CHAPTER V

QUANTIFYING SECURITY REQUIREMENTS USING APPRECIATIVE INQUIRY AND FUZZY THEORY

5.1 Introduction

As presented in the previous chapter (SAIT) the integration done according to the factors related to the requirement of elicitation and security requirements, is not complete without the process of quantifying security requirements. Both of SQUARE and CLASP use the external method to achieve security requirements quantification i.e. (perform risk assessment, categorize and prioritize security requirements). Thus, this study adapts itself to an algorithm of the fuzzy soft set theory to quantify security requirements in the requirement phase. This algorithm will be integrated with SAIT especially in the design phase, where, in this phase SAIT uses external methods to complete the process of eliciting and quantifying security requirements in the SDLC artifact. Moreover, the results of the life cycle in the proposed SAIT must change i.e. destiny phase will contain extra steps and outputs. The aims of this integration is to elicit user and security requirements and quantify security requirements in a proposed technique namely Secure Appreciative Inquiry Fuzzy Quantification Technique (SAIFQT) to finally obtain at the end of the day, the Security Requirements Index (SRI). This enhancement of SAIT technique was proven to be successful based on the case study that quantified security requirements using fuzzy soft set algorithm as a preliminary study (See Appendix B).

5.2 Benefits of Integrating Algorithm of Fuzzy Soft Set Theory into SAIT

There is no reliable concrete way to quantify security requirements of an SDLC artifact (Khan, 2008; Khan, 2011; Savola et al., 2012). This quantification is necessary to know about the security state of an SDLC artifact before (at the early stage especially in the requirement phase) and after the software development process. Moreover, this could help software developers and security experts in allocating further resources to increase the security and decrease the vulnerabilities in any software. The quantified information about vulnerabilities and errors is important because if an error leading to vulnerability is not corrected at an early phase, the cost of correcting it might increase tenfold with every additional development phase (Khan, 2008). In this chapter, the fuzzy soft set theory will be used to find the SRI by integrating the algorithm of the fuzzy soft set theory into SAIT. As discussed earlier, the option of using fuzzy soft set theory to address a problem in a particular field has its own obstacles, because it is a very appropriate and accurate decision making tool.

5.3 Embedding Fuzzy Soft Set Algorithm into SAIT: Design Phase

The embedding of fuzzy soft set algorithm into SAIT especially in design phase, related to this reason; one of the design phase steps in this technique use external method for perform risk assessment, categorize, prioritize security requirements, and document security relevant requirements processes which will be achieved by embedding the fuzzy soft set algorithm as follow.

a) **Users and the developers examine the feasible size of the system.**
   Based on the recognized strengths highlighted by the stakeholder.

b) **Develop artifacts to support security requirements definition.**
   The following relics are to be gathered: system architecture diagram, use case scenarios/diagrams, misuse case scenarios/diagrams, attack trees, and standardized templates and forms. Consequently the dimension and the strength of the system can now be derived.

c) Consequently the dimension and the strength of the system can now be derived (Draw Use Case Diagrams. Identify user roles and resource capabilities and Misuse Case Diagrams; apply security principles to design (detail misuse cases, identify attack surface, and annotate class designs with security properties)); (Perform security analysis of system requirements and design (threat modeling)).

d) **Elicit software and security requirements.**

An essential factor in this phase is to make sure that the software/business requirements are proven, and that, they do not have, implementation or design restrictions, rather than requirements; thus elicit all security requirements that are related to software requirements.

e) **Perform quantification of security requirements by using fuzzy soft set theory as a tool:**

To quantification security requirements, categorize security requirements, prioritize security requirements, and perform risk assessment to the whole system.

5.4 Proposed Quantifying Security Requirements Using Fuzzy Soft Set Theory

In this chapter, fuzzy soft set theory will be used to, first, quantify Vulnerabilities Index (VI) then quantify Errors Index (EI) and finally quantify Security Index (SI) for each single security requirements and for the whole software or system by using the VI and EI.

TABLE 16: Vulnerabilities, Errors and Security index

	Vulnerabilities Index	Errors Index	Security Index
Definition	Is an index for weakness in a product that could allow an attacker to compromise the integrity, availability, or confidentiality of that product	Is an index for undesirable deviation from requirements or to change the functionality of the program	Is an index for the state of feeling safe, stable, and free from vulnerabilities resulting from errors
Example	Denial of Service problems that allow an attacker to cause a Blue Screen of Death	Null pointer references or failed lock releases, rather than redundancy checking	Authorization

Source: SANS, 2014

5.4.1. Quantify Vulnerabilities Index

Every software has vulnerabilities, which are the gates exploited by attackers to control the system as a whole or to destroy it. In this subsection a discussion in quantifying VI that may happen in the system; is presented according to Khan (2008) where the VI depends on:

a) Percentage of vulnerability occurrences in each software; i.e. every vulnerability may happen one time or more than one time in a particular software (the likelihood of each vulnerability being exploited).

b) Percentage of the damage that can happen because of this vulnerability to the software.

c) Percentage of the asset important in particular software. Every software has its own assets and each asset has different value of importance, each single asset may be affected by at least one of the system vulnerabilities.

5.4.2. Cagman , Citak and Enginoglu Algorithm (CCEA)

The definition of the *fuzzy* decision set and the algorithm of Cagman, Citak and Enginoglu (CCEA) as given in their paper (Cagman et al., 2011).

Definition: Let $F_X \in FPS\ (U)$. Then a fuzzy decision set of F_X, denoted by $F^d{}_X$, is defined by equation 1 as follows.

$$F^d{}_X = \{\mu_{F^d{}_X}(u)/u : u \in U\} \quad \ldots\ldots(1)$$

Where a fuzzy set over U, its membership function $\mu_{F^d{}_X}$ is defined by equations 2 and 3.

$$\mu F^d{}_X : u \to [0,1] \quad \ldots\ldots(2)$$

$$\mu F^d{}_X = \frac{1}{|\mathrm{supp}(X)|} \sum_{x \in \mathrm{supp}(X)} \mu_X(x) \chi_{f(x)}(u) \quad \ldots\ldots(3)$$

Where;

FPS: is Fuzzy Proposition Set.

$F^d{}_x$: is a function set for fuzzy decision set of U.

$\mu F^d{}_x$: decision of $F^d{}_x$ (membership function).

Where *supp* (X) is the support set of X, $f_X(x)$ is the crisp subset determined by the parameter x and characteristic function as in equation 4.

$$\chi f_X(x)^{(u)} = \begin{cases} 1, & if\ u \in f_X(x), \\ 0, & if\ u \notin f_X(x), \end{cases} \quad \ldots\ldots(4)$$

The explanation of how the algorithm will work according to the algorithm steps is given below:

Step1: Construct an intended universe (U) which contains a set of elements to make a process of decision making related to them. Construct a power of universe

(Eu); which is a set of parameters that describes all aspects of the elements in (U). Finally give all parameters fuzzy values (E).

Step2: Compute fuzzy set over (U) which is characteristic function (see (4)) that is computed by step3.

Step3: Find $F(e_{u,n})$ for all parameters, i.e. Construct a set for each parameter which contains the elements which have this aspect (parameter) $(e_{u,1})$, $(e_{u,2})$,….., $(e_{u,n})$. Compute the fuzzy decision using this mathematical algorithm that is related to (Cagman et al. 2011) see equation (3) which is the main equation.

Once a *Fuzzy* decision set of a *Fuzzy soft set* has arrived, it may be necessary to choose the best single alternative from the alternatives. Therefore, a decision can be made by the following algorithm.

Step1 : Construct a *soft set* F_X over U.

Step2 : Compute the *Fuzzy* decision set F^d_X.

Step3 : Select the largest membership grade $\max \mu_{F^d_X}$.

When the algorithm is applied, if this element is available in $F(e_{u,n})$ then it will be multiplied by One, otherwise, by Zero as it is demonstrated in the characteristic function (see (4)) mentioned in the definition. After applying the algorithm for all elements, a fuzzy value is gained for all elements; this value describes the element situation according to the parameters used in that case, i.e. if fuzzy value near to One this means that the element that has this value has a high priority, if near to Zero that means it has a low priority.

5.4.3. Quantify Error Index

Every single error or more can lead to one vulnerability or more, which are used in a direct way by attackers to control the system, as to read or modify or destroy it. In this section a discussion on quantifying EI that may be available in the system using the algorithm given by Cagman et al., (2011), is presented according to Khan (2008) where the EI depends on:

a) Percentage of error occurrences in each software; i.e. every error may happen one time or more than one time in particular software (likelihood of each single error being exploited).

b) Percentage of the error effects; i.e. every single error may contain one vulnerability or more, for instance, the percentage of effects of error X which leads to three different vulnerabilities more than the percentage of effects of error Y that leads to one vulnerability.

c) Percentage of the error dangers in particular software. Every software has its vulnerabilities, every single vulnerability is influenced by one or more errors, and some of these errors affect the different levels of vulnerabilities' danger.

5.4.4. Propose Quantification for Security Index

After quantifying the VI and EI related to the proposed software, the SI equation is proposed, then finally, the SI contents are connected with the security requirements in the proposed software. There are relationships between errors and vulnerabilities; these relationships give us the SI as a result. Each security requirement is affected by one vulnerability or more. Each vulnerability is affected by one error or more. On the other hand, the SI value should be a fuzzy value i.e. the value should be between (0,1). According to the previous context, the suitable equation to quantify the SI is proposed as in equation 5.

$$SI^{Vi} = \sum_{i,j \in I} \frac{V_i \; E_j}{j} \qquad \ldots\ldots(5)$$

Where;

SI^{Vi} : is security index for vulnerability i .

a) Categorize security requirements. The elicited specifications are classified (not less than), in accordance with the following conditions: important,

unimportant, system-level, software-level architectural constraint restrictions. Considering this, the latter is not regarded as requirements, but their presence signifies that the previous steps should be implemented again.

b) Perform risk assessment, whereby in this step, the weaknesses and risks relevant to the system are determined, in addition to the chances that the risks will result in attacks. The authors have suggested implementing current risk evaluation techniques.

c) Prioritize security requirements. It is presumed that, only a few requirements can be applied; consequently, the most essential requirements have to be determined.

d) The role of organizations, their companies and procedures are viewed in the external point of view.

e) As a matter of fact, it is not possible to analyze security in an application, therefore application testing and evaluations should be a core element of the entire security technique. Specifically, automated assessments tests can discover security problems that are not recognized during code or implementation reviews, discover security threats unveiled by the operational environment, and act as a protection mechanism, by finding downfalls in the design, requirements, or execution. Typically, test and assessment functions are held by a test analyst, or by the quality assurance organization, but it can be extended to the complete life cycle. The factors and elements that should be considered are:

i. Verify the security attributes of resources.
ii. Perform a source-level security review.
iii. Identify, implement, and perform security tests.
iv. Document security-relevant requirements.

v. Integrate the security analysis into the source management process: A vital objective of software security is to generate and sustain multiple-use source code, which fortifies the fundamental security services in the software and over an organization's applications. This objective is most effectively accomplished, by applying secure development practices into an organization's general development process as soon as possible in the SDLC.

vi. Manage security issue disclosure process, and address reported security issues: It is specifically significant in the perspective of program up-dates and improvements, to determine which actions will be employed to recognize, evaluate, focus on, and resolve weaknesses. Establishing resolving techniques will accelerate response, and reduce risks, by interpreting tasks, obligations, and procedures to adhere, after recognizing the weaknesses. Removal techniques are often fed by application tests, both in in-house or third party, and help to manage information when disclosure happens.

vii. Define and monitor security metrics: A project development team will not be able to deal with what it cannot be determined. However, applying efficient analytics monitoring attempt can be a challenging aspect. In spite of this, metrics are important factors of a general software security attempt. They are essential in determining the present security stance of an organization, and in concentrating on the most crucial weaknesses, and exposing how effectively or improperly the investments are in the improved security of an organization.

viii. Realizing how applications will be employed, and how they could possibly be misused or infected.

ix. Potential preventive controls and their value and efficiency.

Output of Phase 4:

The outputs of this phase are: the 'what should be' needed artifacts: scenarios, misuse cases diagrams, models, templates, threat modeling diagrams. Other outputs include reported security issues, forms, SI, results of risk assessment, initial cut at security and software requirements; categorized and prioritized security requirements.

Table 17 summarizes the proposed integrated technique namely Secure Appreciative Inquiry Fuzzy Quantification Technique (SAIFQT) phases and main steps inside each phase, viewing the security issues concerned.

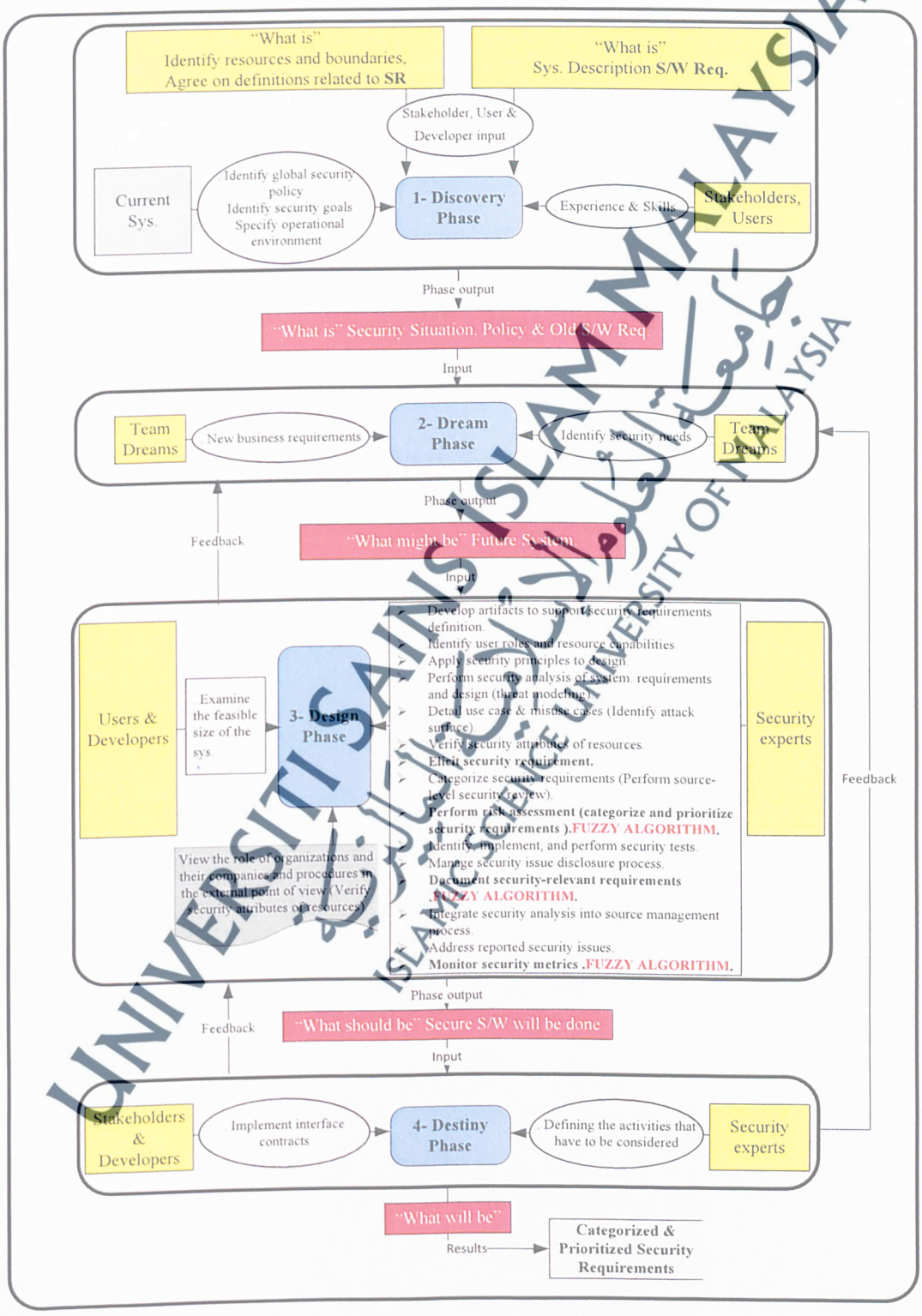
TABLE 17: Summarizing phases of proposed Secure Appreciative Inquiry Fuzzy Quantification Technique (SAIFQT).

Phase name	Executers	Security issue	Expected results
Discovery	• Stakeholder • User • Developers	• Discover security situation • Security definitions agreed • Resources and trust boundaries • Identify the global security policy	• 'What is the Security situation in the old system. • 'What is' the old software/business requirements needed in future system • 'What is' the security definitions agreed • 'What is' Resources and trust boundaries • 'What is' The global security policy • 'What is' The existing situation of the old system (Software Requirements)
Dream	• Stakeholder • User • Developers • Security experts	• Security definitions agreed related to business/software requirements	• 'What might be' the System boundaries, assets, resources besides the agreed security definitions, related to business/software requirements
Design	• User • Developers • Security experts	• Misuse cases • **Make risk assessment** • Threat modeling • **Fuzzy quantification, and reported security issues** • Categorized security requirements • Prioritized security requirements	• 'What should be the' Needed artifacts: scenarios, misuse cases, models, templates, forms, and threat modeling diagrams. • Risk assessment results. Initial cut at security and software requirements • Reported security issues • **Categorized security requirements using fuzzy** • **Prioritized security requirements**

			using fuzzy
Destiny	• Stakeholder • Security experts • Developers	• Initial selected security requirement • **Categorized security requirements** • **Prioritized security requirements** • **Security requirements index**	• 'What will be the' Initial selected requirements, documentation of decision-making process and rationale

The steps in the proposed SAIFQT in the table 17 are illustrated in figure 18, and there are four phases to complete the eliciting user and security requirements process through SAIFQT. The steps are explained in the following sub-sections:

FIGURE 18: Proposed Secure Appreciative Inquiry Fuzzy Quantification Technique (SAIFQT) Process.



5.5 Summary

In this chapter, the embedding of the fuzzy soft set algorithm with Secure Appreciative Inquiry Technique (SAIT) to gain Secure Appreciative Inquiry Fuzzy Quantification Technique (SAIFQT) has been emphasized. SAIFQT is a technique to elicit each of the software and security requirements and quantify security requirements which are related to the proposed software through the SDLC artifact. Moreover, it is used to make the decisions for all elicited security requirements to prioritize and categorize them in the design phase of (SAIFQT), the issue of eliciting and quantifying security requirements will be discussed in the next chapter through a real case study.