design
• **JSP Logic:**

In this component we find part of the business logic work of the system presented in the JSP files. These files receive inputs and commands from the previous component’s forms and they deliver the output back that is presented to users. Some of what belongs to this component are: Login, AddCategory, DeleteElicitationTechnique, Logout, CZT, EditUserAccount, DownloadArtifact, CategorizeRequirement, and more.

• **Java Packages and Classes:**

In this component we find the rest of the business logic work of the system presented in the Java packages with its classes. These classes include methods that are invoked by the previous component’s JSP files. Actually, these Java classes access the data in database and perform the request operation on them. We have three basic packages in eSQUARE:

  o **General Package:** this package has two classes:

    ▪ **Database class:** responsible for creating the connection to MySQL database, processing sql queries and commands sent from other classes, and closing the connection.

    ▪ **FileBean class:** responsible for operations performed on the uploads table in MySQL database including adding, downloading and deleting files.
- **User Package**: this package has two classes:
  - **UserBean class**: responsible for operations performed on the users table in MySQL database including adding, editing and deleting accounts. Also, it is responsible of retrieving object related to a specific user like projects, terms, elicitation techniques, … etc.
  - **LoginBean class**: responsible for managing the login and logout commands. Also, it has a field used to verify the authentication to access pages in sessions.

- **Project Package**: this package has nine classes:
  - **CategoryBean class**: responsible for operations performed on the categories table in MySQL database including adding, editing and deleting categories.
  - **ElicitationBean class**: responsible for operations performed on the elicitation_techniques table in MySQL database including adding, editing and deleting elicitation techniques.
  - **GoalBean class**: responsible for operations performed on the project_goals table in MySQL database including adding, editing and deleting goals.
  - **InspectionBean class**: responsible for operations performed on the inspection_techniques table in MySQL database including adding, editing and deleting inspection techniques.
- **ProjectBean class**: responsible for operations performed on the projects table in MySQL database including adding, editing and deleting projects.

- **RequirementBean class**: responsible for operations performed on projects_requirements table in MySQL database including adding, editing and deleting requirements.

- **RiskBean class**: responsible for operations performed on the project_risks table in MySQL database including adding, editing and deleting risks.

- **ScaleBean class**: responsible for inserting and editing values of the factors used in the calculating the risk values in the project_scale table in MySQL database.

- **TermBean class**: responsible for operations performed on the terms table in MySQL database including adding, editing and deleting terms.

Obviously the design says a lot about the flow of the work and how it starts at the HTML forms that pass the users’ input and commands to the JSP Logic files which invokes the Java classes’ methods that operate on the database tables and finally return the output to the JSP logic files which pass it to users presented in the HTML forms. For example, AddRiskForm belongs to the first component and is presented to the user through the browser allowing the user to fill the form with the risk name and its
description then send a request to insert them in the database. This request is sent to its corresponding JSP Logic file called AddRisk.jsp which processes the input and the request by invoking the method checkNEW( ) in the Java class RiskBean. Then, the checkNew( ) inserts the new entry in the database and sends the success/failure information back to AddRisk.jsp. Based on this, AddRisk.jsp decides whether to display an error message or a success message to the user through AddRiskForm.

3.4 eSQUARE Interface Design

Within eSQUARE, there are several screens that users might use or might interact with. In this section, we present most of these screens and describe the intended purpose of them.

As shown in Figure3, eSQUARE Home Page is designed to give users the ability to login using usernames and passwords or to create new account. It also offers information about SQAURE methodology, the formal based language Z and the eSQUARE tool.
Figure 3. eSQUARE Home Page
Figure 4. eSQUARE Registration Form

Users need to fill the Registration Form to be able to get access to the services provided by eSQUARE (see Figure 4). Fields need to be filled in the right format or an error message is displayed after pressing the submit button. If fields are valid a confirmation message is displayed and users are redirected to the main eSQUARE services page.
Figure 5. eSQUARE Main Services Form

Figure 5 shows the options users can choose to create a new project, open and work on an existing project, or edit their account information by clicking one of the available links in the Main Service Form.
Users fill the Create Project Form with the project name and description and press the Save button as shown in Figure 6. If the project name already exists in the database for the same user a duplication error message is displayed otherwise a success message is displayed and users redirected to the Main Services Menu.
Users in the Open Project Form, presented in Figure 7, select the project that they want to work on from the menu by pressing the Go To Project Form and they are redirected to the Nine Steps Form of the project they selected.
Figure 8. Project Nine Steps Form

Users can change the project name, status, owner and description in the Nine Steps Form as shown in Figure 8. Also, they select one of the nine steps’ links to get started and work on the selected step.
Looking at Figure 9 we can see the Terms Main Menu Form that users reach by clicking the first step link in the Nine Steps Form. They have the choice to add, edit or delete their own terms that will appear on all projects and select the ones they need for each project from the edit link.
Users fill the Add Term Form seen in Figure 10 with the name and description and hit the Save button. If the term name already exists in the database for the same user a duplication error message is displayed otherwise a success message is displayed and users redirected to the Terms Main Menu.
Users get to the Select Term Form by clicking the Edit Term link in the Term Main Menu Form (see Figure 11). They select the term they want to change or select by pressing the Go To Term button that leads to the next described form.
As we see in Figure 12, users in Edit Term Form can change the details about the selected term or/and select it or deselect it from the project then they hit the Edit button. Input is checked, if valid users get a success message and redirected to the Term Main Menu otherwise an error message is displayed.
Users get to the Delete Term Form by clicking the Delete Term link in the Term Main Menu Form. They select the term they want to delete by pressing the Delete button. Figure 13 depicts this feature. After deleting they are redirected to the Terms Main Menu.
Figure 14. Goals Main Menu

Users get to the Goals Main Menu Form by clicking the second step link in the Nine Steps Form. They have the choice to add, edit or delete their project goals (see Figure 14).
Users fill the Add Goals Form, displayed in Figure 15, with the type (Security or Business) and description and press the Save button. If the goal description already exists in the database for the same project a duplication error message is displayed otherwise a success message is displayed and users redirected to the Goals Main Menu.
Figure 16. Select Goal Form

Users get to the Select Goal Form by clicking the Edit Goal link in the Goal Main Menu Form. They select the goal they want to change by pressing the Go To Goal button that leads to the next described form (see Figure 16).
Users in Edit Goal Form can change the details about the selected goal and then they hit the Edit button as depicted in Figure 17. Input is checked, if valid users get a success message and redirected to the Goal Main Menu otherwise an error message is displayed.
Users get to the Delete Goal Form by clicking the Delete Goal link in the Goal Main Menu Form. They select the goal they want to delete by pressing the Delete button. After deleting they are redirected to the Goals Main Menu Form (see Figure 18).
Figure 19. Artifacts Main Menu Form

Users get to the Artifacts Main Menu Form by clicking the third step link in the Nine Steps Form. They have the choice to add, open or delete their own artifacts.
Users in the Add Artifact Form browse the artifact that they want to upload to the system database as seen in Figure 20. If the selected artifact has the same name as another artifact that is already uploaded for the project a duplication error message is displayed otherwise a success message is displayed and users will be redirected to the Artifacts Main Menu Form.
Users get to the Select Artifact Form, which is shown in Figure 21, by clicking the Open Artifact link in the Artifacts Main Menu Form. They select the artifact they want to download from the database by pressing the Download this File link that leads to downloading the artifact selected in the specified folder.
Users get to the Delete Artifact Form by clicking the Delete Artifact link in the Artifact Main Menu Form (see Figure 22). They select the artifact they want to delete by pressing the Delete File link. After deleting, they are redirected to the Artifacts Main Menu Form.
Figure 23. Risk Main Menu Form

Users get to the Risks Main Menu Form by clicking the fourth step link in the Nine Steps Form. They have the choice to define scale values, add, open or delete their project risks as depicted in Figure 23.
Users get to the Risks Scale Form, seen in figure 24, by clicking the Project Risk Scale link in the Risk Main Menu Form. They fill the form with the integer values and hit the Save Scale button. If values are valid, a success message is displayed and users are redirected to the Risks Main Menu Form otherwise an error message is displayed.
Users fill the Add Risk Form with the description, source and selecting the values for Likelihood, Impact and Consequences from the drop down menu then they hit the Save button (see Figure 25). If the risk description already exists in the database for the same project a duplication error message is displayed otherwise a success message is displayed and users are redirected to the Risk Main Menu Form.
Figure 26. Select Risk Form

Users get to the Select Risk Form by clicking the Edit Risk link in the Risk Main Menu Form. They select the risk they want to change by pressing the Go To Risk button that leads to the next described form (see Figure 26).
Users in Edit Risk Form, seen in Figure 27, can change the details about the selected risk and then they hit the Edit button. Input is checked, if valid users get a success message and redirected to the Risk Main Menu otherwise an error message is displayed.
Users get to the Delete Risk Form by clicking the Delete Risk link in the Risk Main Menu Form as seen in Figure 28. They select the risk they want to delete by pressing the Delete button. After deleting they are redirected to the Risks Main Menu Form.
Users get to the Elicitation Techniques Main Menu Form, depicted in Figure 29, by clicking the fifth step link in the Nine Steps Form. They have the choice to add, edit or delete their own elicitation techniques that will appear on all projects and select the ones they need for each project from the edit link.
Users fill the Add Elicitation Technique Form with the name and description and hit the Save button (see Figure 30). If the name already exists in the database for the same user a duplication error message is displayed otherwise a success message is displayed and users redirected to the Elicitation Techniques Main Menu.
Figure 31. Select Elicitation Technique Form

Users get to the Select Elicitation Technique Form seen in Figure 31 by clicking the Edit Elicitation Technique link in the Elicitation Technique Main Menu Form. They select the elicitation technique they want to change or select by pressing the Go To Technique button that leads to the next described form.
Users in Edit Elicitation Technique Form can change the details about the selected technique or/and select it or deselect it from the project then they hit the Edit button (see Figure 32). Input is checked, if valid users get a success message and redirected to the Elicitation Technique Main Menu otherwise an error message is displayed.
Users get to the Delete Elicitation Technique Form as shown in Figure 33 by clicking the Delete Elicitation Technique link in the Elicitation Technique Main Menu Form. They select the technique they want to delete by pressing the Delete button. After deleting they are redirected to the Elicitation Techniques Main Menu.
Users get to the Requirements Main Menu Form by clicking the sixth step link in the Nine Steps Form. They have the choice to add, edit or delete their project requirements (see Figure 34).
Users fill the Add Requirement Form, depicted in figure 35, with the name and description and hit the Save button. If the name already exists in the database for the same user a duplication error message is displayed otherwise a success message is displayed and users redirected to the Elicitation Techniques Main Menu. By clicking the Z Tool link, users are redirected to the next described screen.
Users in Z Tool Form as seen in Figure 36 have the directions they need to end with associating their security requirement specification with the corresponding formal methods based language Z files they write. The first step tells users to write the specification in a file extended with .zed. In the second step all what they need is to launch the CZT tool where they open the file created in step one to be parsed and checked for errors. Finally, users can upload the Z files to the database by browsing and then pressing the Upload button.
Users use the Community Z tool in parsing and checking their files that they select as seen in Figure 37. Through this tool they can also export it to XML files. The user can upload the checked files to the database. All features provided are explained in details in the Example presented in Chapter 5.
Users get to the Select Requirement Form by clicking the Edit Requirement link in the Requirement Main Menu Form (see Figure 38). They select the requirement they want to change by pressing the Go To Requirement button that leads to the next described form.
Users in Edit Requirement Form as seen in Figure 39 can change the details about the selected requirement then they hit the Edit button. Input is checked, if valid users get a success message and redirected to the Requirement Main Menu otherwise an error message is displayed.
Users get to the Delete Requirement Form by clicking the Delete Requirement link in the Requirement Main Menu Form. They select the requirement they want to delete by pressing the Delete button. After deleting they are redirected to the Requirement Main Menu (see figure 40).
Users get to the Categories Main Menu Form, depicted in Figure 41, by clicking the seventh step link in the Nine Steps Form. They have the choice to add, edit or delete their own categories that will appear on all projects and select the ones they need for each project requirement. Users are also able to categorize their defined requirement into the categories they define. It is important to note that the system provide four basic categories (Essential, Not Essential, Software Level and System Level).
Users fill the Add Category Form with the name and description and hit the Save button as seen in Figure 42. If the name already exists in the database for the same user a duplication error message is displayed otherwise a success message is displayed and users redirected to the Categories Main Menu.
Users get to the Select Category Form by clicking the Edit Category link in the Categories Main Menu Form seen in figure 43. They select the category they want to change by pressing the Go To Category button that leads to the next described form.
As shown in Figure 44, users in Edit Category Form can change the details about the selected category then they hit the Edit button. Input is checked, if valid users get a success message and redirected to the Category Main Menu otherwise an error message is displayed. The four basic categories added by the system cannot be edited.
Users get to the Delete Category Form by clicking the Delete Category link in the Categories Main Menu Form (see Figure 45). They select the category they want to delete by pressing the Delete button. After deleting they are redirected to the Categories Main Menu. The four basic categories added by the system cannot be deleted.
Users get to the Select Requirement to Categorize Form by clicking the Categorize Requirement link in the Categories Main Menu Form (see Figure 46). They select the requirement they want to categorize by pressing the Edit button beside it.
Users select one of the values for the Basic Category 1 and Basic Category 2 from the drop down menu as we can see in Figure 47. They also check the box of the category they want from the ones they added before. Finally they hit the Save button and get redirected to the Categories Main Menu Form.
Users get to the Prioritization Form by clicking the eighth step link in the Nine Steps Form. They need to fill the Priority number for each of the displayed requirements based on the risk value and the category they have work on in step four and seven (see Figure 48). Entered values is checked after pressing the Save button if they are valid integers a success message is displayed and users redirected to the Nine Steps Form otherwise an error message is displayed.
Users get to the Inspection Techniques Main Menu Form by clicking the ninth step link in the Nine Steps Form. As seen in figure 49, they have the choice to add, edit or delete their own inspection techniques that will appear on all projects and select the ones they need for each project from the edit link.

Figure 49. Inspection Techniques Main Menu Form
Figure 50. Add Inspection Technique Form

Users fill the Add Inspection Technique Form, which we see in Figure 50, with the name and description and hit the Save button. If the name already exists in the database for the same user a duplication error message is displayed otherwise a success message is displayed and users redirected to the Inspection Techniques Main Menu.
Users get to the Select Inspection Technique Form by clicking the Edit Inspection Technique link in the Inspection Technique Main Menu Form. They select the inspection technique they want to change or select by pressing the Go To Technique button that leads to the next described form (see Figure 51).
Users in Edit Inspection Technique Form can change the details about the selected technique or/and select it or deselect it from the project then they hit the Edit button as seen in figure 52. Input is checked, if valid users get success message and redirected to the Inspection Techniques Main Menu otherwise an error message is displayed.
Users get to the Delete Inspection Technique Form by clicking the Delete Inspection Technique link in the Inspection Technique Main Menu Form. They select the technique they want to delete by pressing the Delete button (see Figure 53). After deleting they are redirected to the Inspection Techniques Main Menu.
Figure 54. Online Help

Users can always get the Online help, as shown in Figure 54, that provides directions on how to use the tool at any point through pressing the Help Box at the right bottom corner (see Figure 55).

Figure 55. Help Box
Chapter 4

eSQUARE IMPLEMENTATION

eSQUARE is implemented using the iterative life cycle model. In each iterative, a part of the requirements was selected based on its importance to the project then implemented and tested. This approach helped in monitoring the progress and planning in a simple and reliable way.

There are a number of technologies that can be used in implementing web-based tools. However, using JSP (Java Server Pages) technology in implementing eSQUARE has a number of advantages that as described in the following:

- It is portable; that means the system can be hosted in any operating system and does not have to be a Microsoft Web server. Write once and run anywhere!
- It is easy to write; custom tags and libraries reduce the length of the Java code.
- It is modular; HTML and JSP pages are not affected by the changes happening in the dynamic JSP and JAVA classes. Static content is separated from dynamic content!

Like all technologies, JSP has its disadvantages too. The one that I found during implementing eSQUARE is:
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.

Previously described in Figure 1 in section 3.2, eSQUARE has three layers that have been implemented using the JSP (Java Server Pages) technology. For the presentation layer, it has been implemented using JSP code embedded in standard HTML code. For the business logic layer, it has been written in pure JSP code. However, the data-access layer has been implemented using JAVA classes. MySQL server 5.1 is used for developing and hosting eSQUARE database. eSQUARE is capable of running on standard internet web browsers, although, it is designed primarily around MS Internet Explorer.

The presentation layer represents the user interfaces that users browse and deal with while using eSQUARE. Its forms are the visual presentation of the functionality and service that users can get from this tool. These forms consist of many controls that are connected to JSP commands to invoke a specific method in the JAVA classes. Some of the controls are static and some are dynamic that appear depending on specific conditions. An example of static controls is: the Online Help link that users can always be able to hit when they need it at any time of the process. Another example for dynamic controls can be the Delete button that appears on the Requirements Delete Form. This button appears when at least one requirement is added by the user for the project.
However, a message will inform the user that no requirements has been added for this project and the Delete button will not appear in the form.

The eSQUARE screen flow diagrams model the interactions that users might have with the tool and enable us to gain a high-level overview of the user interface for eSQUARE. The screen flow diagrams are presented by Figures 56, 57, 58 and 59. The boxes represent major user interface forms and the arrows represent the possible flow between them. For example, when users are on the eSQUARE Main Form, they go to the SelectProject form. Once they are there, they can either go back to the eSQUARE Main Form (going back is always assumed) or go to the Nine Steps Form (see Figure 56).
Figure 56. Screen Flow part 1
Figure 57. Screen Flow part 2
Figure 58. Screen Flow part 3
It is important to mention that at any point of the process the user can invoke the online help and can logout too.

*The business logic layer* consists the JSP files that are responsible of receiving the input and commands from users, process them and determine what method to invoke from the related JAVA class. Also, the JSP files are responsible of processing the output retrieved form the JAVA classes in a way that can be presented in a suitable way for the user through the HTML code. For example, AddProject.JSP is responsible of receiving the project name and description from the AddProjectForm.JSP file that belongs to the presentation layer. Then AddProject.JSP invoke the method checkNew( ) from the ProjectBean JAVA class. ProjectBean returns a number that has indication in
AddProject.JSP. In other words, if ProjectBean returns 0 it means the process was done successfully, otherwise numbers from 1 to 4 has different error messages. Finally, AddProject.JSP redirects the user to a specific page with the message gotten from the Java class.

**The data access layer** consists of three packages developed (user, project and general) and the open source CZT package.

CZT package consists of many classes that build the Z tool which users can launch it in the security requirement elicitation step of the SQUARE automated tool. It enables the user to open their already written Z files, parse it, check it and export it to other formats.

Despite of the number of tools available for the Z specification language, most of them do not support the ISO standard for this language [8]. Andrew Martin saw the many tools developed by school and academic projects and found that they were not complete and robust enough [8]. Based on that in 2001, he proposed the idea of the Community Z Tools to be a useful open source that support the ISO specification of Z language. Integrating parts of the CZT in eSQUARE web-based tool is of great advantage that allows using the existing tools to enhance the SQUARE methodology.

In the architecture section we described the three packages that I developed and the role of each class in them. Figure 60 is the class diagram for these three packages and their relations.
Figure 60. eSQUARE Class Diagram
**The database layer** is the layer that holds the data of users and their projects. In eSQUARE it has been built using MySQL server 5.1 and it has many tables that serve the functionality of the tool. eSQUARE databases is accessed by the JAVA classes in the data access layer where data manipulation is performed in terms of SQL queries like insert, update, delete and select.

eSQUARE is a data centric tool where the database represents the cornerstone of the project. For this reason, database has been carefully designed, normalized and refined so that each table represents one and only one object and its columns to fully describe this object which results in better performance and easier development. This ensures that there is no redundant columns in the tables and enhances the scanning of the tables using the indexes. Add to this, the SQL is additive in its nature so we can use the JOIN to take pieces from different tables which makes it flexible to the project needs. It is important to mention that backups of the database are made periodically for emergency.

Another important issue that took place while designing eSQUARE database was the naming of the tables and their columns. Actually, I meant to use names not only help me to identify the purpose of an object, but also to allow all readers and future developers to understand what this object is and what data it stores. I believe that no one should need to spend time reading pages of documentation to understand the names while better naming can takes care of it.

Figure 61 describes eSQUARE database tables, their columns and relations. Also, primary keys are perceived by PK symbol and written in bold, while foreign keys are
written in a bold format only. Relations’ multiplicity is determined on each end of the connecting lines.

Figure 61. eSQUARE ER Diagram
Chapter 5

EXAMPLE

In summer 2004, seven graduate students under the supervision of Dr. Nancy Mead performed an in-depth case study during the development of the SQUARE methodology aiming at documenting the process and getting feedback for refinement purposes [10]. The work in the case study was performed on two phases. In the first phase, the SQUARE methodology was applied on a product called Asset Management System (ASM), a tool helps in planning and allocating critical IT assets in companies, from a Acme company [11]. One of the noticeable issues in that phase that the nine steps of the SQUARE methodology were not applied in the same proposed order. Refinements and recommendations for the methodology were given based on the findings and the analysis of results in the second phase.

In this chapter, the eSQUARE web-based tool is used to prove its efficiency in automating the SQUARE methodology and in enhancing it with the formal methods based language Z in specifying the security requirements. All the data is taken from the case study mentioned above and entered into eSQUARE to show how the tool can be used for managing the nine step process and save the data in the database. In the next pages we will have a closer look on using eSQUARE through the nine steps with a detailed description of 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 as seen in Figure 62.
Terms that has 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 Figure 63 we can see a number of the security term of ASM and they have been selected in the project.

Figure 63. ASM Security Definitions
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 as seen in Figure 64.

Figure 64. ASM Business and Security Goals
Many artifacts like use cases, misuse cases, architectural diagrams and figures are uploaded to eSQUARE each in a file that the user can download it and review it from any computer with internet access and a browser. This is depicted in Figure 65.

Figure 65. ASM Artifacts
In the designed risk assessment methodology in eSQAURE users are given the ability to give weights for the factors used in a risk scale which we described in section 3.4 (see Figure 24). However, if they do not provide weights for the factors, only the risk impact level is displayed near each risk and this is the case in the ASM as explained in Figure 66.
The agreed technique for eliciting the security requirements in ASM was the Interactive approach. This approach name and description are entered into the system and can be viewed and edited as users need. Figure 67 shows the screen where this step is performed.

Figure 67. ASM Selected Elicitation Technique
In the second phase of the SQUARE methodology case study, the security requirements were refined and summarized in nine security requirements [11]. In eSQUARE these requirements are entered into the system as seen in Figure 68.

Figure 68. ASM Elicited Security Requirements
As we mentioned before, eSQUARE provides the ability of creating concise and precise specification for the security requirements by launching the CZT component. To show how to use this feature, a file of the Z specification in latex format is written for each of the nine security requirements of ASM. In Figure 69 is the Z specification for requirement R_07, where its English description is: 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 69. Z Specification For ASM Security Requirement R_07
After launching the CZT component, the written Z specification file can be opened for parsing and type checking. There are 4 options for the language which are standard Z, the Object Z, Circus or Z Rules. Also, there are 4 options for the markup which are Latex, UTF8, UTF16 and XML. Users can un-check the type checking option if they do not want it. In this example Standard Z and Latex markup are selected and the files are both parsed and checked. Figures 70, 71, 72 describes this feature.

Figure 70. Opening a Z File in CZT component
After determining the options users want the component starts reading the file and gives feedback either the specifications are written correctly or they have syntax errors. Figures 73 and 74 show the different feedback depending on the Z file correctness.
Figure 73. CZT Feedback on Incorrect Z Specifications
In Figure 74, we can see that the tool describes the structure of the Z sections that is written in the Z file asm.tex that we presented previously in Figure 69. As the file has been parsed and checked successfully, users are able to export it to different format as shown in Figure 75.
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 (see Figure 76). Uploading the Z files to the database is also a provided feature in eSQASURE.
Figure 76. U16 Export Result
Going back to the step seven of the SQUARE methodology case study, there were eight defined categories that were used in categorizing the security requirements of ASM. These categories are defined in eSQUARE as seen in Figure 77 and ready to be used in categorizing the nine security requirements of ASM as seen in Figure 78.

![Figure 77. ASM Eight Security Requirements Categories](image-url)
Figure 78. Categorizing ASM Security Requirements
eSQUARE uses integer numbers in expressing the priority of projects’ security requirements. However in ASM case study the ninth security requirements were prioritized under three levels (Essential, Conditional and Optional) [11]. As a result, each of the three level is assigned a number and is used in eSQUARE (Essential: 1, Conditional: 2 and optional: 3). Figure 79 depicted this step.

![Figure 79. ASM’s Security Requirements Prioritization](image-url)
Finally, inspection technique selected for ASM project was the Peer Review technique that its description is inserted in eSQUARE as Figure 80 shows and selected for the project as in Figure 81.

Figure 80. Defining the Peer Review Technique in eSQUARE
Figure 81. Selecting Peer Review Technique for ASM Project
Users can get to the Project Summary Form by clicking the last link in the Nine Steps Form. A part of the report summary is seen in figure 82.

Figure 82. Project Summary
Chapter 6

CONCLUSION

Based on my research, I’m convinced even more that there is a growing attention to security requirements engineering in the Software Development Life Cycle and a real need for software tools that support it. In other words, I see that eSQUARE has a promising future for the unique integration of the formal methods based language Z with SQUARE. Actually, eSQUARE can grow to be a robust and a powerful tool that security requirements engineers can rely on to increase their confidence in the security requirements specifications all the way along the automated SQUARE process. As the case study showed, eSQUARE is simple and easy to use where users are capable of following the nine steps of the SQUARE methodology in an effective way.

The eSQUARE web-based tool provides portability to users where they can access their projects and documents from any computer system connected to the internet with a web browser. The tool makes it possible to follow the SQUARE methodology without the need for experienced requirements engineers which reduces the cost that may not be affordable. Indeed, an online help can be invoked at any time to direct users on how to use the tool towards achieving a successful completion of their eSQUARE projects. Not only it saves effort with its simplicity, but it also saves time by arranging all the screens where data needs to be entered and by saving it in the database. In addition, all artifacts and Z files resulting from the process can be uploaded to the database and
downloaded whenever needed. By launching the CZT tool, users get the ability to parse and type-check their Z files. Also, they can choose one of the four supported languages (Standard Z, Object Z, Circus or Z Rules) and one of the markup styles available (XML, UTF8, UTF16 or Latex). Furthermore, users can export their Z files to different styles after passing the parsing step. In this way, security requirements can be concisely and precisely specified and documented.

Table 1 presents a summary of similarities and differences among eSQUARE, mySQAURE and CyLab SQUARE tools. The three tools have multiple similarities like automating the administration part of SQUARE projects, providing users with help option at any time of the work and generating automatic summary for projects. Also, eSQAURE and CyLab SQUARE automate the risk assessment step, which is part of the technical automation of the steps of SQUARE methodology, but mySQUARE does not. In addition, eSQUARE and CyLab SQUARE are both web based tools which give users with portability advantages. At the same time the tools still have differences, for example: eSQUARE provides the support for the formal methods based language Z while mySQUARE and CyLab SQUARE do not. Another difference is that CyLab SQUARE provides the automatic generation of complete reports with cross references of all the materials while the other tools do not. Also, CyLab SQUARE enables multiple users to work on the same project as a team and manage the communication, but this is not available neither in eSQUARE nor in mySQUARE.
Table 1. SQAURE Automation Tools Comparison

<table>
<thead>
<tr>
<th>Feature</th>
<th>Tool</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>
<td>✓</td>
</tr>
<tr>
<td>SQUARE Administrative Automation</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Risk Assessment Automation</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Z language support</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Automatic Project Summary Generation</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>Automatic Reports generation</td>
<td>×</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Documentation</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>User Help</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Team Work Support</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
</tbody>
</table>

Although eSQUARE has showed its useful and effective usage in the example, there are a number of improvements to be made as future work:

- The categorization and prioritization can be enhanced to be more flexible and cover other techniques that adapt with users’ needs in their projects. Exporting and importing projects to and from eSQUARE can be a great feature.
- It will be great to extend eSQUARE so it can handle team-work projects rather than having only one user for each project. Such a feature needs to manage the communication between users of the same projects and take care of parallel access to the same records in the database too.

- Upgrading the tool from JSP application to J2EE application will increase the performance of the system by distributing the system components among EJB servers which breaks the work load and increases response time. This performance improvement can be noticed when the system grows up and many users issue requests at the same time.

- Reports are an important feature that needs to be added to eSQUARE so that users can get information about their projects summarized in one place. Built in reports that users can request directly are needed. It will be of a great advantage if users can create their customized reports too.
REFERENCES


