UNIVERSITY OF KWAZULU-NATAL

SECURE REQUIREMENTS ENGINEERING IN A CONSTRAINED AGILE ENVIRONMENT

By

N.K. Naicker

214585809

A thesis submitted in fulfilment of the requirement for the degree of Doctor of Philosophy (PhD)

School of Management, Information Technology and Governance

Faculty of Management Studies

Supervisor: Prof. M. S. Maharaj

2017

DECLARATION

I, *Nalindren K. Naicker*, declare that:

(i) The research reported in this thesis, except where otherwise indicated, is my original research.

(ii) This thesis has not been submitted for any degree or examination at any other university.

(iii) This thesis does not contain other persons' data, pictures, graphs or other information, unless specifically acknowledged as being sourced from other persons.

(iv) This thesis does not contain other persons' writing, unless specifically acknowledged as being sourced from other researchers. Where other written sources have been quoted, then:

- Their words have been re-written but the general information attributed to them has been referenced.
- Where their exact words have been used, their writing has been placed inside quotation marks, and referenced.

(v) This thesis does not contain text, graphics or tables copied and pasted from the Internet, unless specifically acknowledged, and the source being detailed in the thesis and in the Reference Section of this thesis.

Signature: *N.K. Naicker*

Date: 15 November 2017

ACKNOWLEDGEMENTS

I wish to express my sincere appreciation to all those who contributed to this study. In particular, I would like to thank the following persons:

- Firstly to God, for granting me the strength, good health and courage to embark on such a significant milestone in my life.

- Professor Manoj S. Maharaj, my supervisor, for his professional guidance, mentoring and assistance in this research study.

- To my wife, Gayshree Naicker and daughter, Thishalia Naicker for their patience, understanding and encouragement.

- Senior Management of the various software development companies who assisted me with the data collection.

- Finally to the Durban University of Technology, Research Directorate for giving me their support for this research study, especially Prof Moyo and Ms Vaneshree Govender.

ABSTRACT

Requirements Engineering (RE) is a software engineering process that takes place early in the software development life cycle namely, during the planning phase of software development. A list of highly refined requirements that is the blueprint for the system, is the output of this process. It is vital to address critical issues such as security within RE, to prevent patching and hot fixing later. Exorbitant losses can be prevented through secure systems development. The purpose of this research study was to delineate the Agile RE practices through a sequential explanatory mixed methods study approach to explicate the relationship between RE practices and the security of an application. An in-depth literature review was undertaken to understand RE processes and security approaches during application development. This mixed methods research study was contextualised at seventeen software development companies in South Africa. Data was collected in three phases. In the first phase, the researcher used a field survey questionnaire as the primary research instrument to gather data on Agile RE practices such as elicitation, security approaches and requirements prioritisation. In phase two of the data collection, interviews were used as a qualitative data gathering tool to explain, triangulate and strengthen the survey results. The security of live Agile Software Development artifacts were then randomly evaluated using a dynamic analysis security testing (DAST) tool. To contribute to the body of knowledge, the researcher used fuzzy logics and fuzzy sets to develop an automated fuzzy tool that assists requirements engineers to control client requirements. The Design Science Research Methodology, an Information Systems (IS) theoretical framework, guided the development of the automated fuzzy software tool. The automated fuzzy tool was evaluated in phase three of data collection and showed positive results for ranking client requirements in Agile RE. The major finding of this study was that although Agile RE practices in the real world are aligned to mainstream RE, proper security approaches are lacking. The problem is exacerbated by the lack of web application security knowledge and insufficient application security training by requirements engineers. The study concludes that poor security practices in Agile RE are having a negative impact on the security of the Agile Software Development product. As an implication of this study, the researcher suggests stricter adherences by practitioners to Agile Software Development principles and values as outlined in the Agile Manifesto and Agile Security Manifesto.

Key words: Agile, Security, requirements engineering, requirements prioritisation, fuzzy TOPSIS, Dynamic Analysis Security Testing (DAST).

Table of Contents

Chapter Seven: Discussion & Interpretation of Findings

Chapter Eight: Summary, Conclusions and Implications of Study

LIST OF ANNEXURES

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

A	Alternative
ANOVA	Analysis of Variance
AHP	Analytical Hierarchical Process
ASD	Agile Software Development
AT	Activity Theory
BNP	Best Non fuzzy Performance
BRD	Business Requirements Document
C	Criteria
CC	Closeness Coefficient
CERT	Computer Emergency Response Teams
CORE	Controlled Requirements Expression
COTS	Commercial off-the-shelf solutions
CSRF	Cross Site Request Forgery
CWE	Common Weakness Enumeration
DAST	Dynamic Analysis Security Testing
DM	Decision Maker
DSDM	Dynamic System Development Method
DSRM	Design Science Research Methodology
FDD	Feature Driven Development
FPIS	Fuzzy Positive Ideal Solution
FNIS	Fuzzy Negative Ideal Solution
ICT	Information and Communications Technology
IS	Information Systems
ISO	International Organisation for Standardisation
JAD	Joint Application Development
MCDM	Multi-Criteria Decision Making
MMR	Mixed Methods Research
MOSRE	Model Oriented Security Requirements Engineering
OOP	Object Oriented Paradigm
OWASP	Open Web Application Security Project
QFD	Quality Function Deployment
UML	Unified Modeling Language

R	Requirement
RE	Requirements Engineering
RO	Research Objective
RSQ	Research Sub Question
SAM	Soft Activity Model
SANS	System Admin, Audit, Network and Security
SQL	Structured Query Language
SRE	Security Requirements Engineering
SSM	Soft Systems Methodology
SPSS	Statistical Package for Social Science
TAM	Technology Acceptance Model
TFN	Triangular Fuzzy Number
TOPSIS	Technique for Order of Preference by Similarity to Ideal Solution
XP	Extreme Programming
XSS	Cross Site Scripting

LIST OF TERMINOLOGY

Ad hoc: Flexible or flexibility in approach.

Agile RE: Specific practices for Requirements Engineering (RE) in Agile methodologies (Kassab 2014).

Automated fuzzy tool: A tool constructed for use in Agile RE to control the 'just in time' requirements prioritisation process.

Process Model: Represents the flow of activities for secure Agile RE.

Product Model: A blueprint that emphasizes plans and intentions for structuring security in Agile RE practices.

Requirements: Capabilities or features of a desired system (Wurfel et al. 2016).

Requirements Engineering: A software development process that occurs in the early stages of software development, namely during the gathering of user requirements.

SAM: Soft Activity Model is a conceptual framework constructed for use in this research study. SAM was constructed from two theoretical frameworks, namely Soft Systems Methodology and Activity Theory.

Secure Agile RE practices: Secure RE activities incorporated with normal RE practices in Agile Software Development.

Sprint: Terminology used in Agile Scrum to denote software development iteration.

Threats: What the attacker can do to violate the security concerns of the system (El-Hadary & El-Kassas 2014).

User Story: Represents a feature customers want in the Agile Software Development Project (Kassab 2014).

Vulnerability: Weakness in the system that may be exploited by an attacker (El-Hadary & El-Kassas 2014).

CHAPTER ONE: INTRODUCTION AND BACKGROUND TO THE THESIS

1.1 Introduction

Traditionally application security was not an issue as applications were built for use on stand-alone computers. In order to meet the demands of the customer in the last two decades, there has been a proliferation of interconnected services via the World Wide Web. We are now living in an era where online web applications are connected by the Internet for the convenience of the end user. Interconnectivity has brought about numerous benefits, namely, customers are now linked to suppliers, purchases and payments are made online, reports can be viewed remotely, there is centralization of data and personal information can be better managed. In order to maintain this status quo, software development has evolved to a point that new methodologies that promote rapid development such as Agile Software Development (ASD) were introduced (Pressman & Maxim 2015).

However, despite the introduction of these new methodologies, the spate of hacking incidents constantly reported in the media suggests that software development has yet to progress to the point where web application security can be considered conventional practice. Eighty percent of all applications have reported vulnerabilities (Marashdih & Zaaba 2016). Recently Ola, India's largest taxi operator uncovered two security vulnerabilities in their software system. Hackers with basic programming skills were able to enjoy unlimited free rides by firstly charging the ride to another person's account or charging Ola for the ride (Computer World UK 2016). This example outlines a lack of authorization, authentication and violation of the online web application. This is a case in point to show that failure to protect system assets from unauthorized users can result in huge losses for the customer and create poor customer satisfaction and low confidence in the software product.

Security threats to a web application such as "theft, vandalism, unauthorized disclosure, destruction, fraud, extortion and espionage" can result from failure to protect the web applications assets through poor software development (Souag et al. 2015). A more robust approach to web application security during software development can prevent these vulnerabilities. An example is used to illustrate how weak security processes in software development can create problems.

Suppose there is no validation on the input of the web application, then hackers can gain unauthorized access to the system by stealing user accounts. A hacker can steal user session tokens through a cross-site scripting attack. In cross site scripting, a malicious script is inserted at the point where the application accepts user inputs. The thieve code can then transfer the user's private information to the attacker (Marashdih & Zaaba 2016). This problem could be averted during software development by ensuring that all input is validated.

A new light weight methodology sparked by the drive for more online web applications warrant that security issues are considered more seriously. The introduction of Agile Software Development was considered the Holy Grail for software developers, but clearly defective processes within software development are still hampering the ability of developers to produce secure software. Therefore, it is important to assess in the real world, if the traditional approach to security, namely, no or limited security, is still the de facto standard for Agile Software Development (Tondel et al. 2008).

Salini and Kanmani (2011) advised that security concerns must be taken into account in the early phases of software development. In software engineering taxonomy, the process is termed Requirements Engineering (RE) and starts during planning and continues through to modeling (Kassab 2014). Identifying and resolving potential problems early in the development life cycle will prevent software failure and the need for rework (Maheshwari & Sharma 2014). In Agile Software Development projects, requirements engineering is considered a critical success factor (Salini & Kanmani 2011; Sheffield & Lemetayer 2013; Fontana et al. 2014). Agile RE is ostensibly different from traditional RE. Cao and Ramesh (2010), in a qualitative study, found the following Agile RE practices dominant in industry namely, "face to face communication over written specifications, iterative requirements engineering, requirements prioritisation, managing requirements change, Prototyping, test-driven development, review meetings and lastly, acceptance tests". In Agile RE, these are considered as 'just-in-time' activities with no clear separation between activities (Schön et al. 2017).

Agile RE takes place iteratively and incrementally within a rapidly changing environment and therefore requirements cannot be frozen (AL-Ta'ani & Razali 2013). Change in requirements are

created as a result of stakeholders changing their preferences, introducing new technology and the pressures around getting the product into the market before competitors (Cao & Ramesh 2010). Considering the nature of Agile Software Development, there is a need to investigate Agile RE practices to uncover the security approach utilised and its impact on application security.

A significant Agile Software Development activity that can impact the security of the system is the prioritisation of requirements. The prioritisation process occurs within the analysis activity and is a means by which requirements engineers control requirements (De Lucia & Qusef 2010). The prioritisation process is instrumental in ensuring which requirements get selected for implementation. In security terms, the process either allows a security requirement to get implemented or ensures that it is kept on hold indefinitely. It is therefore important to conduct an in-depth focus on how requirements engineers control client requirements in industry by means of requirements prioritisation.

Researchers agree that empirical studies on how companies are conducting RE in Agile Software Development are still in its infant stage and much more research is required in this critical area (Cao & Ramesh 2010; AL-Ta'ani & Razali 2013; Inayat et al. 2015). Furthermore, integrating security into requirements engineering processes also poses a research challenge (El-Hadary & El-Kassas 2014). In order to close the research gap there exists a need to conduct an in-depth analysis of Agile RE practices in industry, especially in those processes that are important to security, to assess its impact on secure software development.

1.2 Statement of the problem

Major security breaches in applications have resulted in severe losses to customers all over the world (Elahi et al. 2011). In order for the problem to be addressed, experts have advised that security issues should not be treated as an afterthought but addressed within the software development process (Souag 2012; El-Hadary & El-Kassas 2014; Souag et al. 2015). Greater care is now being placed on building secure systems with a higher emphasis on security requirements engineering in the software development process (Salini & Kanmani 2011). Security Requirements Engineering has now emerged as a significant requirements engineering process. As such, there has been a research fixation for the past two decades on developing various methodologies and

frameworks packaged as security approaches for implementing security requirements engineering in support of secure software development. The primary task of security requirements engineering is to identify and document requirements needed for secure software development through uitilisation of a security approach (El-Hadary & El-Kassas 2014).

Agile Software Development, considered to be a light weight software development process model, is not without challenges. Agile RE offers flexibility in RE, therefore Agile RE practices may vary from developer to developer. Further, Agile RE is an iterative process and therefore far more dynamic and volatile than traditional RE (Cao & Ramesh 2010). In order to satisfy the needs of a demanding customer, Agile Software Development focuses on quick releases and rapid software development. The short iteration times leaves very little time to consider non-functional requirements such as security. Hence the nature of Agile RE can prevent the inclusion of security requirements engineering. The situation is exacerbated as RE processes in real world practice take place within a constraining environment.

There are several constraining factors to consider in Agile RE. Firstly, a combination of project factors such as schedule, risk, cost and human resources are constraints to RE processes. The nature of the requirements is another constraining factor as requirements have complexity, dependencies, importance, business value and volatility. Lastly, customer expectations are a constraint in the dynamic Agile Software Development environment (AL-Ta'ani & Razali 2013). Due to these constraints, Agile RE is much more complex. Adding security into the mix is seen as another constraint, as it detracts developers from the features of the system that are hotly pursued by the customer.

In Agile Software Development, for systems to comply with a basic level of security, it would be reasonable to expect that requirements engineers adopt one of the many security approaches published in security requirements engineering literature. However, addressing the challenges and constraints within a fast paced Agile RE environment leaves little room for eliciting secure requirements through elaborate security approaches. Researchers agree that approaches in the literature on security requirements engineering are far too complex for the regular software development company to implement in Agile Software Development (Tondel et al. 2008).

Therefore, it is important to assess how the regular software developer addresses security within Agile RE. It follows, owing to the nature of the Agile Software Development environment, a research gap exists on whether security requirements are elicited during RE (refer to Annexure M).

Practitioners have found, in instances where security requirements are elicited, in a trade-off between functional and non-functional requirements, preference is given to the much desired functional requirements requested by the customer at the expense of critically important but constraining security features (Salini & Kanmani 2011). The requirements prioritisation process in trying to respond to the constraining Agile Software Development environment, ensures that security requirements do not get selected for implementation. Eventually these security requirements get dropped. A research gap on how Agile RE practices such as requirements prioritisation occur in industry, and how this process impacts on security requirements engineering must be addressed.

Requirements prioritisation makes use of a prioritsation technique to control which requirements get implemented, which requirements remain on hold, and which requirements get dropped. Literature has highlighted a shortage of automated tools to support secure RE processes (Souag et al. 2015). In this regard, a research gap was identified for an automated requirements prioritisation tool to control requirements. A desktop literature review into the various requirements prioritisation techniques revealed that a gap existed for the utilization of fuzzy TOPSIS as a prioritisation technique. Therefore, the development of an automated tool using this novel mathematical approach to support secure RE activities in Agile Software Development will contribute to the existing body of knowledge and fill a research gap.

Amid all the problems experienced in Agile RE, practitioners, at best, resort to a security needs identification at a later stage in the software development life cycle, namely, implementation or maintenance and support. This means that security requirements must now be fitted into a pre-existing design. This ad hoc approach leads to serious design challenges and rework that can be costly (El-Hadary & El-Kassas 2014). The research gap on an appropriate security approach in Agile RE must be addressed (refer to Annexure M). In the midst of escalating security concerns,

the adoption of a security approach that could be implemented in Agile RE is currently more viable than ever before.

There exists a need to conduct a study in industry to establish if there is a relationship between Agile RE practices and the security of Agile Software Development products. Implementing security at requirements engineering phase is the crowning achievement for secure software development. In order to explicate the relationship between requirements engineering and security, the study will attempt to answer the following research question: ***How do Agile RE practices impact the security of Agile Software Development products?***

A literature search revealed that no research studies have been conducted internationally or locally relating to the above research question. Ample scope thus exists for current research into understanding the impact of Agile RE practices on the security of an application (non-functional requirement). More especially the Agile RE practices of requirements prioritisation and security requirements, as the impact of these two RE activities directly influence the security of the product. This research study will initially investigate RE processes used in Agile Software Development projects by first conducting an extensive literature survey on mainstream RE processes, Agile RE practices and secure RE approaches. The study will be contextualized to South African software engineering companies, assessing the real world practices of Agile RE and gauging its effect on the security of the software products. The study will also highlight the challenges faced by Agile Software Development teams in the area of requirements engineering. The construction of an automated tool to support secure requirements engineering and close the research gap, is an output of the study.

1.3 Objective and research questions

The research objectives (ROs) of the study are to:

(RO1): Assess mainstream RE processes and secure RE approaches and Agile RE practices from the existing body of knowledge.

(RO2): Evaluate the extent to which secure RE processes are implemented in Agile RE practices in industry.

(RO3): Apply the Fuzzy Technique for Order of Preference by Similarity to Ideal Solution (Fuzzy TOPSIS) as an alternate method to rank clients requirements.

(RO4): Evaluate the automated fuzzy ranking tool to support secure Agile RE practices.

(RO5): Evaluate application security using a dynamic analysis security testing (DAST) tool.

(RO6): Develop evidence based guidelines of implementing security in Agile RE that will be convenient for a regular software developer to adopt.

To achieve the above objectives, the following Research Sub-Questions (RSQs) will be answered in this study:

RSQ1: What are Agile RE practices in the software development industry?

RSQ2: How do software engineers manage requirements prioritisation in Agile RE?

RSQ3: To what extent are secure RE processes implemented in Agile RE practices in industry?

1.4 Significance of the study

This study will benefit software development organisations in the following ways:

- Gain more insight of Requirements Engineering activities in Agile Software Development that will guide managers and decision makers on RE process improvement strategies.
- Uncover challenges that teams face when implementing Agile RE in the context of security.
- Develop an automated fuzzy tool to help requirements engineers control clients' requirements in Agile RE.
- Develop evidence based guidelines for secure RE within a constrained Agile Software Development environment.
- Show how the use of security metrics tools can improve application security.

1.5 Scope of study

This research study will only consider software development companies implementing Agile Software Development methodology to construct software applications. Agile RE practices are examined from the overall perspective of Agile Software Development methods and not in the context of any particular methodology such as Scrum, Extreme Programming and Feature Driven Development. The study was based on web application development projects. Other types of

projects such as mobile application projects were not considered. Further, the research study steered away from requirement engineering cost estimates which is identified as another rich research area in the field of requirements engineering.

The study is contextualized in the South African software engineering industry. The study excludes industries that develop commercial security software tools and companies that, by their very nature, must have security intensive products such as banks. Although the study focused on software development teams, global software development teams and distributed teams were excluded from the study.

1.6 The Research blueprint

A. The Research Study

Figure 1.1 illustrates the process model for the research study:

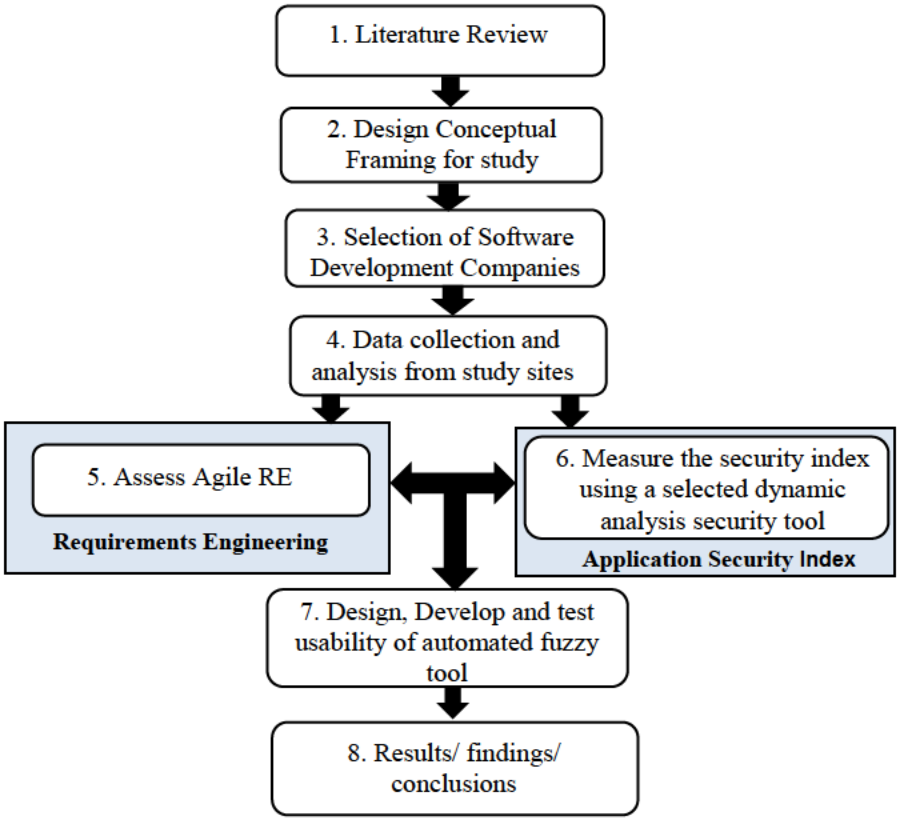


Figure 1.1: Process Model for the research study

Source: Researcher's own construction

1.7 Structure of the thesis

This study is presented in eight chapters, which are arranged in the following manner:

Chapter One: Introduction and background to the thesis

Provides a general background and orientation to the study. The rationale for the study, research problem statement, objectives of this study, key research sub-questions, significance of study as well as the scope of the study, are presented in this chapter.

Chapter Two: Literature review

This chapter comprises the literature review as addressed by the aim and the key research sub-questions. The chapter reviews research studies from literature that are associated with Agile RE practices and explore the link between Agile RE and secure software development. The chapter also analyses and discusses an important RE critical success factor that is responsible for controlling requirements, namely requirements reprioritisation. Further, a mathematical algorithm used in this research project as a basis for the development of the software tool is presented in this chapter.

Chapter Three: Theoretical framework and research design guiding the study

This chapter comprises two sections. Section A provides the theoretical framework for the study and discusses the theoretical models namely, Activity Theory, Soft Systems Methodology (SSM), Design Science Research Model (DSRM) and Technology Acceptance Model (TAM) that are chosen to guide this research study. The chapter breaks down the theoretical models into the various stages, discussing their terminology and showing how they are relevant to this research study. An emergent conceptual model is also discussed in Section A of this chapter.

Section B of this chapter focuses on the research methodology adopted in this study. More specifically, this section outlines the research design and methodological paradigm (sampling procedures and methods of data collection). A description of the research methods and instruments used are outlined. Primary and secondary data collection undertaken at each study site is discussed. This chapter also deals with data analysis as well as the reliability and validity of the research

methods used. Ethical issues considered during data gathering are also discussed in this section of the chapter.

Chapter Four: Presentation of the Automated Fuzzy Tool

The design and development of the automated fuzzy tool is discussed in terms of the Design Science Research Methodology. The fuzzy TOPSIS algorithm is used as a basis for the tool. The tool has several applications in requirements engineering and these are discussed. The main application of the tool is to rank user requirements in Agile RE. The chapter concludes with an evaluation of the tool.

Chapter Five: Presentation of survey results and findings

Presents the survey results and analyses and interprets the data, collated by the researcher, in response to the critical questions form the basis of this chapter. Data from the questionnaires are analysed and summarised graphically using statistical analysis.

Chapter Six: Presentation of qualitative results and findings

Results from the qualitative study are presented in this chapter. Interviews were the main data collection tools employed. Data from the interviews are analysed using content analysis. A summary table at the end of the chapter summarises the qualitative data analysis.

Chapter Seven: Discussion and interpretation of findings

A presentation and discussion of the combined analysis for the study is included in this chapter. The relationship between findings for the quantitative study are explained through the lens of the emergent conceptual model. Insight from the derived quantitative and qualitative results point out defective RE activities and suggest strategies for improvement to advance the development culture into producing secure code.

Chapter Eight: Summary, conclusions and implications of study

The final chapter of the thesis presents the main findings of the research, conclusions and the pertinent recommendations on the basis of the findings. Conclusions are drawn on the basis of the empirical findings. Relevant recommendations are made. Limitations to the study are also

mentioned. The thesis is summarised with an emphasis on results obtained, contribution made by results, recommendations and suggestions for further research.

1.8 Chapter summary

An introduction and background to the study are presented in this chapter. Thereafter, the problem statement was discussed with the key research questions of the study. The research sub-questions and objectives were presented. The significance as well as the scope of the study was outlined. The chapter concluded with a presentation of the research blue print and structure of the study. The next chapter (chapter two) provides a review of pertinent literature pertaining to the study.

CHAPTER TWO: LITERATURE REVIEW

2.1 Introduction

The focus of this research study is on requirements engineering (RE) and as such, mainstream requirements engineering processes from the software engineering literature were assessed. It was important to get an understanding of requirements engineering processes from the voices of subject matter experts as well as a sense of history in this field. Once the terrain was mapped with important concepts introduced, a critical review of security requirements, engineering and requirements prioritisation processes was conducted. The major discourses around security requirements, engineering, and requirements prioritisation were explored and presented in this chapter. Figure 2.1 shows the focussing of topics in the literature review. It outlines the concepts brought to the fore in the literature review.

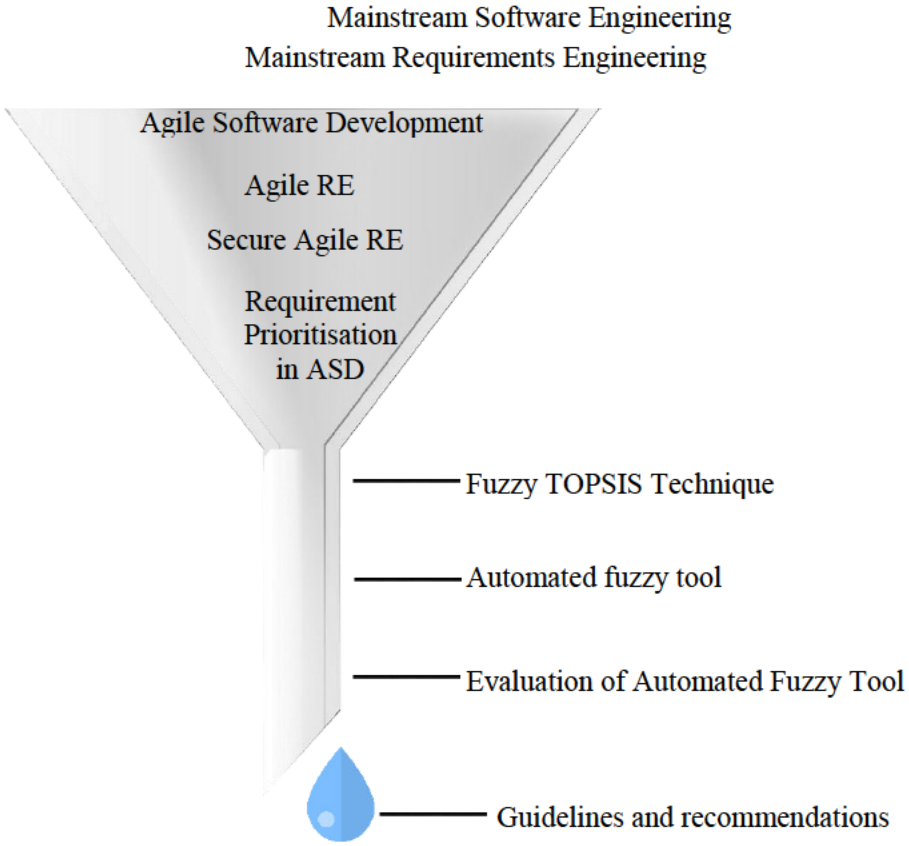


Figure 2. 1: Chapter Overview
Source: Researcher's own construction

The review of the literature begins by first presenting what was considered to be mainstream software engineering and mainstream requirements engineering. Hereafter, the Agile Software Development methodology is introduced. This is followed by the review of requirements engineering in Agile Software Development. Funneling further down, security requirements engineering and requirements prioritisation in Agile RE are explored with the view to identifying the gaps. It was important from previous research studies to examine the views of people involved in attempting to resolve the problems around similar issues and the solutions they bring to the table. Thereafter, fuzzy TOPSIS is presented as an alternate technique for ranking requirements. The automated fuzzy tool is a software app. constructed from scratch using fuzzy TOPSIS as the basis. The chapter concludes with an examination of literature on evaluation frameworks for prioritisation techniques. Guidelines, recommendations and best practice noted from the literature review are factored in this chapter.

2.2 Mainstream software engineering

Conventionally, the phases for software development are planning, a design, analysis, implementation and maintenance. Requirements Engineering occurs in the planning phase which is the earliest phase of software development. Software engineering has evolved with the introduction of new approaches such as Agile Software Development, Prototyping and Commercial off-the-shelf solutions (COTS) where the focus has shifted to rapid development with quick software releases. New approaches require thorough assessment to ensure that the quality of the software product is not compromised. The focus of this research study was on requirements engineering. There was a need to look at mainstream RE processes first, in order to understand shifts that have taken place in the industry.

2.2.1 Mainstream Requirements Engineering

It was important to first understand what RE entails and thereafter assess typical mainstream RE activities. Kumar et al. (2013) defined RE as a software engineering process aimed at identifying requirements, analyzing requirements, preparing documentation and validating requirements, whilst Pressman and Maxim (2015) described RE as obtaining requirements by using a number of activities, techniques and tools. In this phase of software development, requirements engineers

try to elicit information from customers around features of the new system through a series of techniques such as brainstorming sessions, interviews and focus groups (Wurfel et al. 2016).

Academics and practitioners have advocated that project success is dependent on adherences to well defined RE processes. However, there is no agreement of the exact composition of processes in the RE phase in literature. For example, Sommerville (2016) defined the mainstream RE process as comprising of five stages, namely, requirements elicitation, analysis and negotiation, documentation, validation and lastly management while Pressman and Maxim (2015) described seven distinct phases of RE, namely, inception, elicitation, elaboration, negotiation, specification, validation, and management. The 'inception' stage described by Pressman and Maxim (2015) is covered in the 'elicitation' phase of the Sommerville (2016) description. Although the experts name the processes differently, the activities considered in RE are generally similar.

Pressman and Maxim (2015), who have decades of experience as software engineers, academic practitioners, authors and field consultants, outlined the main tasks in each stage of RE as summarised below:

- **Inception:** The business case and project scope are defined by stakeholders from the business community. All stakeholders are identified and it is important for developers to recognize the various view points of the system held by these stakeholders. For example, sales and marketing people are interested in features and functionality of the system that will make it easy for them to sell; system administrators will be interested in authorized users of the system, while software developers may focus on the maintainability of the system. Alert requirements engineers must identify stakeholder's agreement or disagreement in requirements.
- **Elicitation**: Elicitation approaches used must ensure that problems associated with scope, understanding and volatility are addressed. Common techniques to elicit information are brainstorming, meetings, focus groups and surveys. A quality management technique such as Quality Function Deployment (QFD) is used to translate customer needs into refined requirements for the software development. The work products of this phase are a feasibility statement, the scope of the project, a customer list, an end user list, a description of the technical environment of the system, a list of requirements, usage scenarios and prototypes developed.

- **Elaboration:** Focuses on developing a refined requirements model. In Elaboration user scenarios describe how end users interact with the new system.
- **Negotiation*:*** Customers and users sometimes propose conflicting requirements. Developers must resolve conflicts through review meetings. Stakeholders rank requirements and negotiate on any conflicts in the prioritisation process. It is imperative that a win-win result is achieved in conflict resolution.
- **Specification*:*** Requirements are specified in a written document that includes graphical models, formal mathematical models (if necessary), usage scenarios and any prototypes. Non-functional requirements must be specified as well.
- **Validation:** Requirements validation ensures that all specified requirements stated are free of ambiguity, inconsistencies, and omissions. Any detected errors must be corrected. Validation is the process of checking whether the specification captures the customer's needs. Typically the validation is conducted by all stakeholders, but specialist validation teams can be used as well.
- **Requirements management:** In Agile Software Development, changing requirements is expected and a part of the software development methodology. Change management is a process to assist the development team to manage requirements. It enables requirements to be identified, controlled and tracked.

2.2.2 Mainstream Security Requirements Engineering (SRE) approaches

Owing to vulnerabilities in applications, security requirements engineering (SRE) has emerged as an important field in RE to counter security issues at the onset of software development. SRE is discussed to highlight the attempts of software development industry to resolve the security problem. It is important to understand SRE first, before assessing the security gaps in Agile RE. Security Requirements Engineering (SRE) is developing as a process of software engineering that enables requirements engineers to capture security needs of the system in order to build systems with necessary security needs (Salini & Kanmani 2011). It is a systematic way to elicit security requirements to protect the system from potential attacks (El-Hadary & El-Kassas 2014). SRE allows developers to predict vulnerabilities and counter these vulnerabilities before the system is delivered. The extent that SRE approaches are considered in Agile RE was investigated in this research study.

Security literature abounds with SRE frameworks or methodologies that guide secure software development activities for requirement engineering. There are many ways for categorizing these frameworks. For example, Elahi (2009) organised security engineering frameworks provided by researchers and practitioners into four main categories, namely, Agent and Goal Oriented Requirements Frameworks, Trust-Based Requirements Frameworks, Risk and Threat-Based Requirements Frameworks and Unified Modeling Language (UML)-Based Requirements Engineering as shown in Table 2.1. A brief description of each category is given in the table.

Framework type	Framework name	Developer	Brief description
Agent and Goal Oriented Requirements Frameworks	Anti-Model Analysis	Lamsweerde (2004)	Two models are developed iteratively and concurrently. First the model of the system to be and secondly the anti-model.
	Social Actor Analysis	Liu et al (2003)	Analyse attackers and assume all actors are potential attackers.
	Secure Tropos	Giorgini et al. (2002)	Security diagram is constructed and security constraints are imposed on the stakeholders.
Trust -Based Requirements Frameworks	Security Requirements Engineering	Hayley et al. (2004)	Minimise or eliminate the trust assumption between various components. Security requirements are treated as obstacles on the systems functional requirements.
Risk and Threat-Based Requirements Frameworks	CORAS Framework	Braber et al. (2007)	Risk management methods. Integrating requirements engineering practices with security engineering.
UML- Based Requirements Engineering	Misuse Case Analysis	Sindre et al. (2001)	UMLsec is an extension to UML. Expressing security relevant information in a UML diagram. Identify critical assets.
	Abuse Case Analysis	McDermott & Fox (1999)	

Table 2.1: Security Requirements Engineering Frameworks
Source: Created from Elahi (2009)

These frameworks vary on how security requirements are derived. For example, the *misuse case* and *abuse case* frameworks deal with security from the viewpoint that users misuse the functionality of the system (Salini & Kanmani 2011), while *risk based* frameworks extract security

requirements to mitigate the risks (Souag et al. 2012). The challenge posed here is for requirements engineers in Agile RE to develop a systematic approach using a framework or a combination of frameworks that can be incorporated into their RE activities with minimum overheads. However, in order for requirements engineers to utilise a given security approach or some variation of it, training on how to elicit, analyse, and specify security requirements is required (Salini & Kanmani 2011). Furthermore, the requirements engineer must have specialist knowledge on how to identify system assets and analyse threats and vulnerabilities (El-Hadary & El-Kassas 2014). Security requirements elicitation is therefore not necessarily common practice in industry.

Haley et al. (2006) proposed a SRE framework by which comprised the following stages namely, functional requirements identification, identification of security goals, identification of security requirements and finally the construction of satisfaction requirements. Salini and Kanmani (2011) proposed a knowledge oriented framework called Model Oriented Security Requirements Engineering framework. It incorporates security into RE and comprises an inception stage directly followed by elicitation of requirements then followed by an elaboration stage and thereafter negotiation and validation of requirements is conducted and lastly a specification of requirements stage.

The stages of Model Oriented Security Requirements Engineering (MOSRE) are explained below (Salini & Kanmani 2011):

- **Inception:** Identify the objective of the software system and identify the stakeholders and the systems assets. Stakeholders include all people who have a vested interest in the system such as developers, end users, customers and security experts. Techniques like brainstorming, interviews and questionnaires should be used.
- **Elicitation:** Select an elicitation technique. A suitable elicitation technique should be chosen based on the requirements engineer's choice, how much of security they want to achieve, cost versus effort ratio, and lastly the organisational security policies. Elicitation techniques that are advised include Quality Function Deployment (QFD), Soft Systems Methodology (SSM), Joint Application Development (JAD) and Controlled Requirements Expression (CORE). After eliciting requirements, they are grouped into core and non-core requirements. Following this use, cases for the computer based system are generated. Security goals and objectives for

the system must be identified before following identifying threats and potential violations. The next step should be to conduct a security risk assessment to identify where threats and vulnerabilities occur. This should be followed by grouping and prioritising the potential violations and threats for mitigation. Misuse cases for the system should be generated before security requirements are identified. Finally, use cases for security requirements must be generated.

- **Elaboration:** Structural analysis models must be generated and UML diagrams must be developed to understand the security based software system better.
- **Negotiation and Validation:** This stage must be characterized by requirements prioritisation. Ranking must be based on the security.
- **Specification:** Requirements must be specified and they must be validated with stakeholders.

The steps of MOSRE framework proposed by Salini and Kanmani (2011), indicate the ordered flow of activities required in secure RE, namely: outline the objective of application under development, determine who the stakeholders of the system are, identification of assets of the system, choose an appropriate elicitation technique, draw a system architecture diagram, gather requirements, construct use case diagrams, identify security objectives, conduct security risk assessment, create misuse case diagrams, identify security requirements, draw use cases considering security requirements, generate structural analysis models, develop UML diagrams, negotiate and validate requirements and lastly specify requirements.

In another approach, El-Hadary and El-Kassas (2014) proposed an approach for integrating security with requirements engineering for software systems based on problem frames which include the following activities namely, system modeling, asset identification, threats and vulnerabilities identification, security requirements elicitation and security requirements validation. In addition to SRE frameworks, organizations can be guided by the ISO quality assurance standards for the security of their software applications. Security is defined in terms of the following: authenticity, confidentiality, accountability, integrity, availability, non-repudiation and reliability. Wall et al. (2015) advised that organisations must utilise security standards to create security goals and objectives to protect system assets from violation.

Requirements engineers can use SRE frameworks and ISO standards to improve the security of applications. In addition to this, Tondel et al. (2008) recommended eight tasks in RE to fulfill security requirements, namely: definition of key concepts; determining objectives or high-level business goals; identification of threats; asset identification; standards for coding; prioritisation and categorization of the security requirements; inspection and validation of requirements and process planning of security activities. Practitioners can use this as a checklist into their own security practices in RE.

The researcher used these eight tasks identified by Tondel et al. (2008) as an instrument to assess and compare various security approaches highlighted in the literature review. This is shown in Table 2.2.

Approach	Definitions	Objectives	Misuse/ threats	Assets	Coding standards	Categorize & prioritise	Inspect & validate	Process Planning
Haley (2004)	X	✓	✓	✓	X	X	✓	X
Salini & Kanmani (2011)	X	✓	✓	✓	X	✓	✓	✓
El-Hadary & El-Kassas (2014)	X	X	✓	✓	X	X	✓	X

Table 2.2: Comparison of security approaches
Source: Researchers own construction

Table 2.2 indicates that from all the approaches mentioned, the approach adopted by Salini and Kanmani (2011) is the most comprehensive in terms of the Tondel et al. (2008) checklist. However, its suitability to a process model such as ASD must be tested. To ensure that an adequate level of security is built into systems the following security problems related specifically to RE, highlighted by Salini and Kanmani (2011), is summarised below:

- Requirements engineers view security as constraints to the features of the system. Therefore RE practices adopted very often place less focus on security features of the system as this is a non-functional requirement leaving the system vulnerable;
- Security requirements are frequently accidentally replaced with security specific architectural constraints;
- Security requirements tend to be more generic and not application specific;
- SRE more often is an independent process and not incorporated into mainstream RE activities.

To add to this, there are broader challenges related to security and software applications development from an organizational perspective. Organisations are focused on delivering features and have ongoing time-to–market pressures related to increases in system complexities (Schön et al. 2017). Adequate security education and training is essential for the development team to address the scarce security skills and tools in industry (Elahi, G. et al. 2011; El-Hadary & El-Kassas 2014). There is little support from senior management with respect to adequate funding for security (Schön et al. 2017). Development teams are not receiving regular updates about common vulnerabilities from security knowledge sources to keep them informed of the latest security developments (AlBreiki & Mahmoud 2014). Finally, security most often conflicts with performance and efficiency therefore a trade-off analysis between all requirements is critical (Elahi et al. 2011).

In summary, researchers support the contention that RE is a critical success factor for projects and a low emphasis in non-functional requirements such as security during RE can result in project failure (Cao & Ramesh 2010; Elahi et al. 2011; Salini & Kanmani 2011; Pressman & Maxim 2015). Researchers and security specialists are in agreement that security requirements are customised to the system being developed and must protect essential services and assets of the system (Souag et al. 2015). Some suggest that much of the problems associated with security requirements can be eradicated by elevating them from non-functional requirements to functional requirements (Salini & Kanmani 2011). Researchers support the contention that security requirements engineering activities must occur within RE processes and more effective approaches are needed for secure systems development (Salini & Kanmani 2011; El-Hadary & El-Kassas 2014).

2.2.3 Mainstream prioritisation of requirements

The challenges such as limited resources, lack of skill and limited budget experienced in software development can be combated by the requirements prioritisation process (Achimugu et al. 2014). The ranking of requirements occur within RE processes. Traditional process models such as the waterfall and spiral model consider requirements prioritisation as a part of a linear requirements engineering process. However requirements prioritisation in Agile Software Development (ASD)

is unlike traditional prioritisation of requirements. After the elicitation of requirements, software requirements prioritisation occurs during the analysis and negotiation stage of RE. This complex process involves multiple decision makers analyzing requirements and negotiating to decide the order of implementation based on multiple decision making criteria. This stage is followed by documentation where requirements are specified. The final stage of traditional RE is validation (Tuunanen & Kuo 2015).

2.3 Agile Software Development (ASD)

Before discussion of the methodology, it is important understand why the word 'agile' is used to describe a software development process model. Yang and Li (2002), in pioneering research on agility evaluation, provided an original taxonomy of the term 'agility' referring to a winning process improvement strategy employed by enterprises in the manufacturing industry to gain competitive advantage. An organization is 'agile' if it can accommodate change and be flexible in its approach (Lin et al. 2006). Vinodh et al. (2010) in similar research in the manufacturing industry contended that 'agility' centers on an organisation being capable on responding quickly to the needs of their customers.

The term 'agility' has been extended to software development industry as a reaction to the problems experienced in traditional software development. 'Agility' is described as effective software development team communication, making the customer part of the software development team, responding to changes by the customer and self-organising software development teams so that it is in control of the work performed (Pressman & Maxim 2015). 'Agility' yields rapid and incremental delivery of software. Unlike like traditional software development, Agile Software Development teams are highly interactive, cross-functional and closely knit (Inayat & Salim 2015).

Agile Software Development described as a 'light weight' software development methodology was developed to combat the challenges of conventional software development models. In contrast to the traditional approach, ASD methodologies emphasize continuous design of the system, flexibility in project scope, high level of customer interaction and embracing change (Serrador & Pinto 2015).

The four principles that underpin ASD outlined in the ASD Manifesto are as follows (Kavitha & Thomas 2011):

- **Individuals and interactions** over processes and tools.
- **Working software** over comprehensive documentation.
- **Customer collaboration** over contract negotiation.
- **Responding to change** over following a plan.

ASD processes are adaptable and make use of an incremental software development strategy. ASD methods rely on an involved customer from the beginning of the project when goals are set to providing feedback on a regular basis when required (Williams et al. 2015). The iterative nature of ASD allows for frequent stakeholder interaction, adjustments made immediately and project requirements can be re-scoped in light of new information (Serrador & Pinto 2015). There are many ASD methods used in industry such as Extreme Programming (XP), Scrum, Feature-Driven Development (FDD) and Dynamic System Development Method (DSDM).

Extreme Programming is characterized by planning, designing, coding, testing and release stages (Pressman & Maxim 2015). Core practices in XP are collection of user stories, acceptance testing by the client, designing of prototypes, refactoring, pair programming in coding and lastly unit testing. In scrum, development occurs in short iterations called sprints. Requirements for a sprint are obtained from the product backlog which stores all unimplemented requirements. When a sprint ends, stakeholders meet to review the work completed. Scrum has the following roles: Product owner (represents customers interest especially in prioritisation of requirements); Scrum master (removes obstacles from the team) and team member (responsible for completing the work). FDD is an iterative and incremental software development process delivering client-valued features. Finally DSDM is based on rapid application development and comprises of four stages, namely: the feasibility study, the business study, functional model iteration, design and build iteration and implementation.

Figure 2.2 illustrates a generic ASD Process Model.

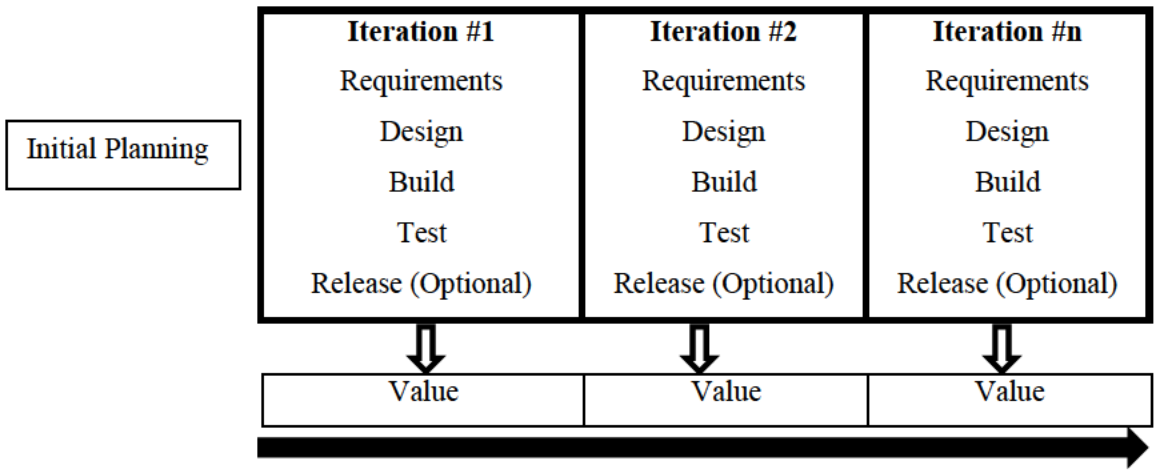


Figure 2.2: ASD Process Model
Source: Created from Pressman and Maxim (2015)

The ASD process is more complicated than traditional approaches as it is iterative, dynamic and open to changes (Kavitha & Thomas 2011). In ASD environments, software development is incremental therefore planning is an ongoing activity as opposed to traditional approaches where it occurs once off, at the beginning (Serrador & Pinto 2015). Gathering user requirements is a critical task in ASD planning. The overall project scope is defined in the initial planning stage and revisited in each iteration as shown in the figure above. The iterations also mean that user requirements are continuously prioritised. The customer works closely with the development to validate the product being delivered (Inayat et al. 2015).

ASD has brought huge benefits to the software development industry however fixation on rapid releases and massification of code have led to more and more ad hoc practices in software development. The focus of this study is on RE processes and how it is streamlined into ASD. Researchers are in agreement that RE is a significant ASD stage and should be the subject of further investigation. In this regard, in a global software engineering study, Nasir and Sahibuddin (2011) identified user requirements gathering as a significant critical success factor. To concur with this finding, a systematic literature review that investigated critical success factors for ASD projects conducted by Sheffield and Lemetayer (2013) found that user requirements can cause a project to fail. Instances where user requirements are of a high quality and satisfy customer needs

is not enough to conclude that the software product is of a high quality. It is necessary to conduct in-depth research to investigate RE, to uncover practices that can comprise the quality of the ASD product. This research study focuses on whether the flexibility offered by the ASD ideology in RE, has led to weaker security of the application. The nature of RE, in Agile Software Development as well as approaches to Agile RE, from the extant of literature is discussed in the next section.

2.3.1 Agile Requirements Engineering

In this section, Agile RE is discussed to gauge what researchers in the field have reported on to get a sense of what gaps exist. Software developers transitioning to ASD had to make a switch from mainstream RE to Agile RE, with cumbersome mainstream processes being bypassed for ad hoc activities. Further, Agile RE differs from mainstream RE because it is ongoing (Fontana et al. 2014). As such, there is a need to assess ASD practices in the literature survey before commencing with the fieldwork. An assessment is important to gauge efforts with regards to the security of applications. Proponents of ASD have challenged mainstream RE by adopting Agile RE. Inayat et al. (2015) proposed that the term Agile RE be used to describe a process derived from flexibility in approaching mainstream RE activities. Whatever the RE approach, researchers are in agreement that RE plays an important role in ASD. To this end, Fontana et al. (2014), in presenting an ASD maturity model, included Agile RE to represent a key area.

In Agile RE, requirements gathering is complex because it is iterative (ongoing) and can change unlike traditional requirements engineering. By accommodating changes in user requirements ASD ensures that the output of the process is high quality requirements and customer satisfaction (Pressman & Maxim 2015). Therefore, ASD literature emphasizes that in gathering requirements it is important to focus on some key aspects. In this regard, Inayat and Salim (2015) advise that owing to the high volatility of requirements constant collaboration is important among stakeholders. They cited communication and awareness as the most relevant socio-technical aspects of collaboration. While Wurfel et al. (2016) suggested that requirements engineers must be mindful of vagueness in requirements from customers and must remain open. Openness prevents requirements engineers from being influenced by existing software that they may have developed for another customer. There are various approaches to requirements engineering in ASD

literature. Kumar et al. (2013) proposed a Joint Application Development (JAD) model for RE in ASD. In a JAD session developers, customers and other stakeholders are brought together to discuss requirements of the new product, whilst Wurfel et al. (2016) used a grounded theory approach to requirements engineering. The approach consisted of the following stages: collect data; transcribe videos; open coding; axial coding and selective coding.

Kassab (2014) proposed five focus areas of Agile RE namely, elicitation, analysis and presentation, management, effort estimation and tools. Kassab (2014) concluded from a survey on existing Agile RE practices, based on the five focus areas of 247 IT professionals from 23 countries, that RE processes within Agile RE are not very clear. Also the main difference between RE processes and Agile RE activities is that there is a lack of documentation in Agile RE. He highlighted requirements management as an important stage in Agile RE to ensure traceability of requirements. On the positive side he found that Agile RE is effective to get continuous feedback from the customer thereby validating the development of the new system.

Inayat et al. (2015) found 17 Agile RE practices by conducting a systematic literature review of Agile RE research papers from 2002 to 2013. The Agile RE practices summarised from 21 papers in this period are namely, customer involvement and interaction, face-to face communication, user stories, requirements prioritisation and reprioritization, iterative requirements, cross functional teams, change management, review meetings and acceptance tests, testing before coding, requirements modelling, pairing for requirements analysis, requirements management, shared conceptualizations, code refactoring, retrospectives and continuous planning. Furthermore, the systematic review conducted by Inayat et al. (2015) summarised the challenges of Agile RE as follows: inappropriate architecture, customer availability, requirements change and its evaluation, neglecting of non-functional requirements, customer inability and disagreement, contractual limitations, budget and time estimation and minimal documentation. The findings are important as they can seriously impact the ASD product. A significant challenge apt to this research is the neglecting of non-functional requirements.

In summation, the ASD literature suggested that although mainstream RE processes is fairly well defined in RE, the practitioners of software development do not clearly agree on standardised Agile

RE practices. The lack of coherent and consolidated views on Agile RE makes this a rich research area (Inayat et al. 2015). This study will conduct an analysis of Agile RE activities by practitioners in industry to gain more insight and provide critical review of the approaches used. The impact of the approaches used on decisions taken to elicit and implement non-functional requirements such as the security of the system was of particular interest in this research study.

2.3.2 Security Approaches in Agile RE

In Agile RE, it is not evidently clear how SRE takes place in a rapid release software development approach. The Agile Manifesto created in 2001, provide firm guidelines to software developers in order to improve traditional practices. Fifteen years later in 2016, the Agile Security Manifesto was launched to combat security problems that plagued software developers. According to Cigital (2016), the four principles of the Agile Security Manifesto are as follows:

1. **Rely on developers and testers more** than security specialists.
2. **Secure while we work more** than after we're done.
3. **Implement features securely more** than adding on security features.
4. **Mitigate risks more** than fix bugs.

Principle 1 states that it is not possible for all companies to have security teams or specialists. The process is rapid and lightweight therefore the security of the system is dependent on the ASD team that owns the security. This means that they must be trained and be aware of security. According to principle 2, security must be incorporated into ASD practices and should not be added as an afterthought. Principle 3 encourages the use of tried and tested security measures, for example security frameworks such as authentication and password storage, to prevent developers from being detracted from focusing on business value. Finally principle 4 promotes security risks assessments over an ad hoc approach to security (Cigital 2016).

The SRE frameworks and approaches provided by security researchers and practitioners in literature are complex and not customised for light weight Agile RE practices. The Agile Security Manifesto supports that they must be blended naturally into Agile RE activities by requirements engineers who must be trained and be knowledgeable about application security issues.

Requirements engineers must have an understanding of what type of security requirements must be included in the system.

Sommerville (2016) advised nine types of security requirements that must be included in any system namely: (R1) the system must identify its users before it can interact with them; (R2) requirements must be able to authenticate these users; (R3) requirements must specify privileges and access rights of identified users; (R4) requirements must protect the system against viruses, malware, worms and other similar types of threats; (R5) requirements must specify how data corruption can be avoided; (R6) requirements must ensure that a party in a transaction cannot deny involvement in that transaction; (R7) requirements must ensure that data privacy is maintained; (R8) requirements must ensure that system requirements can be checked and audited; (R9) requirements must specify how an application can prevent authorised changes from accidentally defeating its security mechanisms. For example, a security requirement for R7 is *the system shall not allow unauthorized individuals or programs access to financial information* and the security requirement for R1 is *the application shall identify all its users before allowing them access to its capabilities*. Similar security requirements can be generated for R2, R3, R4, R5 and R6.

Boström et al. (2006) proposed how security can be factored into Agile XP requirements engineering practices. The approach focused on abuser stories. This means that it takes into consideration the actions of the hacker who intends to abuse the system and mitigates against these actions. The following Agile RE stages were suggested: Identification of critical assets of the system; formulation of abuser stories; assessment of the abuser story to determine risk; negotiation between abuser and user stories; identify security user stories; definition of security-related coding standards and lastly cross-checking abuser stories and countermeasures against system abuse.

The extent that similar security approaches are practiced in Agile RE is the subject of investigation in this research study. In ASD literature, there is no single and distinct approach on how security requirements can be elicited or where in the Agile RE process will the security approach fit. Clearly, researchers have not shown preference for implementation of any particular SRE approach in Agile RE.

Associated with security, is the prioritisation of requirements. In Agile RE, the prioritisation process can significantly impact the security of the system. At the beginning of every iteration, requirements are prioritised. Only those requirements that are ranked highly by stakeholders get implemented. When security requirements are not highly ranked they do not get implemented. In this way the requirements prioritisation process influences the security of the system. Not only does the security approach affect the security of the system but the requirements prioritisation process can also impact the security of the system. The next section of the literature review introduces analysis and prioritisation practices of requirements.

2.3.3 Analysis and Prioritisation of Requirements in Agile RE

2.3.3.1 Requirements Analysis

Requirements analysis occurs immediately after elicitation. In analysis, requirements are examined for complexity, completeness, ambiguity and contradictions (De Lucia & Qusef 2010). Conflicts are resolved through prioritisation and negotiation with stakeholders. When conflicts occur they are resolved through a JAD workshop session where all stakeholders get involved to negotiate and resolve conflicts. System models are also an important part of requirements analysis. Models drawn on a whiteboard or software tool under categories ‘models to be implemented’; ‘models under implementation’ and ‘models completed’ provide a visual representation of the project status. All system models are documented (De Lucia & Qusef 2010). Once conflicts are resolved the list of requirements are now ready to be prioritised. Following the analysis of requirements, the refined list of candidate requirements are sent for prioritisation.

2.3.3.2 Prioritisation and Reprioritisation in Agile RE

Requirements prioritisation is defined as process of ordering requirements based on the importance of the requirement. In ASD requirements prioritisation occurs during the initial planning stage and then at inter-iteration time. Prioritisation and reprioritisation is based on business value. Continuous requirements prioritisation is thus a core activity of Agile RE (AL-Ta'ani & Razali 2013). This ensures that the highly ranked requirements that are important to the project get developed first. This will bring the most business value to the customer and lowers the project risk (Racheva et al. 2010).

However, practitioners are confronted with difficulties in making decisions about which requirements should be considered inter-iteration time. Although developers are very skilled and are the most influential stakeholders, it is the client who makes the final decision on the priority of a requirement. However, what takes place in practice may not necessarily be what is recommended in the literature. Developers will not allow an uninformed client to make decisions that can cause a catastrophe in the project. Furthermore, it is difficult to reach consensus when there are multiple stakeholders and multiple decision making criteria (AL-Ta'ani & Razali 2013). Clearly both developers and clients have a role in the decision making and their input on the value of the requirement in real world practice will be sought in this study.

There are various criteria identified in literature for requirements prioritisation. A pertinent example is a criterion called 'negative value' introduced by Racheva et al. (2010). This means the requirement is valued in terms of how much the system would detract if it is not implemented. Table 2.3, synthesized from ASD literature, provides typical prioritisation criteria that can be applied using an appropriate prioritisation technique.

Criteria	Author
Size (user stories)	Racheva et al. (2010)
Effort	Hasan et al. (2010)
Cost	Hasan et al. (2010)
Risk	AL-Ta'ani and Razali (2013)
Business value	Racheva et al. (2010)
Negative value	Racheva et al. (2010)
Complex	AL-Ta'ani and Razali (2013)
Dependent	AL-Ta'ani and Razali (2013)
Volatility	AL-Ta'ani and Razali (2013)
Easy to use	Hasan et al. (2010)

Table 2.3: Criteria for Requirements Prioritisation
Source: Synthesised from Agile literature

Figure 2.3 represents a conceptual framework of the Agile RE prioritisation process as extracted from AL-Ta'ani and Razali (2013). The framework consists of three main parts namely, Environment, Process and Product. The stakeholders, the project constraints and the requirements nature from the Environment influence the prioritisation process. The process outlines the activities involved in requirements prioritisation. The output of the process is high quality

requirements that meets customer's satisfaction. It is these requirements that are used to build the features of the system.

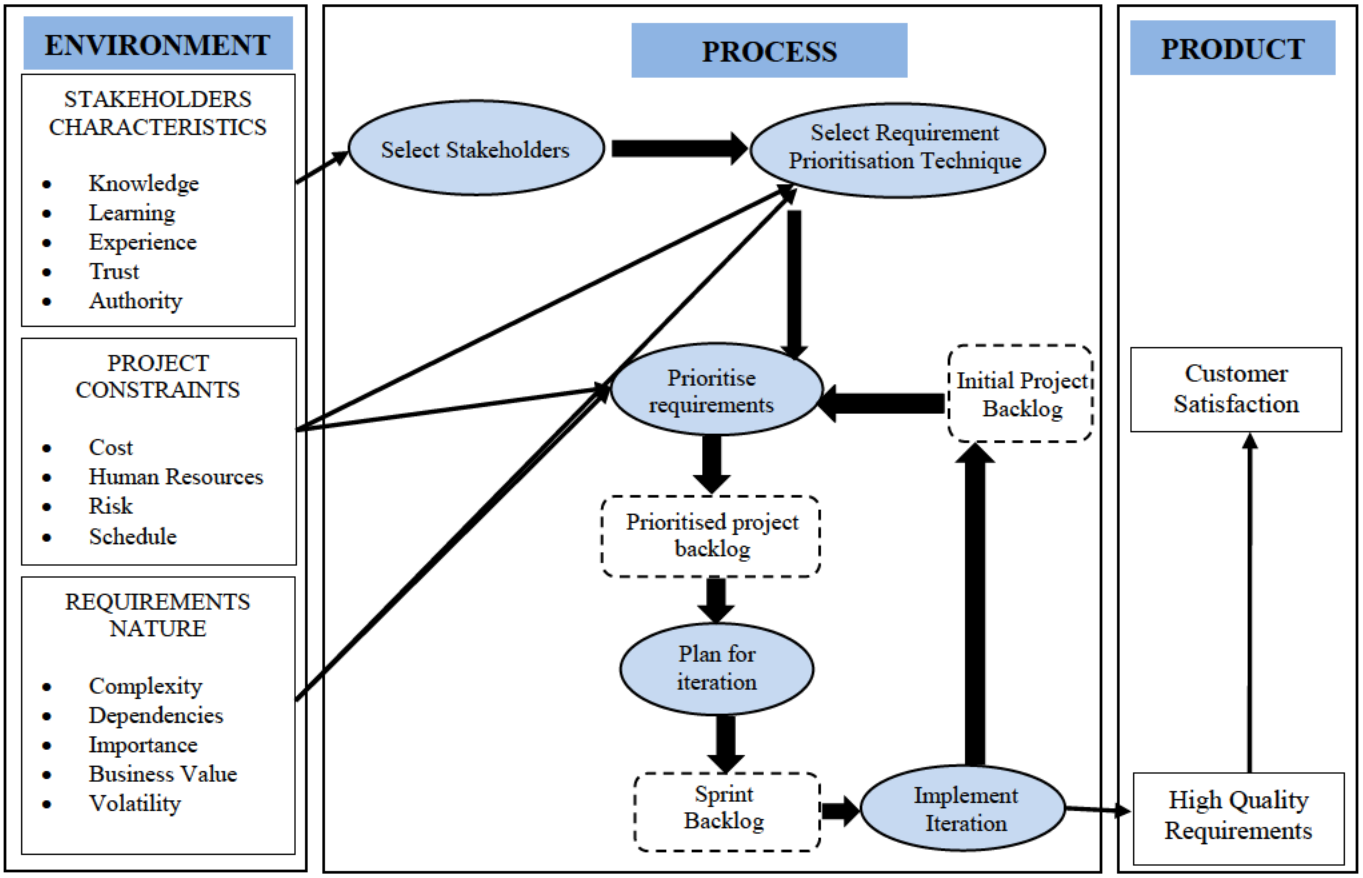


Figure 2.3: A conceptual framework of Requirements Prioritisation Process in Agile Development
Source: Extracted from AL-Ta'ani and Razali (2013)

Requirements selection is accomplished by employing a specific prioritisation technique. Requirements are stored in the product backlog. The outcome of the prioritization process is a prioritised project backlog. The requirements at the top of the backlog will be considered in the first iteration. These requirements are added to the sprint backlog. When a requirement cannot be implemented in the sprint backlog they are sent to the initial project backlog for reprioritisation before the next iteration starts. Besides dropping requirements, adding new requirements is also brought into the initial project backlog for reprioritisation. During the reprioritisation stakeholders rank requirements and identify which requirements will go into the sprint backlog for the next iteration.

An appropriate prioritisation technique is important as an erroneous prioritisation may increase the cost of development and lead to system and project failure (AL-Ta'ani & Razali 2013). Therefore a proper requirements prioritization technique is critical to ensure that the requirements engineering process is efficient (Achimugu et al. 2014). Requirements prioritisation is considered to be a complex multi-criteria decision making process and several prioritisation techniques for ranking requirements exist such as Analytical Hierarchical Process (AHP), JAD, Quality function deployment and Binary search tree. The most cited prioritisation technique for software requirements prioritisation is the AHP. A major drawback of this approach is that requirement engineers must limit the number of criteria and alternatives. In JAD, prioritisation of requirements is done through a viewpoint approach. The viewpoints of the system are in terms of the customer (Achimugu et al. 2014).

A desktop literature review revealed that there are several techniques for ranking requirements (Achimugu et al. 2014). The researcher has proposed a new requirements prioritisation technique. Two reasons are advanced for this. Firstly to add to the existing body of knowledge and secondly, to provide a requirements prioritisation technique that is well suited to Agile RE. To the best of the researcher's knowledge the application of the fuzzy TOPSIS algorithm has not been used to rank requirements. Thus, scope exists for using this technique to rank client requirements including security requirements. A major benefit of fuzzy TOPSIS is that it does not restrict the number of requirements or decision making criteria and is suitable for small and large projects. In this way the new approach combats problems associated with other techniques such as AHP, the most cited technique, which limits the number of requirements and criteria. In the next section, the application of fuzzy TOPSIS as a new technique for requirements prioritisation is presented by first introducing Fuzzy Logic theory.

2.4 Fuzzy Logic Theory

2.4.1 Fuzzy Set Theory

Fuzzy set theory introduced by Zadeh (1965) deals with problems in which there is vagueness in arriving at a decision and this is incorporated into the decision framework. Fuzzy set theory is used to assist decision makers in decision making processes when choices are based on the subjective judgement of the decision maker (Lima et al. 2014). Multi-criteria decision making (MCDM)

problems integrated with fuzzy set theory provides a suitable solution to arrive at a precise judgement based on imprecise information (Lima et al. 2014).

2.4.1.1 Definitions

The definitions of fuzzy concepts are presented below (Matin et al. 2011):

Definition 1*:* A fuzzy set $\tilde{F}$ in the universe of discourse D is defined by a membership function $\mu_{\tilde{F}}(D)$ which assigns each element d in the Universe of discourse D a real number in the interval [0, 1].

Definition 2*:* A triangular fuzzy number (TFN) $\tilde{F}$ has three parameters and represented as a triplet $\tilde{F}$ =(f_1, f_2, f_3) having values in the interval [0, 1] as illustrated in Figure 2.4.

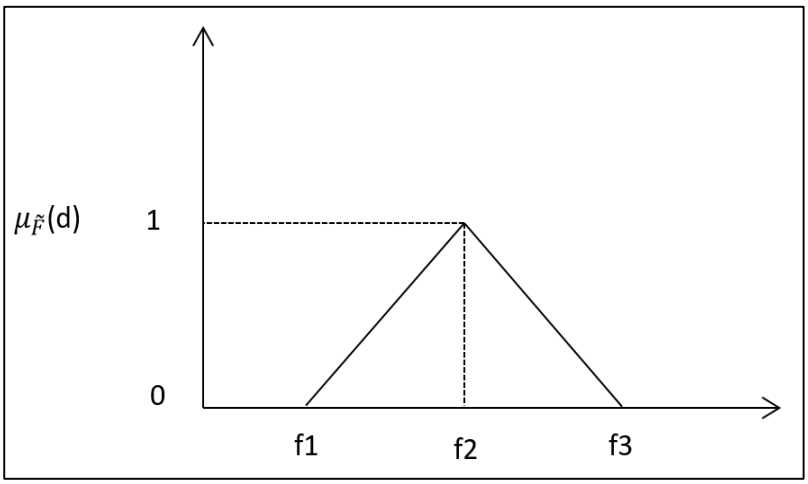


Figure 2.4: Triangular Fuzzy Number $\tilde{F}$
Source: Adapted from Martin (2011)

Definition 3: The membership function $\mu_{\tilde{F}}(D)$ is defined as

$$\mu_{\tilde{F}}(D) = \begin{cases} \dfrac{d - f_1}{f_2 - f_1}, & if\ a_1 \le d \le a_2 \\ \dfrac{f_3 - d}{f_3 - f_2}, & if\ a_2 \le d \le a_3 \\ 0 & Otherwise \end{cases} \qquad (1)$$

where f_1, f_2, f_3 are real numbers

Equation 1: Fuzzy Membership Function

2.4.1.2 Fuzzy algebraic operations

Let k be any real number and let $\tilde{P} = (r_1, s_1, t_1)$ and $\tilde{Q} = (r_2, s_2, t_2)$ two triangular fuzzy numbers the main algebraic operations are as follows (Lima et al. 2014):

(i) Addition: $\tilde{P}\ (+)\ \widetilde{Q} = (r_1, s_1, t_1)\ +\ (r_2, s_2, t_2) = (r_1 +\ r_2, s_1 +\ s_{2,} t_1 +\ t_2)$

(i) Subtraction: $\tilde{P}\ (-)\ \widetilde{Q} = (r_1, s_1, t_1) -\ (r_2, s_2, t_2) = (r_1 -\ r_2, s_1 -\ s_{2,} t_1 -\ t_2)$

(ii) Multiplication: $\tilde{P}\ (X)\ \widetilde{Q} = (r_1, s_1, t_1)\ \times\ (r_2, s_2, t_2) = (r_1 \times\ r_2, s_1 \times s_{2,} t_1 \times\ t_2)$

(iii) Multiplication by a constant *k*

$$k\ (X)\ \widetilde{P} = k\ \times\ (r_1, s_1, t_1) = (k \times\ r_1, k \times s_{1,} k \times\ t_1)$$

2.4.1.3 Cost and benefit criteria

Cost criteria are all criteria that will go against a decision. Cost criteria include criteria such as price, time, effort, stress and loss while benefit criteria are criteria that support a decision ('for a decision'). Examples of benefit criteria are money saved, time saved, stress reduced, personal gain and group gain (Changing Minds 2016). It is cumbersome to use both criteria in a model. Therefore when given a set of criteria, Houška and Dömeová (2003) advised that a transfer from one type of criteria to the other must take place, for example change all cost criteria to benefit criteria (or vice versa). Using one type of criteria (usually benefit) makes calculations much easier. In this research study all cost criteria have been converted to benefit criteria.

2.4.2 Fuzzy TOPSIS

In real life decision makers very often are required to make important choices from several alternatives. The selection of an incorrect alternative based on subjective human judgement can have negative consequences, for example the incorrect choice of project manager in a software development company can impact operating costs and turnover for the company. A more precise way at arriving at decisions involving subjective human preferences is required.

Fuzzy TOPSIS is a method used when a group decision is required for the selection of an alternative from a number of alternatives (possible choices) using multiple decision making criteria to rate alternatives (Matin et al. 2011). Normally a linguistic rating scale such as *very poor, poor,*

satisfactory, *good* and *very good* are used to rate alternatives based on set criteria. The linguistic rating, for example *good* is regarded as vague and imprecise. It is difficult to distinguish which alternative is better from a choice of two alternatives when both are given a rating of *good*. The vague rating *good* can be easily represented as a triangular fuzzy number using fuzzy set theory, for example using a numeric scale of 0-9, *good* represented as a triangular fuzzy number can be written as (7, 8, 9). In the fuzzy TOPSIS approach linguistic ratings are replaced by triangular fuzzy numbers provide mathematically precise ways of arriving at decisions involving human judgement (Sodhi 2012). The final judgement integrates the expert opinion of all decision makers.

In software development, the choice of a requirement (feature of the system) for development from the product backlog (store) requires multiple decision makers who rate requirements based on multiple criteria from a collection of requirements (alternatives). Incorrect decision making can result in low customer satisfaction, losses and hence project failure.

An explanation of two important mathematical concepts that are related to the evaluation of alternatives in fuzzy TOPSIS, namely normalisation and aggregation are discussed before a concrete example is used to illustrate the application of fuzzy TOPSIS to requirements engineering in software development.

- **Normalisation**

A fuzzy triangular number is normalized to ensure that its value is limited to between 0 and 1. Normalisation ensures that triangular fuzzy numbers can be transformed into the interval [0,1], satisfying fuzzy definitions (definition 1 and definition 2 on page 29). Besides ensuring that fuzzy numbers are within the interval [0-1], fuzzy triangular numbers are more easily comparable in normalized form. For example, assume triangular fuzzy number $\tilde{P} = (3,5,7)$ and triangular fuzzy number $\tilde{Q} = (1,2,4)$. In fuzzy data the normalisation is achieved by dividing each entry in the fuzzy triangular number by max value of both the triples (Sodhi 2012). The normalized value for $\tilde{P}$ calculated as (3/7, 5/7, 7/7) is (0.43, 0.71, 1.00) and the normalized value for $\tilde{Q}$ calculated as (1/7, 2/7, 4/7) is (0.14, 0.23, 0.57).

- **Aggregation**

Aggregation is a way of integrating two fuzzy numbers. For example, assume the weightings of triangular fuzzy number $\widetilde{P} = (3,5,7)$ and triangular fuzzy number $\widetilde{Q} = (1,2,4)$ using a rating scale 0-9. The aggregated fuzzy ratings for $\widetilde{P}$ and $\widetilde{Q}$ calculated as (min (1,2); 5+2/2; max(7,4)) is (1, 3.5, 7) (Zahari & Abdullah 2012). In fuzzy TOPSIS aggregation is a way of pooling decision makers scores.

Distance between two fuzzy numbers

$d(\widetilde{P}, \widetilde{Q})$ represents the distance between two triangular fuzzy numbers $\widetilde{P} = (3,5,7)$ and $\widetilde{Q} = (1,2,4)$ expressed mathematically by equation (Sodhi 2012):

$$d(\tilde{x}, \tilde{z}) = \sqrt{\frac{1}{3}\left[(3-1)^2 + (5-2)^2 + (7-4)^2\right]}$$

$$= 2.71$$

2.4.3 Application of Fuzzy TOPSIS to Requirements Engineering

In this section an illustrated example from a small scale project shows the application of fuzzy TOPSIS to rank requirements. To promote understanding a step by step description is given. The mathematical translation follows each step. The example is an application on how fuzzy TOPSIS is used to show the preferences by decision makers to security requirements from a list of security requirements.

A security risk analysis has resulted in four security requirements being identified. Due to constraints in budget, only the top two requirements can be accepted into the product backlog. Prioritise these security requirements for an online book store project, consisting of the following elicited security requirements:

- A1 Sales information to be modified only by members of the sales team.
- A2 Modification to any data in the system must be logged.
- A3 Prevent SQL injections on all input fields.
- A4 Allow authorized users to access the system.

Three decision makers, namely the project manager (DM1), the book store client (DM2) and the business analyst (DM3) must rank the requirements based on the following benefit criteria:

- C 1 Business value: Valuable to the business.
- C2 Size: Capable of finishing in a sprint.
- C3 Effort: Mental effort is manageable to implement in a sprint.
- C4 Core: The system must have this feature.

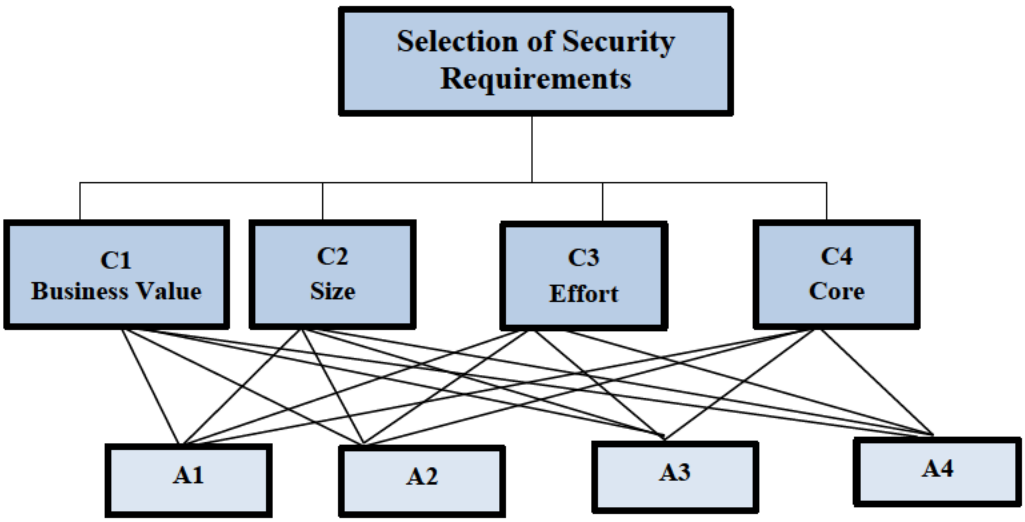


Figure 2. 5: Hierarchical structure of selection process of the Top 2 security requirements

Figure 2.5 shows the hierarchical structure for the selection of the best two requirements. Each decision maker must rate each requirement on the four decision making criteria. This is a multiple criteria and multiple decision making problem. Fuzzy TOPSIS will be used to solve this multi-criteria, multi-decision making problem based on the using complex mathematical equations. The steps for the solution are given below with explanations.

Step 1: Determine the fuzzy weighting of evaluation criteria and fuzzy weighting of alternatives

(i) Linguistic scales for evaluation criteria must be represented as a triangular fuzzy number

Criteria

C1 Business value
C2 Size

C3 Effort
C4 Core

The fuzzy ratings for the criteria expressed as a fuzzy triangular number are illustrated in Table 2.4. The rating scale is determined by decision makers through consensus. In this example the linguistic rating scale ranges are from Very Low (VL) to Very High (VH) and the fuzzy rating scale ranges from 1 to 9. Any appropriate rating scale can be chosen.

Linguistic term	Fuzzy triangular number
Very Low (VL)	(1,1,3)
Low (L)	(1,3,5)
Medium (M)	(3,5,7)
High (H)	(5,7,9)
Very High (VH)	(7,9,9)

Table 2. 4: Fuzzy ratings for the criteria by decision makers

(ii) Represent linguistic scales for alternatives as a triangular fuzzy number. The alternatives in this instance are the set of requirements from the project backlog.

Requirements (Alternatives)

- A1 Sales information to be modified only by members of the sales team
- A2 Modification to any data in the system must be logged
- A3 Prevent SQL injections on all input fields
- A4 Allow authorized users to access the system

Linguistic term	Fuzzy triangular number
Very weak (VW)	(1,1,3)
Weak (W)	(1,3,5)
Average (A)	(3,5,7)
Good (G)	(5,7,9)
Very Good (VG)	(7,9,9)

Table 2.5: Fuzzy ratings for the alternatives by decision makers

Step 2: Expert decision makers must rate the criteria and the alternatives with respect to criteria using appropriate linguistic variables. Construct aggregated fuzzy weights vector for criteria and construct the aggregated fuzzy decision matrix for alternatives.

(i) Decision makers DM_1, DM_2 and DM_3 weigh the criteria using the linguistic rating scale for criteria based on their expert knowledge and experience.

Table 2.6 shows decision makers ratings of criteria, for example criterion 1 (C_1) were rated as H (DM_1), H (DM_2) and L (DM_3) using linguistic scales.

Criteria	Decision-makers		
	DM1	DM2	DM3
C1	H	H	L
C2	H	VH	M
C3	VH	VH	VL
C4	H	H	M

Table 2.6: Linguistic Scales for Rating Criteria by Decision Makers

(ii) The decision makers DM_1, DM_2 and DM_3 ratings must be represented as triangular fuzzy numbers in terms of the numeric rating scale for criteria.

Table 2.7 shows the conversions from a linguistic scale to a triangular fuzzy number. For example the rating for C_1 by DM1 is H (5, 7, 9), DM2 is H (5, 7, 9) and DM3 is L (1, 3, 5) respectively.

Criteria	Decision-maker		
	DM1	DM2	DM3
C_1	5,7,9	5,7,9	1,3,5
C_2	5,7,9	7,9,9	3,5,7
C_3	7,9,9	7,9,9	1,1,3
C_4	5,7,9	5,7,9	3,5,7

Table 2.7: Fuzzy weights for criteria

Now taking C_1 as an example, the aggregated rating for C_1 is calculated as follows (min (5+5+1), 7+7+3/3, max (9+9+5)). Similarly the aggregated rating for all other criteria are calculated. Then assemble the fuzzy decision matrix of criteria as shown in the Table 2.8.

Criteria	Weight		
C1	1	5.7	9
C2	3	7	9
C3	1	6.3	9
C4	3	6.3	9

Table 2.8: Aggregate Fuzzy Weights for criteria

Mathematically aggregate weights of each criterion are calculated as follows:

Let k = number of decision makers

Let j = 1,2, ...n representing the criteria

$$\widetilde{w}_j^k = (w_{j1}, w_{j2}, w_{j3})$$

where: $w_{j1} = \min_k\{w_{jk1}\},\ w_{j2} = \frac{1}{K}\sum_{k=1}^{k} w_{jk2},\ w_{j3} = \max_k\{w_{jk3}\}$ (1)

(iii) Expert decision makers rating for the alternatives with respect to criteria using the appropriate linguistic variables. Thereafter the aggregated fuzzy decision matrix is constructed.

Table 2.9 shows that decision makers rating of alternatives in terms of the criteria based on linguistic the rating scale for alternatives.

Criteria	Alternatives											
	A1			A2			A3			A4		
	DM1	DM2	DM3	DM1	DM2	DM3	DM1	DM2	DM3	DM1	DM2	DM3
C1	A	VG	W	A	W	A	G	VG	VW	G	A	VG
C2	VG	A	VG	W	A	VG	A	G	W	VG	VG	VW
C3	VW	W	A	G	G	W	A	VG	G	VW	A	VG
C4	VG	G	G	VG	W	G	A	VW	A	VG	G	G

Table 2.9: Linguistic assessments for 4 alternatives by decision makers

Table 2.10 is populated by representing each entry in Table 2.9 as triangular fuzzy numbers in terms of the rating scale for alternatives.

Criteria	Alternatives											
	A1			A2			A3			A4		
	DM1	DM2	DM3	DM1	DM2	DM3	DM1	DM2	DM3	DM1	DM2	DM3
C1	3,5,7	7,9,9	1,3,5	3,5,7	1,3,5	3,5,7	5,7,9	7,9,9	1,1,3	5,7,9	3,5,7	7,9,9
C2	7,9,9	3,5,7	7,9,9	1,3,5	3,5,7	7,9,9	3,5,7	5,7,9	1,3,5	7,9,9	7,9,9	1,1,3
C3	1,1,3	1,3,5	3,5,7	5,7,9	5,7,9	1,3,5	3,5,7	7,9,9	5,7,9	1,1,3	3,5,7	7,9,9
C4	7,9,9	5,7,9	5,7,9	7,9,9	1,3,5	5,7,9	3,5,7	1,1,3	3,5,7	7,9,9	5,7,9	5,7,9

Table 2.10: Aggregated fuzzy decision matrix

Now the decision maker's fuzzy ratings are aggregated. Table 2.9 shows that alternative 1 based on criterion 1 was rated as A (DM_1), VG (DM_2) and W (DM_3). The decision makers corresponding

fuzzy rating were A (3, 5, 7), VG (7, 9, 9) and W (1, 3, 5) as shown in Table 2.10 above. Therefore the aggregated rating of the 3 decision makers for alternative 1 using criterion 1 is (1, 5.7, 9) as shown in Table 2.11. In the same way assemble the aggregated fuzzy decision matrix for all other alternatives as shown in the Table 2.12.

Mathematically aggregate weights of each alternative are calculated as follows:

Let k = number of decision makers

Let j = 1,2, ...n represent the criteria

Let i = 1,2,....s represent the alternatives

Aggregate weightings $\tilde{x}_{ij} = (l_{ij}, m_{ij}, u_{ij})$ of alternatives (*i*) with respect to each criterion *(j)* based on the fuzzy ratings by the *k* decision maker is expressed by the calculations below:

$$l_{ij} = \min_k \{l_{ij}^k\},\ m_{ij} = \frac{1}{K}\sum_{k=1}^{k} m_{ij}^k,\ u_{ij} = \max_k \{u_{ij}^k\} \quad (2)$$

The aggregated fuzzy decision matrix is shown in Table 2.11.

Criteria	**Alternatives**											
	A1			**A2**			**A3**			**A4**		
C1	1	5.7	9	1	4.3	7	1	5.7	9	3	7	9
C2	3	7.7	9	1	5.7	9	1	5	9	1	6.3	9
C3	1	3	7	1	5.7	9	3	7	9	1	5	9
C4	5	7.7	9	1	6.3	9	1	3.7	7	1	7.7	9

Table 2. 11: Aggregated fuzzy decision matrix

Mathematically the aggregate fuzzy decision matrix for the alternatives ($\widetilde{D}$) is expressed below:

$$\widetilde{D} = \begin{array}{c} \\ A_1 \\ A_i \\ A_n \end{array} \begin{array}{c} \begin{array}{cccc} C_1 & C_2 & C_j & C_m \end{array} \\ \begin{bmatrix} \tilde{x}_{11} & \tilde{x}_{12} & \tilde{x}_{1j} & \tilde{x}_{1m} \\ \vdots & \vdots & \vdots & \vdots \\ \tilde{x}_{n1} & \tilde{x}_{n2} & \tilde{x}_{nj} & \tilde{x}_{nm} \end{bmatrix} \end{array} \quad (3)$$

Step 3: Normalise the fuzzy decision matrix

Normalise the fuzzy decision matrix of the alternatives ($\widetilde{D}$). All criteria are benefit criteria. A matrix is normalized to ensure that the value for each entry is in the interval [0,1]. It ensures that the values of each criterion are comparable. In fuzzy data the normalization is achieved by dividing each entry in the fuzzy triangular number by max value of the triple for each alternative (Sodhi 2012). For example consider the entry (1, 5.7, 9) for criterion 1 for alternative1 illustrated in Table 2.11. Normalisation is achieved by dividing each value in the triple by 9. Therefore the normalized entry is (1/9, 5.7/9, 9/9) = (0.333, 0.633, 1) as shown in Table 2.12.

	A1			**A2**			**A3**			**A4**		
C1	0.333	0.633	1	0.111	0.478	0.778	0.111	0.633	1	0.333	0.778	1
C2	0.333	0.856	1	0.111	0.633	1	0.111	0.556	1	0.111	0.7	1
C3	0.111	0.333	0.778	0.111	0.633	1	0.333	0.778	1	0.111	0.556	1
C4	0.556	0.856	1	0.111	0.7	1	0.111	0.411	0.778	0.111	0.856	1

Table 2. 12: Normalised Aggregated fuzzy decision matrix for alternatives

Mathematically the normalized fuzzy decision matrix ($\widetilde{R}$) is written as follows:

Let $\tilde{R}$ = normalized decision matrix

Let j = 1,2, ….n represent the criteria

Let i = 1,2,….s represent the alternatives

Let (l_{ij}, m_{ij}, u_{ij}) = aggregated weighting of alternative (i) with respect to criterion (j)

Let $\tilde{r}_{ij}$= triangular fuzzy number of alternative (i) with respect to criterion (j) in matrix r with n rows and s columns

The normalized decision matrix is calculated as follows:

$$\widetilde{R} = [\tilde{r}_{ij}]_{nxs}$$

where $\tilde{r}_{ij} = \left(\frac{l_{ij}}{u_j^+}; \frac{m_{ij}}{u_j^+}; \frac{u_{ij}}{u_j^+}\right)$ and $u_j^+ = \max_i u_{ij}$ (benefit criteria) (4)

therefore $u_j^+ = \max_i$ (9,9,9)

Step 4: Compute the weighted normalised decision matrix

Table 2.13 shows the aggregated weights for each criteria. This table is used together with Table 2.14 to calculated the weighted normalized decision matrix , for example, the weighted normalised value for alternative 1(A_1) is obtained by multiplying the criterion 1 (C_1) the values for C_1 (1, 5.7, 9) from Table 2.13 by the elements (0.333, 0.633,1.00) from the normalized fuzzy decision matrix in Table 2.14.

Criteria	Weight		
C1	1	5.7	9
C2	3	7	9
C3	1	6.3	9
C4	3	6.3	9

Table 2.13: Aggregate fuzzy weights for criteria

	A1			A2			A3			A4		
C1	0.333	0.633	1	0.111	0.478	0.778	0.111	0.633	1	0.333	0.778	1
C2	0.333	0.856	1	0.111	0.633	1	0.111	0.556	1	0.111	0.7	1
C3	0.111	0.333	0.778	0.111	0.633	1	0.333	0.778	1	0.111	0.556	1
C4	0.556	0.856	1	0.111	0.7	1	0.111	0.411	0.778	0.111	0.856	1

Table 2.14: Normalised Aggregate fuzzy decision matrix for alternatives

Table 2.15 shows the completed weighted normalized fuzzy decision matrix for alternatives.

	A1			A2			A3			A4		
C1	0.333	3.61	9	0.111	2.723	7	0.111	3.61	9	0.333	4.433	9
C2	1	5.989	9	0.333	4.433	9	0.333	3.889	9	0.333	4.9	9
C3	0.111	2.1	7	0.111	3.99	9	0.333	4.9	9	0.111	3.5	9
C4	1.667	5.39	9	0.333	4.41	9	0.333	2.59	7	0.333	5.39	9

Table 2. 15: Weighted Normalized Fuzzy Decision Matrix for Alternatives

Mathematically the weighted normalized decision matrix, $(\widetilde{V})$ is computed as follows:

Let j = 1,2, ….n represent the criteria

Let i = 1,2,….s represent the alternatives

The weights of the evaluation criteria, $(\widetilde{w}_j)$ is multiplied by the elements $\tilde{r}_{ij}$ of the normalized fuzzy decision matrix.

$$\widetilde{(V}) = [\tilde{v}_{ij}]_{nxs} \quad (5)$$

Where $\tilde{v}_{ij} = \tilde{r}_{ij}$ x $\tilde{w}_j$

Step 5: Fuzzy Positive-Ideal solution (FPIS) values and the Fuzzy Negative-Ideal solution (FNIS) values are determined

The best performance values for each alternative are referred to as the FPIS and the worst performance values are referred to as FNIS (Vinodh et al. 2010). Table 2.16 shows the weighted normalized fuzzy decision matrix for alternatives. This table is used to calculate the FPIS and FNIS for alternatives under each criterion.

	A1			A2			A3			A4		
C1	0.333	3.61	9	0.111	2.723	7	0.111	3.61	9	0.333	4.433	9
C2	1	5.989	9	0.333	4.433	9	0.333	3.889	9	0.333	4.9	9
C3	0.111	2.1	7	0.111	3.99	9	0.333	4.9	9	0.111	3.5	9
C4	1.667	5.39	9	0.333	4.41	9	0.333	2.59	7	0.333	5.39	9

Table 2.16: Weighted Normalized Fuzzy Decision Matrix for Alternatives

Table 2.17 shows the FNIS and FPIS values. Table 2.16 is used for the calculation of FPIS and FNIS, for example row 1 of Table 2.16 show that the FNIS value is 0.111 and the FPIS value is 9. These values are now written in row 1 of Table 2.17 under FNIS(A-) and FPIS(A+) respectively.

FNIS(A^-)			FPIS(A^+)		
0.111	0.111	0.111	9.000	9.000	9.000
0.333	0.333	0.333	9.000	9.000	9.000
0.111	0.111	0.111	9.000	9.000	9.000
0.333	0.333	0.333	9.000	9.000	9.000

Table 2.17: FNIS and FPIS

Mathematically the Fuzzy Positive Ideal Solution (FPIS, A^+) and the Fuzzy Negative Ideal Solution (FNIS, A^-) is calculated as follows:

Let j = 1,2, ….n represent the criteria

$A^+ = \{ \tilde{v}_1^+, \tilde{v}_j^+, \dots, \tilde{v}_n^+ \}$ (6)

$A^- = \{ \tilde{v}_1^-, \tilde{v}_j^-, \dots, \tilde{v}_n^- \}$ (7)

where $\tilde{v}_j^+$ = (1, 1, 1) and $\tilde{v}_j^-$ = (0, 0, 0)

Step 6: Calculate the distance of each alternative from FPIS and FNIS

Table 2.18 shows the distance values for each alternative. For example, the distance between alternative 1 criterion 1 (0.333,0.361,9.00) from Table 2.16 and FNIS for criterion 1 (0.111,0.111,0.111) from Table 2.17 is calculated as follows:

$$d= \sqrt{\frac{1}{3}\left[(0.333-0.111)^2+(0.361-0.111)^2+(9.00-0.111)^2\right]}$$

$$= 5.517$$

	d^-				d^+			
	A1	**A2**	**A3**	**A4**	**A1**	**A2**	**A3**	**A4**
C1	5.517	4.254	5.515	5.708	5.892	6.388	6.002	5.656
C2	5.987	5.535	5.408	5.656	4.935	5.656	5.809	5.535
C3	4.140	5.599	5.831	5.492	6.599	5.891	5.535	6.035
C4	5.844	5.530	4.064	5.793	4.719	5.662	6.330	5.420

Table 2.18: Distances from FPIS and FNIS for alternatives

Mathematically the distances d_j^+ and d_j^- of each alternative respectively from $\tilde{v}_j^+$ and $\tilde{v}_j^-$ are calculated as follows:

Let j = 1,2, ….n represent the criteria

Let i = 1,2,….s represent the alternatives

$$d_i^+ = \sum_{j=1}^{n} d_v(\tilde{v}_{ij}, \tilde{v}_j^+) \quad (8)$$

$$d_i^- = \sum_{j=1}^{n} d_v(\tilde{v}_{ij}, \tilde{v}_j^-) \quad (9)$$

Where d represents the distance between two triangular fuzzy numbers. For example the distance $d(\tilde{v}_{ij}, \tilde{v}_j^+)$ where $\tilde{v}_{ij}$ =(l_1,m_1, u_1) and $\tilde{v}_j^+$ =(l_2,m_2,u_2) is expressed as follows:

$$d(\tilde{v}_{ij}, \tilde{v}_j^+) = \sqrt{\frac{1}{3}\left[(l_1-l_2)^2+(m_1-m_2)^2+(u_1-u_2)^2\right]} \quad (10)$$

Step 7: Obtain the closeness coefficient (CC_i)

Table 2.19 shows the values for the Closeness Coefficients of the four alternatives. The alternative with the **highest closeness coefficient** represents the best alternative as it is closest to the FPIS and farthest from FNIS.

	A1	A2	A3	A4
d-	21.488	20.918	20.818	22.649
d+	22.145	23.597	23.676	22.647
CCi	**0.492**	**0.470**	**0.468**	**0.500**

Table 2.19: Closeness Coefficients of 4 Alternatives

The Closeness Coefficient values of alternative 1 calculated as 21.488 **(d-)** /(21.488 **(d-)** + 22.145 **(d+)**) is 0.492. Similarly the values for the closeness coefficient of other alternatives in Table 2.19 are calculated.

Mathematically the closeness coefficient (CC_i) is calculated as follows:

$$CC_i = \frac{d_i^-}{d_i^+ + d_i^-} \quad (11)$$

Step 8: Define the Ranking of Alternatives

Define the ranking of the alternatives according to the closeness coefficient, CC_i, in decreasing order. The best alternative is closest to the FPIS and farthest to the FNIS.

Order:

A4 Allow authorized users to access the system

A1 Sales information to be modified only by members of the sales team

A2 Modification to any data in the system must be logged

A3 Prevent SQL injections on all input fields

The results of fuzzy TOPSIS indicate that only A4 and A1 must be sent to the product backlog.

Summarising fuzzy TOPSIS Mathematically

The fuzzy TOPSIS algorithm comprises of the following steps, according to Sodhi (2012):

Step 1:

Decision makers must determine the linguistic and fuzzy weighting of evaluation criteria and alternatives.

Step 2:

(i) Mathematically aggregate weights of each criterion as follows:

Let K = number of decision makers

Let j = 1,2, ….n representing the criteria

$\widetilde{w}_j^k = (w_{j1}, w_{j2}, w_{j3})$

where: $w_{j1} = \min_k\{w_{jk1}\},\ w_{j2} = \frac{1}{K}\sum_{k=1}^{k} w_{jk2},\ w_{j3} = \max_k\{w_{jk3}\}$ (1)

(ii) Mathematically aggregate weights of each alternative as follows:

Let k = number of decision makers

Let j = 1,2, ….n represent the criteria

Let i = 1,2,….s represent the alternatives

Aggregate weightings $\tilde{x}_{ij} = (l_{ij}, m_{ij}, u_{ij})$ of alternatives (*i*) with respect to each criterion *(j)* based on the fuzzy ratings by the *k* decision maker is expressed by the calculations below:

$$l_{ij} = \min_k\{l_{ij}^k\},\ m_{ij} = \frac{1}{K}\sum_{k=1}^{k} m_{ij}^k,\ u_{ij} = \max_k\{u_{ij}^k\} \quad (2)$$

(iii) Mathematically the aggregate fuzzy decision matrix for the alternatives ($\widetilde{D}$) is expressed below:

$$\widetilde{D} = \begin{array}{c} \\ A_1 \\ A_i \\ A_n \end{array} \begin{array}{c} \begin{array}{cccc} C_1 & C_2 & C_j & C_m \end{array} \\ \begin{bmatrix} \tilde{x}_{11} & \tilde{x}_{12} & \tilde{x}_{1j} & \tilde{x}_{1m} \\ \vdots & \vdots & \vdots & \vdots \\ \tilde{x}_{n1} & \tilde{x}_{n2} & \tilde{x}_{nj} & \tilde{x}_{nm} \end{bmatrix} \end{array} \quad (3)$$

Step 3:

Mathematically the normalized fuzzy decision matrix ($\widetilde{R}$) is written as follows:

Let $\tilde{R}$ = normalized decision matrix

Let j = 1,2, ….n represent the criteria

Let i = 1,2,….s represent the alternatives

Let (l_{ij}, m_{ij}, u_{ij}) = aggregated weighting of alternative (i) with respect to criterion (j)

Let $\tilde{r}_{ij}$ = triangular fuzzy number of alternative (i) with respect to criterion (j) in matrix r with n rows and s columns

The normalized decision matrix is calculated as follows:

$$\widetilde{(R)} = [\tilde{r}_{ij}]_{nxs}$$

where $\tilde{r}_{ij} = \left(\frac{l_{ij}}{u_j^+}; \frac{m_{ij}}{u_j^+}; \frac{u_{ij}}{u_j^+}\right)$ and $u_j^+ = \max_i u_{ij}$ (benefit criteria) (4)

therefore $u_j^+ = \max_i$ (9,9,9)

Step 4:

Mathematically the weighted normalized decision matrix, $\widetilde{(V)}$ is computed as follows:

Let j = 1,2, ...n represent the criteria

Let i = 1,2,....s represent the alternatives

The weights of the evaluation criteria, $\widetilde{(w_j)}$ is multiplied by the elements $\tilde{r}_{ij}$ of the normalized fuzzy decision matrix.

$$\widetilde{(V)} = [\tilde{v}_{ij}]_{nxs} \quad (5)$$

where $\tilde{v}_{ij} = \tilde{r}_{ij}$ x $\tilde{w}_j$

Step 5:

Mathematically the Fuzzy Positive Ideal Solution (FPIS, A^+) and the Fuzzy Negative Ideal Solution (FNIS, A^-) is calculated as follows:

Let j = 1,2, ...n represent the criteria

$A^+ = \{ \tilde{v}_1^+, \tilde{v}_j^+, \dots, \tilde{v}_n^+ \}$ (6)

$A^- = \{ \tilde{v}_1^-, \tilde{v}_j^-, \dots, \tilde{v}_n^- \}$ (7)

where $\tilde{v}_j^+ = (1, 1, 1)$ and $\tilde{v}_j^- = (0, 0, 0)$

Step 6:

Mathematically the distances d_j^+ and d_j^- of each alternative from respectively $\tilde{v}_j^+$ and $\tilde{v}_j^-$ are calculated as follows:

Let j = 1,2, ...n represent the criteria

Let i = 1,2,....s represent the alternatives

$$d_i^+ = \sum_{j=1}^{n} d_v(\tilde{v}_{ij}, \tilde{v}_j^+) \quad (8)$$

$$d_i^- = \sum_{j=1}^{n} d_v(\tilde{v}_{ij}, \tilde{v}_j^-) \quad (9)$$

Where d represents the distance between two triangular fuzzy numbers. For example the distance $d(\tilde{v}_{ij}, \tilde{v}_j^+)$ where $\tilde{v}_{ij} = (l_1, m_1, u_1)$ and $\tilde{v}_j^+ = (l_2, m_2, u_2)$ is expressed as follows:

$$d(\tilde{v}_{ij}, \tilde{v}_j^+) = \sqrt{\frac{1}{3}\left[(l_1 - l_2)^2 + (m_1 - m_2)^2 + (u_1 - u_2)^2\right]} \quad (10)$$

Step 7:

Mathematically the closeness coefficient (CC_i) is calculated as follows:

$$CC_i = \frac{d_i^-}{d_i^+ + d_i^-} \quad (11)$$

Step 8:

Define the ranking of the alternatives according to the closeness coefficient, CC_i, in decreasing order. The best alternative is closest to the FPIS and farthest to the FNIS.

A high level diagrammatic representation of the Fuzzy Logic Algorithm is shown in Figure 2.6:

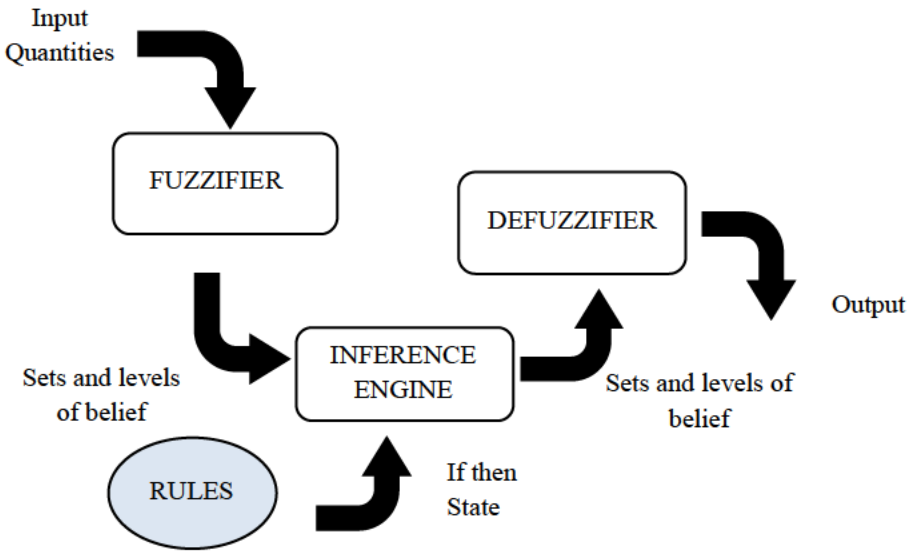


Figure 2.6: The Model of the Fuzzy Logic system
Source: Extracted from Hagras and Wagner (2009)

2.4.4 Evaluation of the Prioritisation Technique

The fuzzy TOPSIS technique has been specifically designed for use in Agile RE. In this regard Hatton (2008) is in agreement that it is important to choose a prioritisation technique that is well suited to the process model. For example, the prioritisation technique for the Waterfall process model will be different from that chosen for ASD. Therefore, over and above the normal benefits around prioritisation techniques discussed previously, a prioritisation technique chosen for use in Agile RE must show suitability to the process model taking into consideration the following (Ramesh et al. 2010):

- RE is conducted in a rapid development methodology and there is not much time available;
- RE is ongoing in ASD with high likelihood that requirements will be added and removed;
- Frequent reprioritisation of requirements;
- Multiple criteria that varies from project to project is used;
- Decision makers can vary from iteration to iteration;
- The number of requirements that need prioritisation is unknown;
- At this stage more information about the requirement is known to the customer than the developer; and
- Customer involvement must be accommodated.

Fuzzy TOPSIS, via the automated fuzzy tool has been designed with the intention of being well suited to Agile RE. In order to validate this, it must be empirically evaluated on the field using an appropriate evaluation framework. Various frameworks are suggested to evaluate prioritisation techniques in the extant of literature.

Hasan et al. (2010) advocated an evaluation framework with the following criteria, namely, ease of use (simple to use); certainty (accuracy of the end result); total time taken (how long it takes to perform prioritization); scalability (use of a large number of requirements) and number of comparisons (how many comparisons required). While Karlsson et al. (1998) considered inherent characteristics such as being able to indicate consistency in the decision makers judgement and describing the scale of measurement that ranking is based on; objective measures such as the required number of decisions, the total time taken by a decision maker to complete the process and the time consumption per decision and lastly subject measures such as ease of use, reliability of

the results and fault tolerance. Hatton (2008) advised the following criteria: quick and easy to perform, easy to deal with additional requirements, category information, ranked information, ratio scale information, suitable for goals and high level requirements and suitable for more detailed requirements.

Tuunanen and Kuo (2015) proposed a more comprehensive framework of evaluation synthesized from research on prioritisation metrics. Their evaluation framework was based on five dimensions, namely, resources, performance, adaptation, design and usability. Each dimension is expanded having sub-criteria as depicted in Table 2.20.

Resources	Performance	Adaptation	Design	Usability
Cost	Efficiency	Expandability	Correctness	Ease of learning
Time	Integrity	Flexibility	Maintainability	Task efficiency
Technologies	Reliability	Interoperability	Verifiability	Ease of remembering
Skills	Survivability	Portability		Understandability
		Reusability		Subjective satisfaction
				Attractiveness

Table 2.20: Evaluation Framework based on Five Dimensions

Source: Extracted from Tuunanen and Kuo (2015)

Through a process of synthesis from the evaluation frameworks discussed and the characteristics of Agile RE presented earlier, the following criteria were chosen to evaluate the fuzzy TOPSIS technique for requirements prioritisation:

- Ease of use (Karlsson et al. 1998; Hasan et al. 2010).
- Total time taken (Karlsson et al. 1998; Hasan et al. 2010; Tuunanen & Kuo 2015).
- Scalability (Hasan et al. 2010).
- Suitability to high level requirements (Hatton 2008).
- Suitability to detailed requirements (Hatton 2008).
- Correctness (Tuunanen & Kuo 2015).
- Scale of measurements (Hatton 2008; Hasan et al. 2010).

2.5 Application security

According to a survey by Positive Technologies (2015), an application security and vulnerability company, 68% of applications had high severity vulnerabilities. AlBreiki and Mahmoud (2014) describe two types of software vulnerabilities namely, "bug and flaw". Bug vulnerability is caused by a problem in the syntax of the code and flaw vulnerability is any non-syntax problem. These vulnerabilities appear during system development.

Lonescu (2015) advised that the best defense against vulnerabilities was to put security measures in place during application development. He contended that developers must be aware of how an attacker works to build defenses to combat this in their applications. A web application gets violated when the following occurs: sensitive data gets leaked, a hacker impersonates a trusted user (identity theft), the application shuts down or the site is unavailable, when there is defacement or content modification and when arbitrary code is executed on the server (remote execution) (Dukes et al. 2013).

There are organizations dedicated to finding and fighting the cause of insecure software providing guidelines and standards for best practice in application security. These security knowledge sources include Open Web Application Security Project (OWASP), Common Weakness Enumeration (CWE) and Computer Emergency Response Teams (CERT) (Dukes et al. 2013; AlBreiki & Mahmoud 2014). CERT maintains and publishes reports of discovered vulnerabilities (Elahi et al. 2011).

OWASP, the applications security standard or guideline used by software developers, provides a list of the top 10 application security risks. A new list is created every three years (Lonescu 2015). The list of common vulnerabilities found in web applications that appeared in the latest OWASP include SQL injection, sensitive data exposure, cross-site scripting, insecure direct object reference, cross-site request forgery, security misconfiguration, broken authentication, missing function level access control, using components with known vulnerabilities, unvalidated redirects and forwards and session management, (Dukes et al. 2013). Table 2.21 provides a brief description and the malicious intent of each of the top 10 vulnerabilities.

No.	Vulnerability	Description	Malicious intent
1	SQL Injection	The direct insertion of SQL commands into unsanitised input fields.	Read and modify data in the database. Assume control of the server
2	Broken authentication & Session management	Impersonates users by gaining access to their users account, session IDs and passwords.	Attacker can do anything the victim can do.
3	cross-site scripting (XSS)	Embeds malicious script (e.g. JavaScript) into unsanitised input fields.	Hijack user sessions.
4	Insecure direct object reference	An attacker usually an authorized user can change a parameter value that references a system object due to a lack of protection by the developer.	Attacker gets access to unauthorized objects such as a file, directory or database key.
5	Security Misconfiguration	Incorrect misconfiguration of the server or of the application itself.	Access to server or application.
6	Sensitive Data Exposure	Lack of data encryption in transport of credit cards numbers and tax IDs.	Attackers can steal or modify the data to conduct credit card fraud and identity theft.
7	Missing Function Level Access Control	Access control checks on the server when each function accessed is not done.	Access functionality without proper authorization.
8	Cross-site request forgery (CSRF)	HTTP cookies are primarily used to transmit session tokens. Misuse of the token can allow the attacker to make the victim perform actions of his choice.	Victims change data and perform functions they are authorized to do.
9	Using Components With Known Vulnerabilities	Attackers can easily exploit old third-party components libraries, frameworks, and other software modules because their vulnerabilities have been publicized.	An attack can facilitate serious data loss or server takeover.
10	Unvalidated Redirects and Forwards	Manipulate the URLs of a trusted site to redirect to an unwanted location.	The victim is tricked into navigating to a malicious site-used in phishing attacks.

Table 2.21: Common Weaknesses in Application Software

Source: Adapted from Dukes et al. (2013)

Common Weakness Enumeration (CWE) provides a formal list of common software weaknesses from various sectors (academia, government, commercial). Table 2.22 provides a list of the

common software weaknesses by Common Weakness Enumeration (CWE) (AlBreiki & Mahmoud 2014):

CWE-ID	Name
CWE-78	Improper Neutralization of Special Elements used in an OS Command('OS Command Injection')
CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-Site Scripting')
CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('Cross-Site Scripting')
CWE-798	Use of Hard-coded Credentials
CWE-362	Race Condition
CWE-35	Cross-Site Request Forgery (CSRF)
CWE-285	Improper Access Control
CWE-311	Missing Encryption of Sensitive Data
CWE-209	Information Exposure Through an error message

Table 2.22: Table of Common Weaknesses in Application Software by CWE
Source: Extracted from AlBreiki and Mahmoud (2014)

Analysing Table 2.21 and Table 2.22 reveals the following common vulnerabilities in security, namely: *Injection, Broken Authentication and Session Management, Cross-Site Scripting and Cross-Site* Request Forgery (CSRF). These vulnerabilities provide insight into common weaknesses in applications that can be detected using security scanning software tools.

2.5.1 Security Metrics Tools

In order to integrate security into software development, security metrics tools can be used to review code to identify potential vulnerabilities to ensure secure development. Two of the most popular types of code review security testing are static analysis testing and dynamic analysis testing. Static analysis tools review code in a non-run-time environment early in the development lifecycle. Programs are analysed automatically for weakness that can lead to vulnerabilities without the need to run the code. Static analysis is performed at the source code level, byte code level or binary code level (AlBreiki & Mahmoud 2014). Examples of source code, byte code and binary code analysers are *Yasca, FindBugs and CAT.NET* respectively (AlBreiki & Mahmoud 2014). Binary code scanners work at the binary code level after the code has been compiled which allow for the tool to find vulnerabilities in the compiled and imported binary libraries (AlBreiki & Mahmoud 2014, p96).

Dynamic analysis tests are performed on the finished software product when the application is 'live' and in use. Dynamic testing validates the finished product through a rigorous security evaluation. Acunetix web vulnerability scanner, IBM App Scan and Qualys are an examples of dynamic analysis security testing (DAST) tools. To ensure secure software development it is important to deploy both static and dynamic testing (Pearson 2016). Security metric tools will be employed in this study to measure how secure an application is.

2.6 Chapter summary

A survey of the literature as related to the main research questions and research objectives was presented. Requirements engineering processes were outlined from software engineering as well as ASD literature. Various SRE approaches were presented. The literature review has shown the implications that a security approach and the Agile RE requirements prioritisation process can have on the security of an application. The chapter is concluded by looking at the security of the end product in terms of vulnerabilities. In this regard, application building must incorporate static and dynamic analysis testing. Theoretical Frameworks, Conceptual Model and the Research Design are presented in the next chapter.

CHAPTER THREE: THEORETICAL FRAMEWORKS, CONCEPTUAL MODEL AND RESEARCH DESIGN

Section A: Theoretical Framing and Conceptual Model guiding the Thesis

3.1 Introduction

A theory is a set of related concepts about a phenomenon and is useful for the explanation, systematic view or description of that phenomenon (Walker & Avant 2011). This research study involved diverse concepts. Therefore, the concepts needed to understand the various phenomena under study came from several theoretical frameworks from the existing body of knowledge. In order to create a logical structure of connected concepts to guide the development of various aspects of the research study the following theoretical frameworks have been chosen: *Activity Theory (AT), Soft Systems Methodology (SSM), Design Science Research Model (DSRM)* and *the Technology Acceptance Model (TAM).*

The use of the theories in this research study are two-fold:

a) That which underpins research design to inform research methods and tools (using the theory as a paradigm).
b) That which informs our understanding of the phenomenon under investigation as explained in the results (using the theory as a lens).

In Chapter 4, Design Science Research Methodology was used to support the development of the automated fuzzy tool with the Technology Acceptance Model used to guide the evaluation of the automated fuzzy tool. In Chapter 5, Activity Theory and Soft Systems Methodology were used as paradigms to underpin the quantitative study. In Chapter 6, Activity Theory was used as a lens in the qualitative study.

At this juncture it is important to distinguish between a theoretical framework and conceptual framework. Theoretical frameworks are tried and tested theories that exist in literature. A conceptual framework is obtained by the researcher joining parts (concepts) of different theories to form a cohesive framework to explore a research problem. The researcher designs a conceptual model as it can better represent concepts in the research as compared to an existing theoretical

model (Regoniel 2010). The conceptual framework has more appropriate concepts and constructs that are more relevant to explore the research problem.

A conceptual model was crafted in this study from the existing Activity Theory and Soft Systems Methodology called Soft Activity Model in order to interpret the combined quantitative and qualitative data analysis. The interpretation is presented in Chapter 7. The new conceptual model is further discussed in Section 3.6 where academic credibility is given to the process of its creation. Figure 3.1 depicts how the theories were used in the study as well as their links to the research questions (RQs).

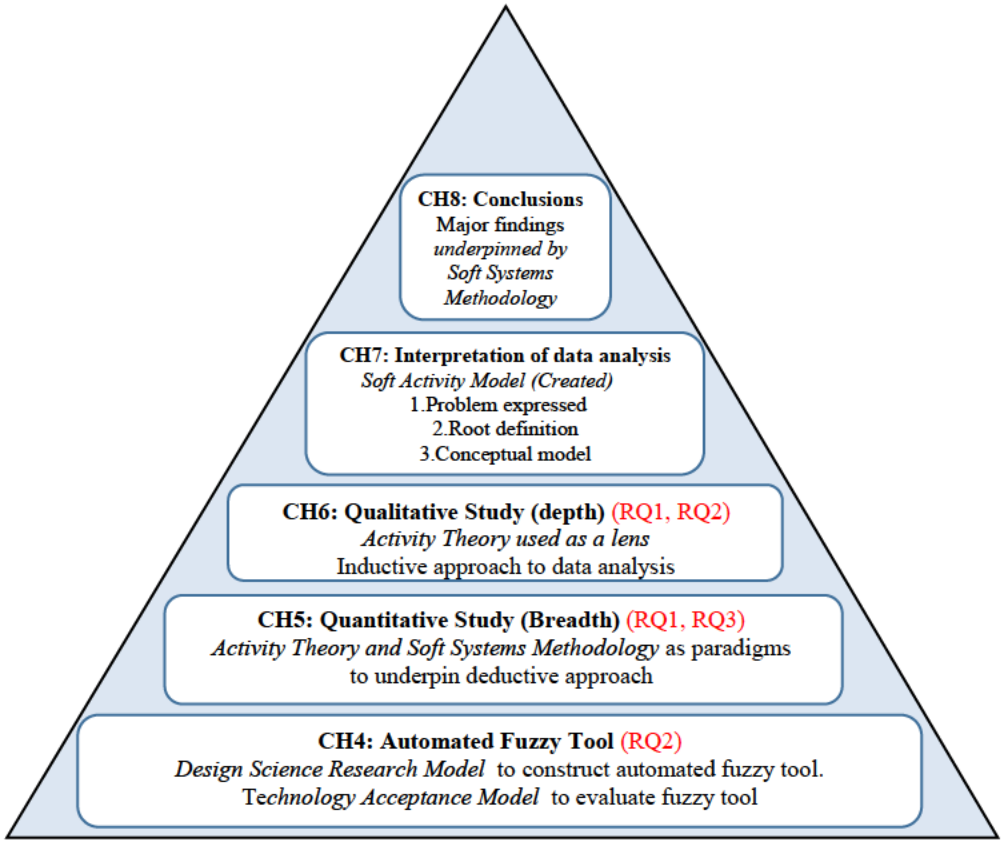


Figure 3.1: Pyramid showing the threading of theories in the study

The chapter begins by providing a theoretical overview of *Activity Theory, Soft Systems Methodology (SSM), Design Science Research Model (DSRM) and Technology Acceptance Model (TAM)*. Key aspects of the theories and its usefulness to the study are discussed. Thereafter the emergent conceptual model is presented. The emergent conceptual model together with the other

subordinate theories will help guide, shape and inform our understanding of the phenomenon under study. In the next section, Activity Theory is discussed.

3.2 Activity Theory

Activity Theory originated in the former Soviet Union by Vygotsky and Leont'ev in the nineteen seventies. Activity theory considers social and cultural factors towards the attainment of higher-level goals and values (Kaptelinin 1995). In Activity Theory, different forms of human practices are studied at individual and social interaction levels to unblock constraints and achieve goals (Nardi 1995).

3.2.1 Key Principles of Activity Theory

The following are the key constructs of Activity Theory (Kuuti 1995):

a) *Activities are units of analysis*

   An individual's actions in an activity is the basic unit of analysis. An individual can and does participate in several activities at the same time.

b) *History and development*

   Activities have a history of their own. Historical analysis is often necessary to comprehend the new situation.

c) *Artifacts and mediation*

   The study of artifacts such as procedures, methods and laws are an essential and inseparable component of human functioning.

3.2.2 How does the theory work

Figure 3.2 presents the framework for Activity Theory.

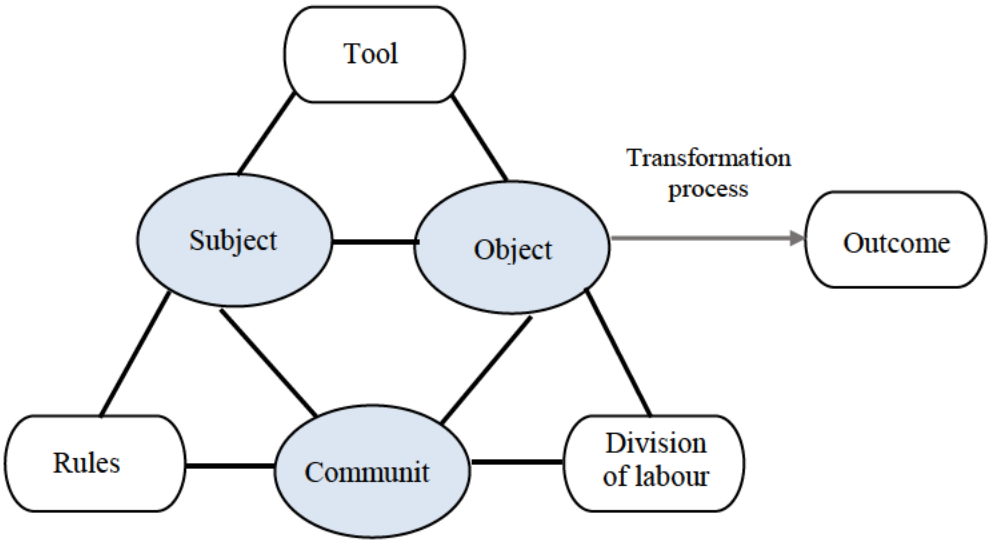


Figure 3.2: Activity Theory Framework
Source: Extracted from Kuuti (1995)

Activity Theory provides an explanation for behaviour of actors in the system based on their interactions. In an activity system the intention of the subject is directed towards an object, for example, requirements engineers (subject) must ensure that critical security requirements are included in the system (object). Figure 3.2 shows the *subject* is mediated by the tool in order to achieve the *object*. The *tool* (e.g. textbook, computer, language) is enabling and limiting as it can empower the subject and can also restrict the subject from achieving the objective. The component, *community* are those in the activity system that share the same object as the subject, for example members of the software development team. Two new relationships are formed with the community component, namely: object community and subject community (Kuuti 1995).

The relationship between *subject and object* is mediated by *tools,* the relationship between *subject* and *community* is mediated by *rules* and the relationship between *object* and *community* is mediated by the *division of labour*. Rules are norms that regulate actions and interaction with the system, for example adhering to the values and principles in the Agile Software Development Manifesto (Murphy & Rodriguez-Manzanares 2008). In activity systems there are systemic

tensions (contradictions), which initiate innovation and change thereby being a source of development (Blin & Munro 2008). There are several activities in an activity system, based on their object (motive), leading to the common outcome (Basharina 2007). However, the network of intertwined activities can also be the source of contradictions.

3.2.3 Contradictions

When the components of an activity system interact, contradictions emerge within and between components. Contradictions are described as constraints, conflicts, problems and disruptions in the activity system. These problems manifest within elements or among the elements, among different activities or different development phases of the same activity. Activity theorists see contradictions or tensions as a source of development (Basharina 2007). Contradictions can only lead to a transformation if they are openly discussed and acknowledged by those experiencing them. An activity system has four levels of contradictions. Primary contradictions are tensions experienced within a single node. Secondary contradictions occur between nodes e.g. between the skills of the *subject* and the *tools* he is using. A tertiary tension takes place between an existing activity and a more advanced activity. Tensions between a central activity and a neighbouring activity is called a quaternary contradictions occur (Uden 2006).

3.2.4 Activity Theory in other studies

Activity Theory has been used extensively in various studies as an analytical tool, providing a high-powered lens, to describe behaviour. For example, Lim and Hang (2003) used AT to explain Information and Communications Technology (ICT) integration in Singapore schools. In this study, ICT was the mediating tool in classrooms to achieve the object of engaging students in different kinds of higher order thinking. Several contractions prevented the object from being achieved, for example tension existed between the division of labour and the community where teaching staff who were more familiar with traditional classroom teaching now had to integrate ICT into their classrooms. Resolving such conflicts led to effective integration of ICT in schools. Similarly, in this research study AT will be used to explain the integration of security or the lack of it in requirements engineering practices in ASD. Tensions identified will go a long way to resolve the problems around security integration in requirements engineering.

In another study, Uden (2006) used AT for designing mobile learning. AT was used to uncover social and cultural tensions in designing a mobile learning environment while Barab et al. (2002) understood tensions in a new technology-rich learning system for an introductory astronomy course using AT. The central tenets of AT were used to analyse students and instructors' interaction with technology leading to the understanding of 3-D models and astronomy. In this study, the researcher used the central tenets of AT to uncover tensions in the activity system towards secure requirements engineering.

3.2.5 Usefulness of Activity Theory in this study with illustrated example

Activity Theory can be appropriately used in the context of human computer interaction (Blin & Munro 2008). Secure software development is about social interactions of stakeholders using software development tools to create secure software. The software development tools are mediating the interaction of humans with the environment (Murphy & Rodriguez-Manzanares 2008). Activity Theory was used as a broad lens of inquiry in this study to explain and interpret the quantitative results. The theory was also used to identify challenges experienced for secure requirements engineering in terms of how the environment impedes security approaches from being included.

Figure 3.3 illustrates an example of an activity system with the goal of secure systems development using the Engëstrom (1987) triangle.

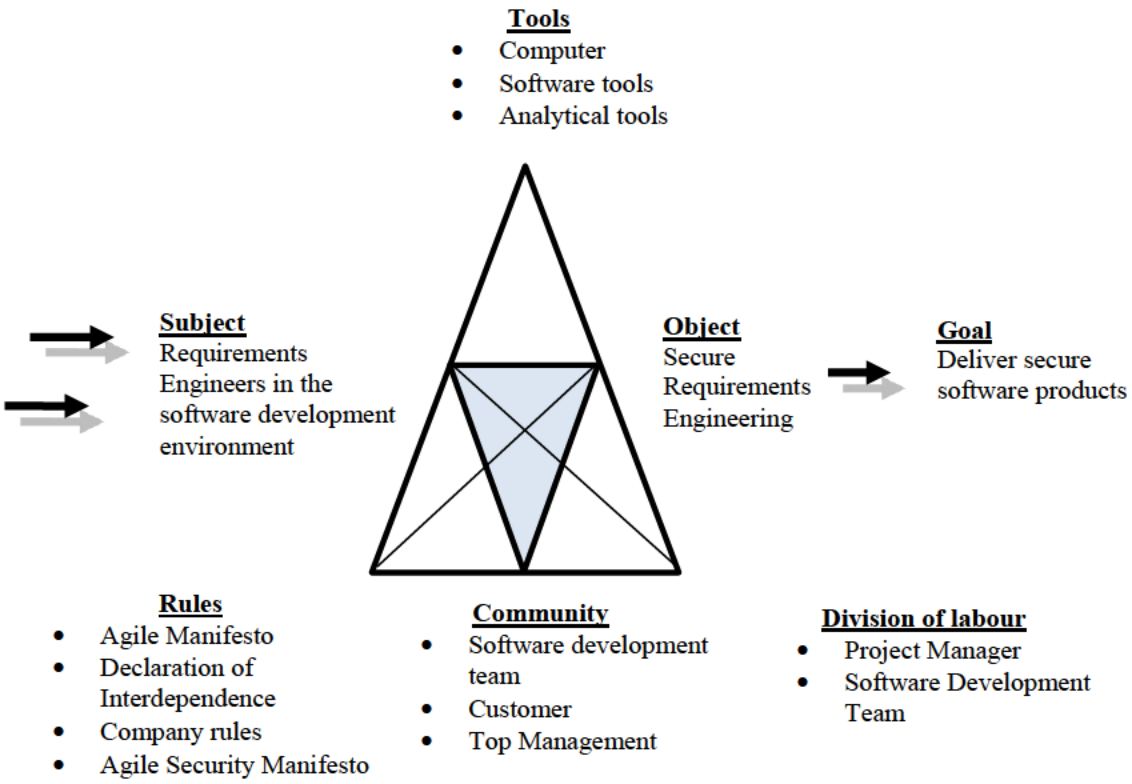


Figure 3.3: Application of Engeström (1987) Triangle to Software Development
Source: Researcher's adaptation

In Figure 3.3 the object is to transform the vulnerable system into a delivered, threat free application. In this activity system, the subjects who are requirements engineers are directed towards the object of ensuring that requirements are secure. The team is the community sharing the object, including the customer or customer representatives. There is a certain division of labour, namely, between the software developers and user representatives, between the team members and between the project manager and his or her subordinates. There is a set of rules covering what it is to be a member of this community as stated explicitly in the Agile Manifesto and Declaration of Interdependence (DOI) (Stankovic et al 2013). There are also implicit rules such as the general working conditions, rules for the programming language and rules on who should speak to the customer.

In order to reach the objective, a number of tools or instruments can be used, such as computers, web applications, programming tools, analysis methods. The collection of these tools has a history with the software development team. In order to understand preferences for tools an historical

analysis is needed. The type of tools utilised is shaped by the context of the activity. To illustrate how activities are intertwined, another activity in the system is for the project manager to ensure that the project is on budget. He has his own tools, division of labour and rules in order to achieve this. When the subject participates in connected activities that have very different objects several tensions can result, for example, the project manager must decide between secure ASD products and a project on budget. AT is used in Chapter 6 of this study to unravel all the tensions that exist for secure requirements engineering to ensure transformation towards secure systems development in a constrained ASD environment.

3.3 Soft Systems Methodology

Soft Systems Methodology helps solve and clarify problems that contain social and political elements (Biggam & Hogarth 2001). Peter Checkland developed an iterative approach to facilitate the clarification of soft problems called Soft Systems Methodology (SSM) that comprises of seven distinct stages (Antunes et al. 2016), namely:

a) "Entering the problem situation
b) Expressing the problem situation
c) Formulating root definitions of relevant systems
d) Building Conceptual Models of Human Activity Systems
e) Comparing the Conceptual Models with the real world
f) Defining changes that are desirable and feasible
g) Devise a step-by-step plan to improve the real world situation"

Figure 3.4 represents the framework for Soft Systems Methodology.

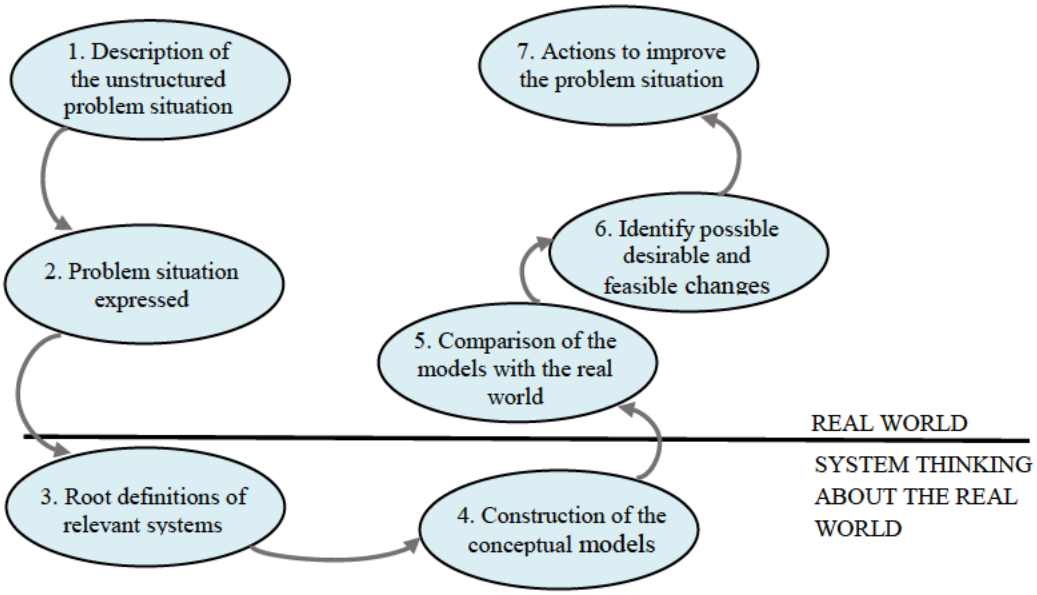


Figure 3.4: Soft Systems Methodology
Source: Extracted from Antunes et al. (2016)

The line separating stages 1, 2, 5, 6 and 7 from stages 3 and 4 indicates that the SSM analysis addresses two main concerns. One is associated with the real world and the other one focused on the systems world in a systemic perspective (Antunes et al. 2016). The SSM approach begins with the identification of a real-world situation that is considered problematic by some stakeholders. In stage 2 the problem is represented as a 'rich' picture and a SSM tool, a CATWOE, is used to structure the problem situation. A CATWOE analysis is a description of the system and is explained in the table below. It leads to the construction of a root definition. In stage 3 a root definition which is a clear definition of the system to be modeled, is produced. In stage 4 conceptual models are produced. Stage 5 compares stage 4 (the conceptual models) with stage 2 (the expressed problem situation). Stage 6 identifies the feasible changes and finally in stage 7 changes to improve the problem situation are actioned (Biggam & Hogarth 2001).

C	*"Client* – the immediate beneficiaries or victims of the system results."
A	*"Actors*- the participants in the transformation, i.e. those who carry out activities within the system"
T	*"Transformation*- the core of the human activity system, in which some inputs are converted into outputs and given to clients. Actors play a role in this transformation process."
W	*"Weltanschauung* (world view) - the perspective or point of view that makes sense of the root definition being developed."
O	*"Owner*-the individual or group responsible for the proposed system. He/she has the power to modify or even stop the system, overlapping to other systems."
E	*"Environmental constraints*- the human activity systems work is constrained by the external environment such as legal, physical or ethical constraints."

Table 3.1: Root definition-CATWOE

Source: Extracted from Antunes et al. (2016)

3.3.1 Soft Systems Methodology usefulness in this study

The increase in e-business has resulted in much more security incidents (Biggam & Hogarth 2001). SSM is used to enhance this problem situation, identify feasible change and action changes to improve the problem. What is the problem situation? Customer confidence in the software developer's capacity to protect customer's data has suddenly become an issue. Many questions have been raised, namely: Are software developers taking security issues seriously? Are top managers providing sufficient budget for security? Are software developers trained to deal with security issues? Are customers being made aware of security issues? Many stakeholders in the software development industry play an important part in counteracting these security breaches. This problem situation can be expressed using a CATWOE to produce a 'rich' picture. It can identify changes that can made to improve security in software development. However, in this study the researcher did not opt for the use of these SSM tools and provides reasons for this in Section 3.6. Stakeholders must implement changes to minimize the security problem even further. These guidelines and recommendations to improve the situation underpinned by SSM are made in Chapter 7 and Chapter 8 of the study.

3.4 Design Science Research Methodology (DSRM)

Design science has been widely used in engineering and computer science and recently researchers have brought design research into the Information Systems (IS) research community (Peffers et al. 2007). An advantage of using this theoretical framework in this research study is that it enables the researcher in the domain of information systems to conduct high quality design science

research that is accepted as rigorous, valuable and publishable in the IS research field (Peffers et al. 2007). Figure 3.5 illustrates the Peffers et al. (2007) Design Science Research Methodology (DSRM) Process Model.

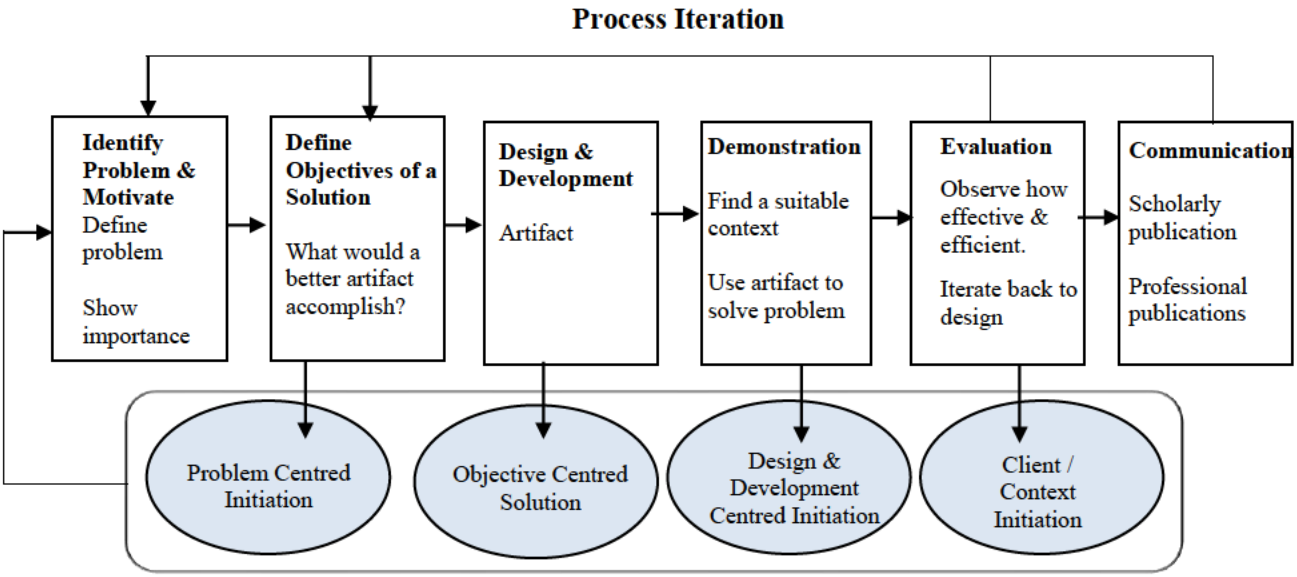


Figure 3.5: Design Science Research Methodology Process Model
Source: Extracted from Peffers et al. (2007)

As illustrated in Figure 3.5, the DSRM process model is divided into the following stages viz. "Identify Problem & Motivate; Define Objectives of a Solution; Design & Development; Demonstration; Evaluation and Communication". This theoretical framework includes three elements namely, conceptual principles to define what is meant by design science, practice rules and a process for carrying out and presenting the research (Peffers et al. 2007).

Peffers et al. (2007) contended that this theoretical model will assist the acceptance of design science research within the IS discipline as it uses analytical techniques to create things to serve humans. It involves a robust process to design artifacts, to solve problems, to make evidence based research contributions, to evaluate the designs, and to communicate the results to appropriate audiences. Such artifacts may include models, constructions, methodologies, and theories (Henver et al. 2004). The DSRM process model is a conceptual model depicting how IS researchers produce and present high quality artifacts that is valuable and publishable in IS research (Peffers et al. 2006).

During the evaluation stage of Design Science Research Methodology, Venable et al. (2012) advises that three important goals must be achieved for a positive evaluation, namely: rigor, efficiency and ethics. Rigor is established when the artifact causes an observed improvement in the problem situation and the artifact works in a real-life situation. The DSR evaluation is applicable within resource restrictions (e.g. money, equipment and time). Persons or the public should not be put at risk during the evaluation.

3.4.1 Usefulness of Design Science in this study

In this study, DSRM is used as a paradigm to underpin the research. This theoretical framework guided the design and development of the automated fuzzy tool to assist requirement engineers to rank clients' requirements. The automated fuzzy tool was produced using the six stages of the DSRM, namely: "*Identify Problem & Motivate; Define Objectives of a Solution; Design & Development; Demonstration; Evaluation and Communication".* The automated Fuzzy tool artifact produced by the design science research is classified according to Venable et al. (2012) as a *socio-technical, product* artifact. Socio-technical artifacts require human interaction to provide their utility. Product artifacts are technologies such as tools, diagrams and software that people use to complete some tasks. DSRM is used in Chapter 4 of the study.

3.5 Technology Acceptance Model (TAM)

Davies (1986) proposed a Technology Acceptance Model (TAM) as a model related to new technology whose factors predict acceptance of new technology (Hussein 2017). When users are presented with new technology, two cognitive beliefs determine if they will use it namely, the perceived usefulness and the perceived ease of use (Legris et al. 2003).

Figure 3.6 represents the Technology Acceptance Model.

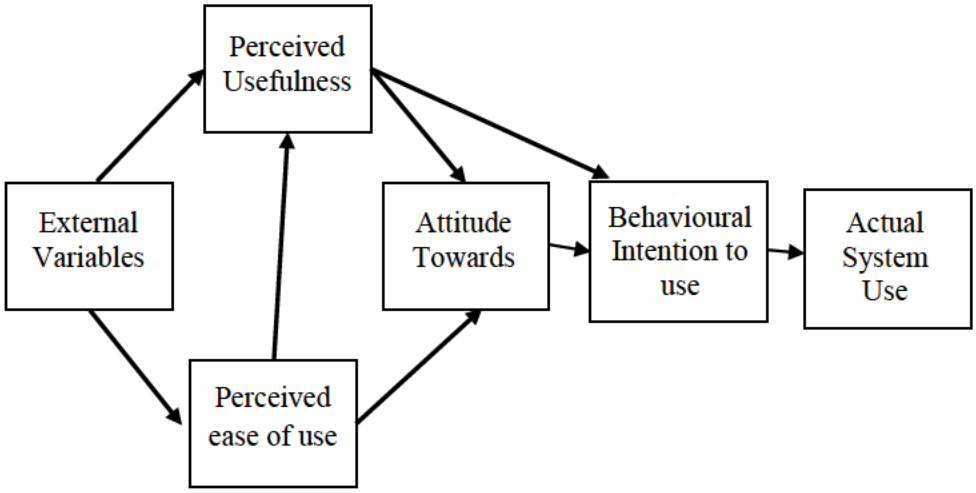


Figure 3.6: Original Technology Acceptance Model
Source: Extracted from Legris et al. (2003)

In the literature, studies have measured the acceptance of technology through the Technology Acceptance Model (TAM). In a recent study, Hussein (2017) investigated students use of e-learning through the technology acceptance model. The findings indicated that attitude was a significant contributor of students' intention to use e-learning. Although the Technology Acceptance Model is useful, one identified weakness is that it must be integrated into a broader theoretical framework which could include human and social interaction (Legris et al. 2003). In this study TAM was integrated into the DSRM framework.

3.5.1 Usefulness of Technology Acceptance Model in this study

The Technology Acceptance Model (TAM) is used primarily as *a paradigm* to underpin the research. TAM is used to measure the acceptance of the automated fuzzy tool. TAM will influence the type of questions asked in the interviews. The construct of 'perceived usefulness' of the automated tool was gauged from the user's interaction based on concepts such as effect on quality control, productivity, effectiveness as well as making the task easier. The construct of 'perceived ease of use' of the automated tool was assessed on concepts such as operability, complexity, flexibility, mental effort, clarity and effort required. TAM is utilised in Chapter 4 of the study within the DSRM framework.

3.6 Emergent Conceptual Model

A model provides an overall framework that we can use to understand reality. The ideas derived from a particular model are called concepts. Concepts are the general expressions of a particular phenomenon e.g. nature of reality, tensions, root definitions, outcomes and others (Walliman 2004). In this study, *Soft Systems Methodology was* combined with *Activity Theory* to create the emergent conceptual framework called *Soft Activity Methodology (SAM)* as depicted in Figure 3.7.

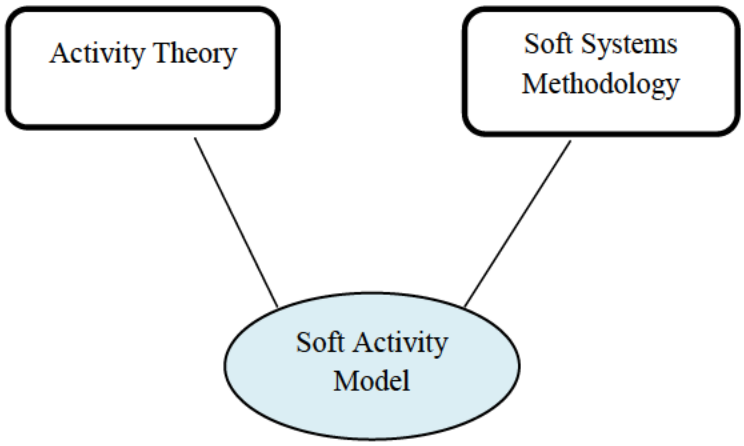


Figure 3.7: Conceptualisation of Soft Activity Model
Source: Researcher's own construction

The Soft Activity Model was specifically designed for the interpretation of results in this study. There was a need for a conceptual model with a wider angled analytical lens suitable to express the diverse problem situations in the research. The researcher needed a more systematic way of finding and presenting specific solutions to these problems. Constructs were also needed to support the creation of activities to achieve these specific solutions. AT, DSRM, SSM and TAM, the chosen theoretical frameworks in this study were not suitable enough to achieve this. Hence, Soft Activity Methodology (SAM), a new conceptual model was constructed, specifically for this study. Hereafter, SAM is referred to as the 'emergent theoretical model'.

3.6.1 Designing and validating the Emergent Conceptual Model

The concept validity of the emergent conceptual model was designed through a mini Delphi process as a way of maximizing consensus about its credibility. In the Delphi method, several rounds of questionnaires are sent to a panel of experts. Their responses after each round are aggregated and sent back to them until consensus is reached (Investopedia 2017).

Three experts were asked to give comments to questions posed by the researcher on the validity of emergent conceptual framework. See Annexure A for Delphi questionnaire. After the first round, one expert raised concerns around the AT components that were omitted. The emergent model was adjusted and sent back to the experts. In the second round of the Delphi questionnaire, the experts reached a general sense of consensus that SAM was valid and could be used in the interpretation of study results.

3.6.2 Emergent Conceptual Model: Soft Activity Methodology (SAM)

The stages for SAM is *express the problem situation*; *write up root definitions* and *construct conceptual models* as shown in Figure 3.8. In the first step i.e. *expressing the problem situation,* AT is integrated into SAM as the lens to magnify the problem situation. The researcher saw AT as a more appropriate analytical lens to explain tensions as opposed to CATWOE analysis and rich pictures of SSM. According to Barab et al. (2002) Activity theory is a more suitable approach for human and computer interaction problems.

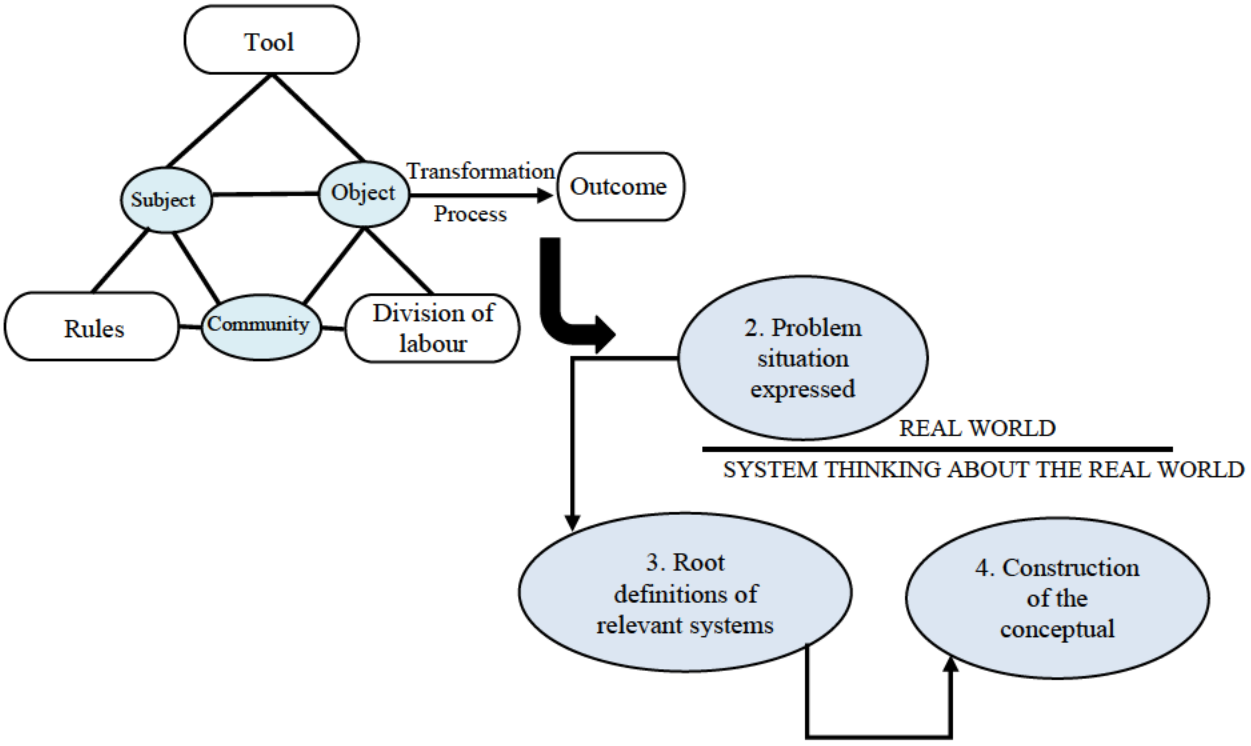


Figure 3.8: Soft Activity Model (SAM)

Source: Researcher's own adaptation

SAM was used *both as a paradigm and a theoretical lens***.** As a paradigm SAM was used to describe the nature of reality (ontology), what we accept as valid evidence of that reality (epistemology) and inform the research methods to investigate that context (methodology). SAM structured data collection as a means of gathering evidence through survey questionnaires and interview schedules on secure requirements engineering practices in ASD.

SAM also structured data analysis using the lens of AT. SAM described the problem situation in a more apt way than SSM. It expressed the problem situation with stakeholders' involvement in requirements engineering processes and the constant tensions that existed between stakeholders. These tensions impeded the subject in the activity system from achieving their objectives and hence outcomes of the system could not be fulfilled. SAM not only provides a lens through which the survey results was explained and interpreted but also provided a framework for finding solutions to minimize the problem situation. SAM is used in Chapter 7 to interpret both quantitative and qualitative results.

Section B: Research Design

3.7 Introduction

The research design consists of a **philosophical worldview, strategies of inquiry and research methods** (Creswell 2009). The philosophical worldview of this study is *Pragmatism* as discussed in Section 3.8; the strategy of inquiry employed is an *Explanatory Sequential Mixed Methods Research* as described in Section 3.9 and the research methods employed in are *Qualitative and Quantitative* as discussed in Section 3.10.

Figure 3.9 illustrates the research design approach for this study.

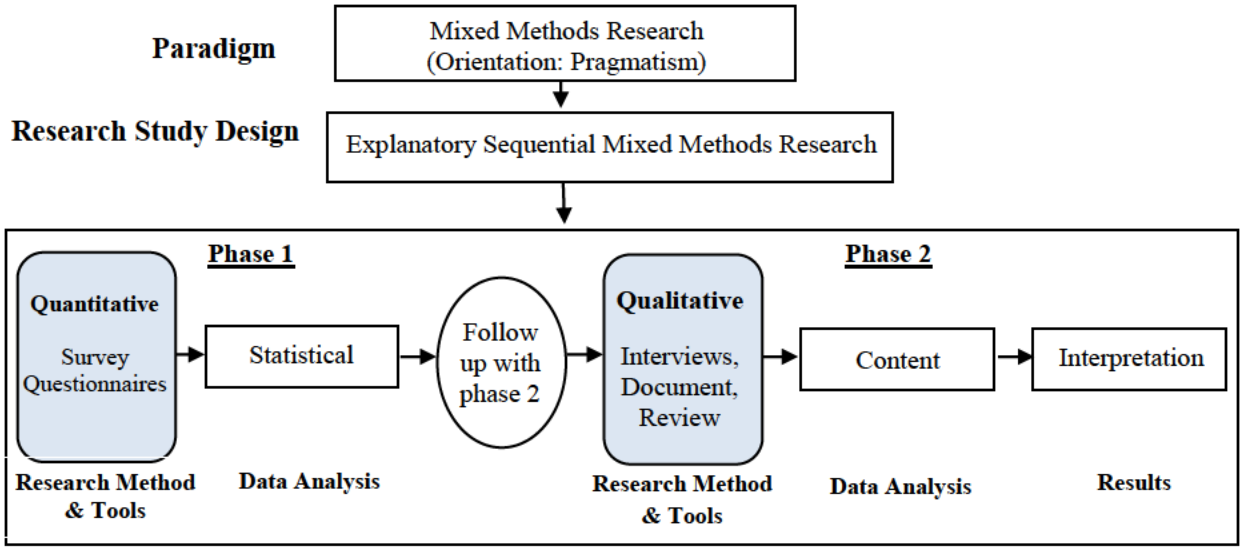


Figure 3.9: Illustration of the Research Design Approach

3.8 Philosophical Worldview

The philosophical orientation of this study is pragmatism. The pragmatists view is that knowledge must be useful and help individuals solve practical problems and the ultimate test of a proposition is whether it 'works' (Vogt 2008). In this research study, security is a practical problem that is affecting the real world. A pragmatic approach was adopted to understand Agile RE practices in terms of security. The findings will be of benefit to a pragmatist as it will make a difference in the real world. The pragmatist researcher uses all methods available to understand the problem (O'Leary 2007). Pragmatism is the philosophical orientation underpinning mixed methods research studies. In support, Creswell (2009) stated that pragmatism opens the door for multiple methods and different forms of data collection and analysis.

3.9 Strategy of Inquiry

The selected strategy of inquiry employed is an *Explanatory Sequential* Mixed Methods Research. Several reasons are advanced as rationale for using Mixed Methods Research (MMR). By combining qualitative and quantitative methods, results are triangulated which enhances the reliability and validity of the study (Mouton 2004). Yin (1999) advised that evidence is stronger if more data collection methods are used in the same study. Teddlie and Tashakkori (2010) pointed out that MMR has commonly been associated with the convergence of results but equally

important is the divergence of results. He stated that when combining information from different sources the researcher can discover how similar or dissimilar they are.

The reasons for using MMR in this study are *triangulation, completeness and explanation*. *Triangulation* refers to mutually corroborated evidence from quantitative and qualitative research to verify and strengthen findings. Agile RE practices in software companies were assessed through qualitative and quantitative data collection and the findings were triangulated. *Completeness* refers to a more comprehensive account of the research area in which both quantitative and qualitative research is employed. A more comprehensive account of how secure Agile RE practices are conducted in the field was established using both research methods. *Explanation* means that one research method is used to help explain findings generated by the other. In phase one of the study, the quantitative data obtained first on Agile RE practices were analysed. Further explanations from the findings of this study were sought by the researcher. In phase 2, the qualitative study was used to explain the findings on Agile RE practices generated in phase 1 (Bergman 2008).

In the broad scheme, this research study was an *Explanatory Sequential* MMR. The research methodology was appropriate as the researcher was able to gauge a general understanding in the first phase after collecting and analysing quantitative data related to RE practices in ASD. Following the quantitative analysis further explanations were required based on the findings. This helped to shape and inform the areas that were the subject for in-depth focus in the qualitative study. Therefore a follow up with collecting and analysing qualitative data in a second phase was required. At the end, after completion of phase 2, data sets were mixed by merging the results during the interpretation of results. This is discussed in Chapter 7. Creswell (2009) cautioned that in a MMR design, the researcher must be mindful that data collection and analysis for both qualitative and quantitative methods are rigorous and time-consuming processes and should consider the amount of time needed for data collection and analysis. The researcher was very mindful of the limitations of this approach during the field work.

3.10 Research Methods

This study employed both the quantitative and qualitative research methods. In a quantitative approach a narrow hypothesis is specified. This hypothesis is tested by collecting numerical data

to support or refute the hypothesis whereas in the qualitative approach the researcher is interested in finding meaning to a phenomenon from the diverse views of participants (Creswell 2009). The data collection tools employed by both methods are described in this section. The quantitative data collection tools used in this study is discussed first.

3.10.1 Quantitative Data Collection Method & Tools (Breadth of Study)

In phase one, the quantitative design was used to find answers to the research questions and assess the hypothesis using numerical data and statistics. Objective data collected was based on facts and were free from opinions and values. The goal in employing this data collection was to avoid bias. The term "bias" here refers to the respondent's subjectivity in answering the questionnaire. Good data is unbiased. In data collection people tend to record as data what makes sense to them and what intrigues them. By employing the quantitative data collection method, selectivity and subjectivity of the respondent was reduced (LeCompte 2000). Furthermore bias was reduced through random sampling.

When employing the quantitative approach, Graziano and Raulin (1997) advised that it was important for the researcher not to get involved with respondents by allowing them space to provide reliable data. Figure 3.10 represents phase one of the data collection representing the quantitative study.

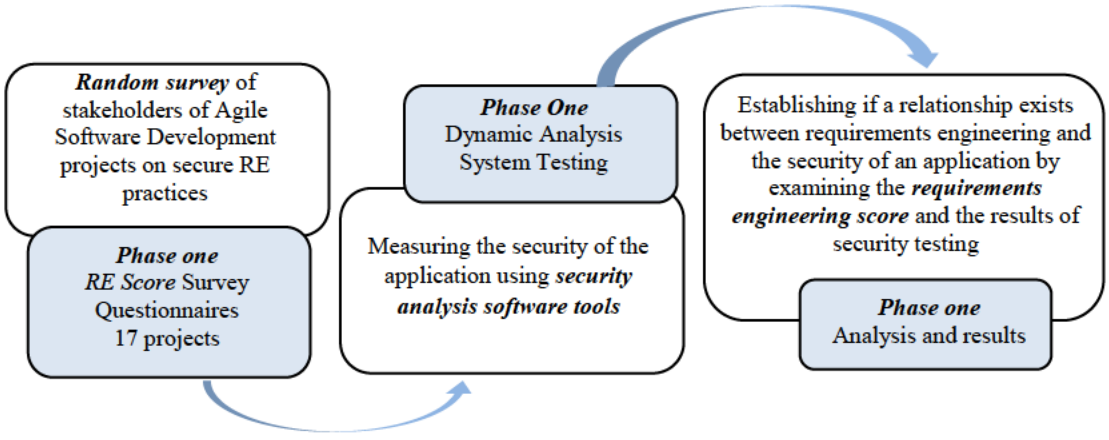


Figure 3.10: Phase 1. Data Collection on Agile RE & Application Security testing
Source: Researcher's own construction

Figure 3.10 shows that in phase one of the data collection a random survey was conducted on stakeholders, to obtain data on RE practices of ASD practitioners and assess the extent to which security approaches were used in 17 software projects. Further, for the same projects, a dynamic analysis security tool was used to measure how secure the actual running application was. Projects for the dynamic analysis test were chosen randomly. The project results were compared to the end product results to establish if a relationship exists with secure RE approaches and the security of the end product. In the next section the quantitative data collection tool, namely the survey questionnaire is discussed.

a) Surveys

A survey was used to collect data by asking questions in a standardised way from the sample population. There are three types of surveys namely, face-to-face interviewer surveys, telephone and internet surveys and self-completion survey questionnaires (Payne & Payne 2004). This study utilised a self-completion survey questionnaire as the survey instrument. In administering the survey questionnaire the researcher was mindful of the limitation that there may be delays in getting results from respondents (Allison et al. 2001). Therefore, data was collected over a two year period.

Several guidelines from literature with regard to the self-completion survey questionnaire were adhered to. Firstly, the design of questionnaires is important as the researcher is not present to help informants who experience problems in answering questions. Secondly, questionnaires must be anonymous as this makes people feel comfortable to answer questions honestly (Kellett 2005). Thirdly, when crafting the questionnaire one must have an understanding of the research questions and knowledge of the research area that the problem resides in (Converse & Presser 1986). Fourthly, the layout of the questionnaire must be pleasing to the eye which makes it easy to get answers as well as analyse responses (Allison et al. 2001). Finally, in order to enhance response rates, questionnaires must be kept as short as possible (Walliman 2004).

Four levels of measurement were used to measure data in the survey questionnaire, namely, nominal, ordinal, interval and ratio levels. Firstly, nominal is very basic and unrefined such as operational definitions e.g. sex (male or female). Secondly, an ordinal scale implies a comparative

entity e.g. unskilled, semi-skilled or skilled. Thirdly, the interval scale has equal units of measurement e.g. temperature in degrees Celsius. Finally, the ratio level has a true zero e.g. distance (Walliman 2004). The Likert scale was utilised in the questionnaire, as it was deemed the most appropriate technique by the researcher to assess this particular problem. The Likert procedure measures attitude on a 5 point scale, namely, 'strongly agree', 'agree', 'not sure', 'disagree' and 'strongly disagree'. The respondents ticked or crossed one of the five positions. Likert scales tends to offer good reliability and a reliability index of 0.85 is often achieved (Oppenheim 2003).

Processes for requirements engineering and security requirements engineering was gathered in the literature review (Pressman and Maxim (2015);Schön et al. (2017);Cao & Ramesh (2010)). The Likert scale questions were constructed from mapping requirements engineering processes from the existing body of knowledge. The questionnaire was developed to measure the extent that secure requirements engineering approaches were implemented in Agile RE practices in Industry as well as to establish how software engineers manage client security requirements. A Pilot survey conducted allowed the researcher to modify the Likert scale questionnaire when necessary.

Once the construction of survey questionnaire was completed, a pilot survey was conducted with experienced software engineers before administering the actual survey. Piloting is the term used for trying out the survey on typical respondents. The pilot survey assisted to identify problems and sources of confusion such as ambiguities with the questionnaire. It was also important to ascertain if the proposed times frames for answering the questionnaire were correct.

Overall the higher the response rate the lower the bias (Ornstein 2013). A sample survey questionnaire for this study is attached in the addendum. An appropriate cover letter was sent with the survey questionnaires explaining the purpose of the research, contact information, return information and the need for a quick response.

3.10.2 Quantitative data analysis

Detailed data analysis was completed using specialized software packages namely, Statistical Package for Social Science (SPSS). Descriptive statistics and inferential statistics were obtained.

3.10.2.1 Descriptive statistics

Descriptive statistics involve the tabulation and organization of data in order to demonstrate their main characteristics (Cramer & Howit 2004). Madan (2014) describes basic descriptive statistics as mean, median, standard deviation and variance. In this study, the mean was used to calculate the RE score given to a project by the ASD team. The median was required in the calculation of variance. Variance is used in the calculation of standard deviation. Standard deviation gives an indication of the spread of the data and knowing what is normal, for example data within standard deviation can be considered normal. This information was important to understand the mean RE score per project. A high standard deviation meant the scoring is more towards one extreme and a low standard deviation meant scoring is normal. When the standard deviation was high for the RE mean score the researcher was able to ask several why questions in the qualitative study.

Descriptive statistics in this study was used for the frequency distribution and cross tabulations. The frequency distribution for every question was calculated. Cross tabulation such as the "role" of the person crossed with "I have attended security training" brought about useful insights into the data.

3.10.2.2 Inferential statistics

Inferential statistics allows one to draw conclusions or inferences from data (Vogt 2008). The researcher examined the data to establish statistical significance. The following tests were conducted: internal reliability tests, Chi-squared test, Pearson's correlation coefficient (r) and t-tests.

(i) Internal reliability tests

The internal reliability of the survey questionnaire was tested using Cronbach's alpha. The internal consistency of items in the survey questionnaire were measured to ensure that items correlated with one another and that they all measured the same construct (Lavrakas 2008). Cronbach's alpha values ranges from between 0 and 1. In a research instrument when the alpha value obtained is 0.7 and above, this is considered reliable. When the alpha value is 0.8, this is considered moderate

reliability and when the alpha value is 0.9 and closer to 1, this is considered high reliability (Peng 2009).

(ii) Correlation Analysis

Correlation is used to demonstrate whether two variables are correlated or related to each other. The measure of the association between two numerical variables, *x* and *y* is called the correlation coefficient. The relationship is considered symmetrical if *x* is correlated with *y* and if *y* is correlated with *x* (Crow 2006).

Chi-squared Test

Several tests known as Pearson chi-square denoted as chi^2, x^2 and C^2 were conducted to test if a relationship was significant on categorical data. Chi^2 must be less than 0.05 for a significant relationship (Crow 2006). In this study various correlations were made using a two way table. Variables in the rows were crossed with variables in the columns.

Pearson's correlation coefficient (r)

Pearson's correlation coefficient was used in the study to test for correlation between two variables x and y. The value of the correlation coefficient (r) lies between +1 and -1. A positive coefficient indicates that a high value of *x* tends to be associated with a high value of *y* and a negative coefficient indicates that as the value of *x* increases the value of *y* is likely to decrease. A coefficient of 0 means that there is no relationship between the two variables (Peng 2009). The closer to +1 or −1, the stronger is the relationship (Vogt 2008).

3.10.2.3 Factor analysis

Factor analysis is a statistical technique that is used to identify the small number of variables (factors) that constitute a broad construct. The main aim of factor analysis is to identify the latent variables (variables that are not directly measured) on the basis of the observable variables. The latent variables can have an impact on the study results and are therefore considered important. These latent variables are referred to as the underlying dimensions of a construct (Woolford 2015). One application of factor analysis is data reduction. This is achieved by combining the underlying dimensions (factors) into summary indices, namely, sub-themes. Another use of factor analysis is

that it is a confirmatory method for assessing construct validity If the underlying dimensions are inconsistent with expectations then construct validity is compromised. (Floyd & Widaman 1995).

In this research study factor analysis was used to analyse the structure of a construct in the survey instrument to assess construct validity. The researcher wanted to test if the questions in the questionnaire adequately reflected the structure of the constructs in requirements engineering. Factor analysis was also used in an exploratory manner to identify external variables that relate to the various dimensions of Agile RE. Further to this factor analysis helped to examine interrelationships between items measuring Agile RE practices (Pett et al. 2003).

Tests required before the factor analysis method can be conducted

Before conducting the factor analysis procedure, two tests are necessary, namely:

- The first requirement is that Kaiser-Meyer-Olkin (KMO) measure of sampling adequacy should be greater than 0.50;
- Secondly Bartlett's Test of Sphericity is statistically significant (less than 0.05).

In all instances, both these conditions must be satisfied; which allows for the factor analysis procedure (Woolford 2015).

3.10.2.4 Regression model

In statistics regression analysis is used to measure the impact of one or more independent variables on single dependent variables. The independent variables are referred to as predictors as they predict the dependent variable. Independent variables will not perfectly predict the dependent variable therefore there will always be an error term in the regression equation. The regression equation is a linear function and best fits the data. The regression line summarises the relationship between dependent variables and independent variables (Harvard Business School 2017).

Thus in order to understand the impact of independent variables on Agile RE practices a regression model was constructed. The model examined which independent variables had the most impact on the dependent variable and was used to infer causal relationships. In order to conduct secure requirements engineering in Agile Software Development it was important for the researcher to mathematically determine which factors impact the process the most and which factors impact the

least. The regression model for secure requirements engineering practices is presented in detail in Chapter 5.

3.10.2.5 Structural Equation Modelling (SEM)

SEM is also called a causal model. SEM was used to test the theoretical model by analyzing the structural associations amongst a set of variables to determine if a relationship exists. Independent variables are referred to as predictor variables or causal variables. Dependent variables are influenced by the variations in independent variables. The independent variables cause the outcome of the dependent variable (Franzosi, 2004).

The observed (measured) variables are directly measured from the data obtained in the study are called exogenous (independent) variables. The dependent variable is referred to as the endogenous variable. Latent variables are not directly measured but are indirectly measured by the observed variables. Path diagrams play a fundamental role in structural modeling. Path diagrams are like flowcharts that show variables that are interconnected. Lines with arrowheads are used to indicate causal flow (Franzosi, 2004).

A simple structural equation model is given by the following equation:

$$Y = \alpha X_1 + \mu X_2 + \beta X_3 + \gamma X_4 + \delta X_5 + \ldots. \lambda X_n + e$$

Where: Y is dependent variable

X_1 …..X_n are the independent variables

$\alpha, \mu, \beta, \gamma, \delta, \lambda$ are the coefficient values

e is error term

The model shows that the dependent variable is predicted by the independent variable. The structural equation model was created using a computer application called Listrel.

3.10.2.6 Dynamic Analysis Testing: Acunetix Web Vulnerability Scanner

Security testing in application development is becoming a mandatory practice of software development. Two tests are used to test for security breaches in a web application. While static analysis testing is used to test the application in a non-runtime environment, dynamic analysis system testing, tests the web application when it is in operation. DAST was used in this research

study to test for vulnerabilities of the live web application. The live systems were scanned and a vulnerability report was produced by the web security scanner.

Acunetix, a Malta based provider of Dynamic Analysis System Testing (DAST) tested the vulnerability of the ASD product in this study. This security scanning application performed DAST on the finished product. It simulated attacks in the same way a malicious user would use the system and analysed the web applications reaction. Acunetix was rated the top five security scanners for web application security testing (Gartner 2016). The results from the DAST reports are presented in Chapter 5.

3.10.3 Reliability and validity

"Reliability refers to the consistency in the results of the measurement, while validity measures whether the questionnaire is measuring what it claims to be measuring" (Brinkman 2009). To test reliability of the questionnaire the *test-retest reliability* was used. The validity measures for the questionnaire in this study were *face validity*, whether the questionnaires looked valid; *content validity*, whether the questionnaires captured the full content of the construct; *criterion validity*, whether the results of the questionnaires correlates with other valid sources; *construct validity*, whether the questionnaires measured the unobservable, theoretical construct (Brinkman 2009).

3.10.4 Qualitative data collection method and tools (Depth of study)

Qualitative research involved collecting more detailed information from a small number of people. The underpinning principles of qualitative research are naturalism, a holistic approach and seeing through the eyes of others (Harding 2013). The researcher attempted to obtain the inside view of the phenomenon by getting as close as possible to the subject of the research (Walliman 2004). Data analysed consisted of words recorded on audio or transcribed, researcher's notes and documents or other pre-existing items (Rapley 2004).

Figure 3.11 represents phase 2 of the data collection process. Following phase one of data collection, detailed explanations and clarity from the survey analysis was sought using the structured interview as a data collection tool in phase two. For example, the survey questionnaire

helped identify requirements engineering activities used by ASD practitioners but the interview assisted to uncover the nature of these activities, best practices as well as challenges experienced.

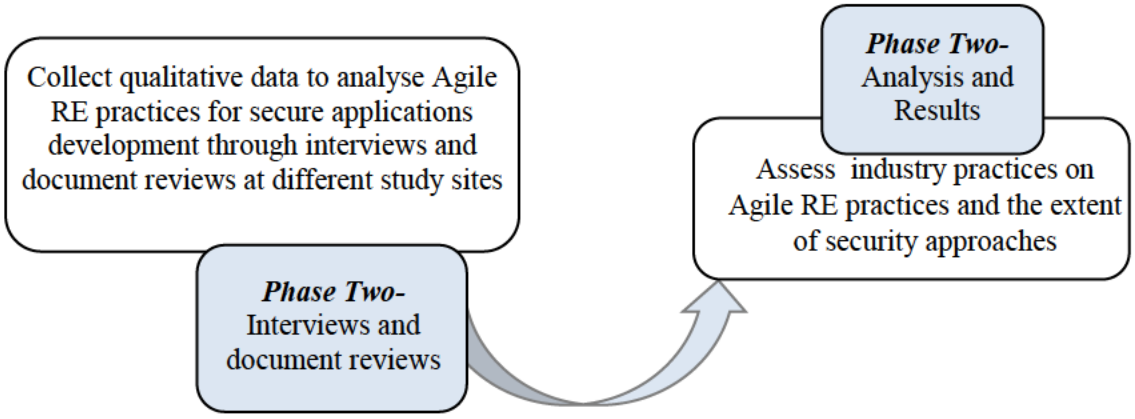


Figure 3.11: **Phase 2. Data Collection from Interviews on RE practices**
Source: Researcher's own construction

Figure 3.12 represents phase three of the data collection process. In phase three, structured interviews were conducted on key stakeholders to measure the usability and acceptance of the automated fuzzy tool.

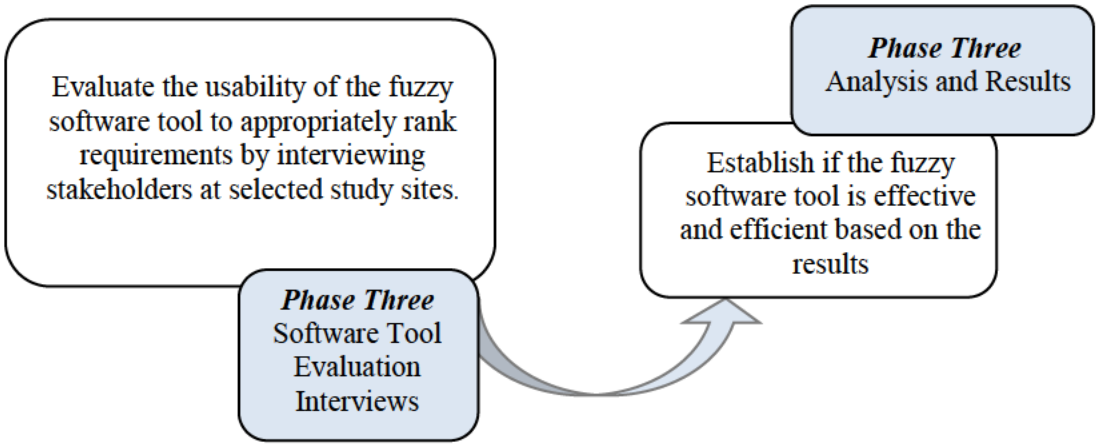


Figure 3.12: Phase 3. Data Collection on Usability of Fuzzy Software Tool for requirements prioritisation
Source: Researcher's own construction

3.10.5 Interviews

Interviews allowed for interaction between the researcher and informant where more probing questions were asked. According to Silverman (1998), an interview helps us to generate data which gives authentic insight into people's experiences. Interviews can be used to seek 'deep' information that cannot be obtained by a survey (Gubrium & Holstein 2002). The interviews allowed for social cues for example voice and gestures. The researchers explored in detail the experiences, motives, and opinions of others and saw the situation from the perspectives others. The interviewer used probes in the questioning technique. Probes are questions, comments, or gestures used by the interviewer to help manage the conversation. It allowed the interviewee to complete an idea or fill in missing information (Rubin & Rubin 2012). Harding (2013) suggested that an interviewer must encourage respondents to ramble on, which can demonstrate what is important to them. Further, new and unplanned questions were asked as a result of something the respondent had said.

The seven stages of an interview inquiry identified by Kvale (2007) namely, thematising, designing, interviewing, transcribing, analyzing, verifying and reporting were utilised. Interviews in this study were audio recorded. The researcher ensured that the recording was audible for the transcriber and sufficient time was allocated for transcribing. To verify that the transcription was reliable two persons independently transcribed the same recording and then did a word count on the number of words that differ between the two transcriptions. Drawbacks of in-depth interviews that the researcher was mindful of, is that it has a relatively long duration and it is an extremely obtrusive data collection method (Gubrium & Holstein 2002).

3.10.6 Qualitative data analysis

Qualitative data was analysed using Content Analysis. Content Analysis analyses the content of the text or document to refer to words, themes, meanings, pictures or patterns (Mouton 2004). Thematic analysis is a common approach to content analysis, in which the dominant themes are captured based on categories (Franzosi 2004). Written narratives that summarise what the researcher found were another source of analysis. Transcripts were produced from note taking and audio recordings of interviews. Audio recordings allowed the interviewer to interact with the interviewee, reduced time lost writing and finally, audio recordings captured a much more detailed

record of the conversation than note taking or reflection. Rapley (2004) advised that recordings may be replayed to selectively search for patterns to provide evidence of an argument.

Reliability or rigor in data analysis was enhanced by asking an external person to redo the analysis. Notes were compared with the external source to validate the data analysis. More than one type of qualitative data namely, observation, interviews and document analysis were used in the analysis. The qualitative analysis was supplemented with data from the quantitative research source. A record (audit) of how data categories were established and themes identified was stored. As data accumulated it was important to organize the shapeless mass of copious data using a coding system.

Codes and coding

Codes are used to give collected data meaning (Walliman 2004). Coding was a way of indexing or categorizing the text in order to establish a framework of thematic ideas about it. It was easier to code by reading a transcript and deciding what it is about than directly from the audio recording. Codes formed a focus for thinking about the text and its interpretation. Codes used were descriptive codes such as 'elicitation techniques', categorical codes such as 'security information sources' and analytical codes such as 'Core RE activities' was used for this research project (Gibbs 2013). A computer program, Nvivo Pro was used for filing and retrieving coded information.

3.11 Population and Sampling

Figure 3.13 represents the population and sampling techniques employed in this study.

<table>
<tr><th colspan="10">Random Sampling of 17 Agile Software Development (ASD) Companies</th></tr>
<tr><th colspan="10">Random selection of a minimum one ASD Project from each Company</th></tr>
<tr><td colspan="2">Project 1
±7 stakeholders</td><td colspan="2">Project 2
±7 stakeholders</td><td colspan="2">Project 3
±7 stakeholders</td><td colspan="2">Project n
±7 stakeholders</td><td colspan="2">Project 17
±7 stakeholders</td></tr>
<tr><td colspan="10">  </td></tr>
<tr><td colspan="2">Sampling Units</td><td colspan="2">Sampling Units</td><td colspan="2">Sampling Units</td><td colspan="2">Sampling Units</td><td colspan="2">Sampling Units</td></tr>
<tr><td colspan="2">ASD Team</td><td colspan="2">ASD Team</td><td colspan="2">ASD Team</td><td colspan="2">ASD Team</td><td colspan="2">ASD Team</td></tr>
<tr><td>Survey
Random</td><td>Interviews
Purposive</td><td>Survey
Random</td><td>Interviews
Purposive</td><td>Survey
Random</td><td>Interviews
Purposive</td><td>Survey
Random</td><td>Interviews
Purposive</td><td>Survey
Random</td><td>Interviews
Purposive</td></tr>
</table>

Figure 3.13: Population and Sampling

Source: Researcher's own construction

The population is referred to as the total number of cases that are subject to the study (Walliman 2004). The population size comprised of 300 Agile Software Developers. The sampling frame comprised of 7 team members from each of 17 companies. Simple random sampling was used in the choice of study sites. A sample is a subset of projects drawn from the target population in a cross sectional design (Graziano & Raulin 1997). A single ASD project was the sampling unit or unit of analysis at each study site. A maximum of one project is chosen from each study site. A random sampling scheme was employed for the selection of projects/teams at study sites from the sampling frame to prevent bias.

Software development teams of the selected projects were invited to be part of the data collection for the study. Data was collected from sampling units by studying project documents, team members completing survey questionnaires and interviews conducted on team members.

3.12 Ethical consideration

3.12.1 Recruitment of participants

Agile Software Development companies were approached to participate in this study. The companies were obtained from the list of Agile Alliance Companies; Companies that are members of the Durban University of Technology (DUT) IT Advisory Board as well as ASD companies obtained by networking with past DUT students in employment.

The companies ranged from small, medium and large software development enterprises. These companies ranged from single teams to multiple teams handling software development projects. Larger companies specialised in software products such as Gaming and Medical products while the small to medium companies had no specific focus area and took on projects adhoc. The composition of the companies in terms of their domains are as follows: Gaming(5), Medical(3); Transport(1); Finance(2); Education (2) and Adhoc(4). These companies represented the gatekeepers for the study and access to data was sought from senior management. Larger companies were the gatekeepers and permission was required from these companies as per the University of KwaZulu Natal's ethical clearance requirements.

Recruitment of participants took place in the natural habitat of the participants namely the study site (Company). The study site was equipped with comfortable boardrooms with ergonomic chairs and sufficient ventilation. Adequate safety and toilet facilities were available. Participants felt comfortable to participate in this environment.

Once a company agreed to participate in the study a meeting with all project managers was held in advance. They were briefed about the study and the data collection approaches used in the study. The researcher found out in advance if participants were prohibited from receiving incentives. Permission was obtained for use of the venue. Convenient times for data collection were negotiated with project managers. Project managers then made team leaders and seniors aware of the study. The team leaders disseminated the information to the software development team members.

Following this meeting, an e-mail with a brief description of the study was sent out to all software development teams (eligible participants) in the company shortly before data collection commencement. The times required for the interview (one hour) and for the survey questionnaire (15 min) were also stated on the e-mail. The e-mail advised that participation was voluntary. Information regarding incentives provided for participation was also discussed. A small gift such as a memory stick to the value of R200-00 was provided to interview participants. A stationery set consisting of a pen, pencil and ruler were provided to survey participants. The incentive encouraged participants to volunteer for the study and minimized 'no-shows'.

Before the data collection session, signed consent forms were obtained from participants. The rights of all respondents were respected by treating them equally with dignity and respect. The freedom of choice of the individual was respected, for example they were allowed to discontinue with the interviews at any time, participants were free to express concerns at any time and they were given access to any information that they required. Participants were provided with repeated assurance of anonymity.

3.12.2 Informed consent

A letter informing participants of the possible risks and benefits of participating in the research project and that participation was voluntary was sent in advance. Informed consent was thereafter obtained from all participants of this research study.

3.12.3 Anonymity and confidentiality

Privacy and confidentiality was maintained throughout this research study. Participants remained anonymous and no names or personal details were divulged. Data returned to the researcher was free from personal information to ensure confidentiality.

Ethical considerations were adhered to for all interviewees. The researcher ensured that participants were confident that the anonymity was maintained. Anonymity of transcription was adhered to by concealing details that could identify respondents and settings.

3.12.4 Protection of study participants

The right of participants to withdraw at any stage without providing reasons was specified on the participant information sheet. This was addressed in the presentation prior to obtaining informed consent from potential participants. Further, participants were assured that the returned participant data would be kept safe in a locked cupboard for access only by the researcher. All data entered onto the computer would be protected by password. It was critical that the behaviour of researcher encompassed honesty, sensitivity and trust.

3.13 Chapter summary

This chapter was divided into two sections. Section A discussed the theoretical framing for the study while Section B discussed the research design. In Section A, a number of theories chosen for this study were presented, namely: Activity Theory, Design Science Research Methodology, Soft Systems Methodology and the Technology Acceptance Model. These theories were introduced by providing some history on each of them. Important concepts in each theory were outlined. The researcher also discussed the usefulness of each theory to this study. The use of the theory in other studies was also discussed. The researcher created a new model called Soft Activity Methodology (SAM) to discuss and interpret the results of the study. The use of SAM in this research study was discussed.

In Section B, the research design was discussed. This study is a mixed methods research study. The study was broken up into three data collection phases. In phase 1, data was collected for the quantitative study. Further explanations sought from the findings of phase 1 were clarified in phase 2, the qualitative study. The primary data collection instrument in the quantitative study was the survey questionnaire whilst interviews in the qualitative study were the main source of data. Data

analysis for the quantitative study was statistical analysis whilst data analysis for the qualitative study was content analysis. Phase 3 of the data collection entailed the evaluation of the automated fuzzy tool with stakeholders of the software development team. The chapter was concluded by discussing the ethical considerations for the study. In the next chapter the automated fuzzy tool is presented.

CHAPTER FOUR: PRESENTATION OF AUTOMATED FUZZY TOOL

4.1 Introduction

"Mathematics is the supreme judge; from its decisions there is no appeal."–Tobias Dantzig

The automated fuzzy tool was developed and is presented using Design Science Research Methodology (DSRM) theoretical framework. DSRM was utilised primarily as a process model to guide the development of the automated fuzzy software tool and assess usability evaluation by software developers. A brief report on the software construction of the automated fuzzy tool for requirements prioritisation is presented below through the lens of each DSRM stage namely. "Identify Problem & Motivate; Define Objectives of a Solution; Design & Development; Demonstration; Evaluation and Communication".

4.2 Stages of artifact development through the lens of Design Science Research Methodology

4.2.1 Stage 1: Identify problem and motivate

Requirements prioritisation is essential in any Agile Software Development approach. Unlike traditional software development, requirements prioritisation occurs continuously in ASD. Prioritisation takes place before every sprint. At the end of a sprint there are requirements that are not implemented for a number of reasons such as time constraints. These requirements are sent back to the project backlog for reprioritisation or are dropped from the project. At the start of an iteration the customer is allowed to add new requirements. These new requirements are brought into the product backlog. The nature of Agile Software Development also allows the customer to change requirements. Therefore, at the start of every sprint the product backlog must be reprioritised. Therefore, prioritisation of requirements becomes a core task at the start of every iteration. It is therefore imperative to use a proper requirements selection technique for ranking requirements.

When the wrong requirements are identified for implementation it can lead to project failure because the success of developing a feature is dependent on business value, project constraints, personnel availability and various other factors (AL-Ta'ani & Razali 2013). Clearly there are

multiple evaluation criteria in the selection of requirements for implementation therefore requirements selection done ad hoc can easily lead to an incorrect sprint backlog for that iteration. This can result in negative outcomes for that project. Hence a more systematic approach is required.

Furthermore, various stakeholders must be involved in requirements prioritisation. The customer has a strong voice in the prioritisation of requirements as they need to generate the most business value from the software application for their business (Tuunanen & Kuo 2015). The presence of an important feature at the most opportune time can assist in increasing the profits of the company. Requirements prioritisation becomes a multiple decision maker problem. It is imperative that the selection technique ensures that all stakeholders especially the customer contribute to the decision making process (Achimugu et al. 2014). A fast and efficient tool is required to ensure that all stakeholders, especially the customer, participate.

An added benefit of using a proper prioritisation technique is that it can also guide stakeholders about the level of importance attributed to each evaluation criterion. Stakeholders will have a more precise method of knowing what criterion is most important and what criterion is least important in that project. Ordinary prioritisation selection techniques focus on the requirements and tend to ignore the criteria (Achimugu et al. 2014). There are many iterations in an Agile Software Development project and stakeholders may place importance on different criteria at each iteration. This means that if a project has four iterations then it is possible that four different criteria could be deemed most important at each iteration, based on available resources. At the end of the prioritisation process it is important to know what criteria was most important and what criteria was least important for that iteration. A requirement prioritisation technique in Agile Software Development must steer away from a ‘one size fits all’ policy.

An automated tool to rank security requirements is not evident in the literature. Another important application of a prioritisation selection tool during requirements engineering is determining which security requirements must be selected for the product backlog. Security requirements are identified based on business and system assets (Salini & Kanmani 2011). These requirements must also be prioritised. An automated system for assessing risk when threats and vulnerabilities occur

is needed. The prioritisation of security threats is based on many factors such as the level of security required and the assets to be secured. Stakeholders can rank security requirements in the threats and vulnerability table using multiple evaluation criteria. Once the list of security requirements is prioritized, not all security requirements can be implemented due to budget and project constraints. Only the high priority security requirements can be selected and promoted to the project backlog. An appropriate tool is needed to assist with security requirements identification for implementation.

Based on the above discussion two important points must be highlighted. Firstly, requirements prioritisation is imperative in Agile Software Development. Secondly, the selection of requirements is based on multiple decision makers and multiple evaluation criteria. After conducting an extensive desktop literature review of requirements prioritisation research, a new approach using the mathematical fuzzy TOPSIS algorithm as a basis is presented. The algorithm is coded programmatically in an object-oriented programming language. The automated fuzzy tool is an online web application that provides a strong mathematical and scientific basis for the ranking of requirements. It is aligned specially for use in Agile Software Development and ensures that the selection of requirements meets the satisfaction of project goals and more importantly the customers' expectations.

4.2.2 Stage 2: Define objectives of a solution

The following objectives were defined:

- Investigate current practice of requirements prioritisation and design an automated fuzzy tool that can promote best practice in Agile Software Development.
- Investigate how the automated fuzzy tool can be merged with conventional practices to prioritize the product/sprint backlogs.
- Construct an automated fuzzy tool artifact for improving Multi-Criteria Decision Making (MCDM) when ranking requirements using the fuzzy TOPSIS method as a basis for the development.
- Use the automated fuzzy tool in other areas of software development such as ranking of sprint backlogs, identification of security requirements for the project and the ranking of evaluation criteria.

4.2.3 Stage 3: Design & development

An object oriented paradigm (OOP) creates software applications using precise data. However, information in the real world is often imprecise and vague. In the design and development of the automated fuzzy tool, the fundamental issue was how to model and implement fuzzy data using the crisp structures of an object oriented paradigm. This meant that the conventional crisp object oriented approach structures needed to be extended to accommodate fuzzy data. The focus of this section is to show how conventional object oriented and extended fuzzy object oriented constructs were used in the design and development of the automated fuzzy software tool.

The approach used in presenting this section is to first discuss fundamental design issues in Section 4.2.3.1 and then discuss the development in Section 4.2.3.2.

4.2.3.1 The Design: Modeling Conventional and Fuzzy Data in an OOP

a) Design pattern

A singleton design pattern was chosen for the web application. The singleton pattern ensures that a class has only one object and provides a global point of access to this object. The singleton pattern was useful for this application because it allowed system wide actions to be co-ordinated from a central place. The single object could be used by all other classes. This design pattern also ensured that memory is saved because only one instance can be created.

The fuzzy TOPSIS algorithm was broken up into seven steps as explained Section 2.4.2. A reusable class was created for each step of the algorithm. In this application the DataWarehouse class is implemented as a singleton and has reference to all classes as shown below.

```
public class DataWarehouse
{
    private static DataWarehouse instance = null;
    private Utils _utils;
    private Step1 _step1;
    private Step2 _step2;
    private Step3 _step3;
    private Step4 _step4;
    private Step5 _step5;
```

```
    private Step6 _step6;
    private Step7 _step7;

    private DataWarehouse()
    {
        Utils = new Utils();
        Step1 = new Step1();
        Step2 = new Step2();
        Step3 = new Step3();
        Step4 = new Step4();
        Step5 = new Step5();
        Step6 = new Step6();
        Step7 = new Step7();
    }

    public static DataWarehouse Instance
    {
        get
        {
            if (instance == null)
            {
                instance = new DataWarehouse();
            }
            return instance;
        }
    }
}
```

b) UML extended to Fuzzy UML

UML provides software developers with design tools for the construction of software applications. One UML tool, the class diagram is a key artifact in object oriented programming as it constitutes the backbone of the object oriented development and provides a solid foundation for design and implementation of the software product. Moreover the UML class diagram allows developers to

understand the system and provides a concrete mapping to source code (Kagdi et al. 2005). The class diagram depicts the classes and the inter-relation between them. Fuzzy classes are depicted in UML notation using a dashed outlined rectangle.

c) Objects, Classes, Inheritance and Encapsulation extended to Fuzzy Objects, Fuzzy Classes, Fuzzy Inheritance and Fuzzy Encapsulation

In building the automated fuzzy tool, constructs of conventional OOP needed to be extended to fuzzy OOP in the design and development stages. In conventional OOP an object is an entity of the real world the remains unique and unchanged over the life-time of the object. An object is associated with a class. An object is fuzzy if it belongs to a fuzzy class (Ma et al. 2012).

A class is a set and methods and attributes that can be called upon by its objects. A fuzzy class is a class with at least one attribute that is a fuzzy value or fuzzy set, for example a project managers rating on the requirement, **allow users to make secure online payments** can be rated in terms of importance as Average, Good or Very Good. The value for the attribute rating is fuzzy.

Classes are organized into hierarchies where a sub-class can be produced from a superclass. The subclass can inherit some of the attributes and methods of the superclass, override some attributes and methods and define some new attributes and methods. This property is called inheritance. A fuzzy class is a class produced by a fuzzy class by means of specialization or if the class is produced by many classes with at least one fuzzy class (Ma et al. 2012).

Encapsulation is a technique for allowing the clients of a class to use the class by interacting with its external interface. This means that the internal implementation of the class is encapsulated from its clients. In a large program this ensures that designers of the class can make changes to the program safely. Classes make use of the principle of data abstraction meaning that users of the classes do not need to understand how operations of the class are implemented. Fuzzy rules are encapsulated into the fuzzy class for fuzzy encapsulation (Dwibedy et al. 2013), for example linguistic values such as very low is entered by an expert decision maker for an evaluation criteria. The conversion of this linguistic value to a fuzzy value is encapsulated by the method *QualitativeToFuzzy().*

d) Object Oriented Relationships extended to Fuzzy Object Oriented Relationships

Four relationships exist amongst classes namely, Association, Aggregation, Generalisation and Dependency. In the construction of the automated fuzzy tool these four relationships were modeled in the design. The approach used to explain each relationship, with examples taken from construction of the automated fuzzy tool project, is as follows: the relationship is described in words; UML diagrams are drawn for the relationship and a program segment in C# code is presented. In the examples below fuzzy classes are denoted with a dashed line while conventional classes are denoted with a solid line in UML notation.

i. Association

An association relates two or more independent objects. In UML notation it is denoted by an open arrowhead to show directionality (Ma et al. 2012).

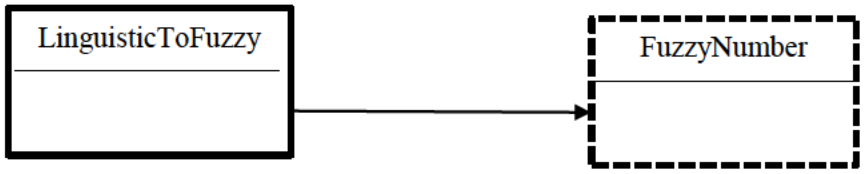


Figure 4.1: **UML showing Association Relationship**

An object of the LinguisticToFuzzy class is connected to objects of the FuzzyNumber class. This association can be described as a fuzzy association. The code below shows how A linguistic term is associated with a FuzzyNumber in the application.

```
public FuzzyNumber LinguisticToFuzzy(string code)
{
...
// Very Poor or Very Low
if (code == "Very Weak" || code == "Very Low")
{
return new FuzzyNumber(1, 1, 3);
}
...
```

}

ii. Aggregation

Aggregation is a 'has-a' relationship that exists between two classes. The parts once created, lives and dies with the whole. In UML it is denoted by ◇——. The arrows in the UML class diagram point towards the composite entity (Ma et al. 2012). Figure 4.2 shows aggregation.

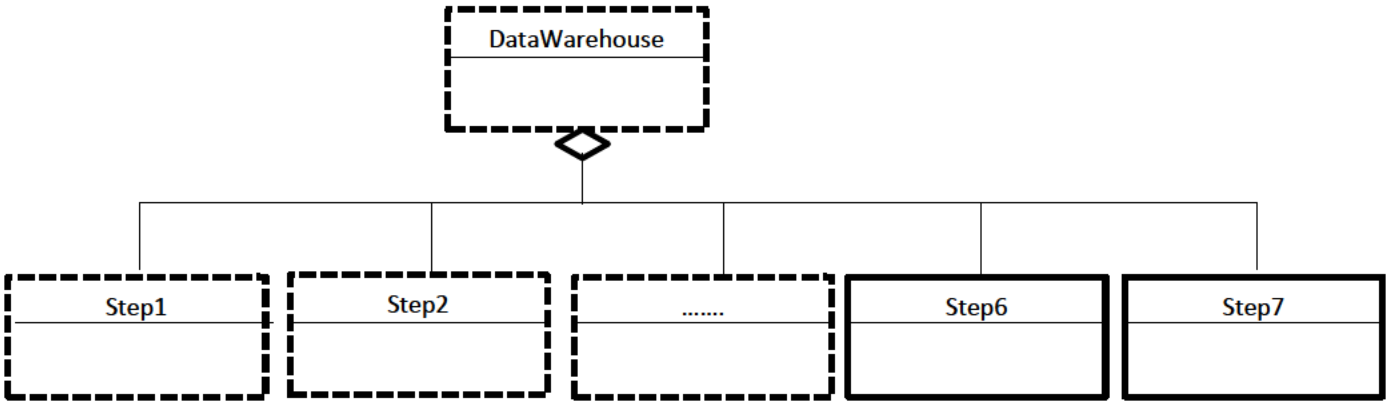


Figure 4. 2: UML showing Aggregation Relationship

The code below shows the DataWarehouse.cs class in the application has references to each StepX class and Utils.cs

```
/// <summary>
/// Contains all the data that will be taken as input from the user
/// </summary>

public class DataWarehouse
{
private Utils _utils;
private Step1 _step1;
private Step2 _step2;
private Step3 _step3;
private Step4 _step4;
private Step5 _step5;
private Step6 _step6;
```

```
private Step7 _step7;
}
```

Generalisation

A generalization relationship is an 'is-a' relationship that occurs between a superclass and a sub-class. The super-class is a generalized form of the more specialized sub-class. In other words a generalization relationship relates a generalized class with a more specific class through inheritance. In the UML class diagram an unfilled triangular arrowhead at the end of the superclass is used to symbolize a generalization relationship (Booch et al. 1997). Figure 4.3 illistrates generalisation.

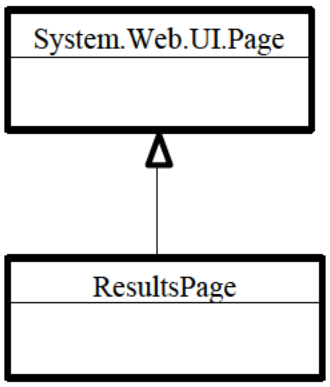


Figure 4. 3: UML showing Generalisation Relationship

In the automated fuzzy tool application all code behind each ASPX Web Form inherits from System.Web.UI.Page. The code below shows the ResultsPage inheriting from System.Web.UI.Page.

```
public partial class ResultsPage : System.Web.UI.Page
{
  /// A ResultsPage is a System.Web.UI.Page
}
```

Dependency

Relates two objects. The changes in one object will affect the other object. The UML notation for dependency as shown in Figure 4.4

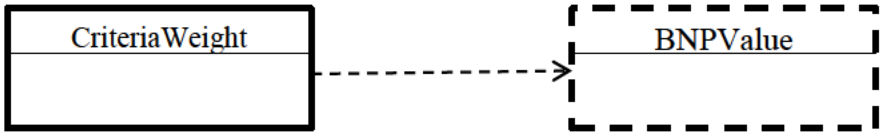


Figure 4. 4: UML showing Dependency Relationship

The BNP value of a criterion depends on its weight. If you adjust the weight, the BNP value will change.

```
public class Criteria
{
...
private FuzzyNumber _fuzzyNumber;
private double _bnpValue;
...
}
```

e) Polymorphism

As key feature of object oriented programming it is important to observe polymorphism and how it can be used in this application. Programmers use polymorphism to create code that is easier to reuse. Polymorphism has ability to provide different implementation of methods that are implemented with the same name. There are two types of polymorphism namely compile-time polymorphism and run-time polymorphism. The former is associated with method overloading whilst the latter is associated with method overriding. Compile-time polymorphism means same method names with different signatures. Run-time polymorphism or method overriding means same method names with same signatures. In run-time polymorphism the binding of the method call to its signature is done dynamically at run-time. Below is a code example of run-time polymorphism based on the application.

```
1. class LinguisticToFuzzy
2. {
3.     virtual void ConvertToFuzzy()  {}
4. }
```

```
5.  class ConvertCriteria : LinguisticToFuzzy
6.   {
7.    override void ConvertToFuzzy()  {}
8.    }
9.    public static void main(String args[])
10.  {
11.   LinguisticToFuzzy  b = new ConvertCriteria  ();//upcasting
12.    b. ConvertToFuzzy ();
13.   }
```

The method in the child class is invoked.

4.2.3.2 The Development: Automated Fuzzy Software Tool implemented in an Object Oriented Programming Language

The approach used in this section is as follows:

a) Each step of the fuzzy TOPSIS algorithm is delineated in words.
b) The mathematical equations for the step are provided,
c) The program implementation for the step is written as a program segment in an object oriented language namely, C#. Step 2 and Step 5 references a utility class with helper methods. The code for this class is available in Annexure I.

Step 1: Determine the weighting of Evaluation Criteria and weighting of Alternatives. Convert linguistic code to its corresponding fuzzy number

Linguistic scales for evaluation criteria and alternatives are represented as a triangular fuzzy number. The fuzzy ratings for the criteria weights and alternative weights by expert decision makers have been preset in the web application as shown in the Table 4.1. Expert decision makers must choose the appropriate linguistic variables for criteria and the appropriate linguistic variables for alternatives with respect to the criteria.

Criteria Weights	Fuzzy Number	Alternative Weights
Very Low (VL)	(1,1,3)	Very weak (VW)
Low (L)	(1,3,5)	Weak (W)
Medium (M)	(3,5,7)	Average (A)
High (H)	(5,7,9)	Good (G)
Very High (VH)	(7,9,9)	Very Good (VG)

Table 4.1: Fuzzy Ratings for the criteria and alternatives

The code below converts the linguistic weight to its corresponding fuzzy weight.

```
public class Step1
{
  /// <summary>
   /// To Convert a linguistic rating to its corresponding FuzzyNumber </summary>
   /// <param name="code"> Linguistic term code </param>
   /// <returns> The fuzzy.FuzzyNumber corresponding to the linguistic code </returns>

   private FuzzyNumber[] _fuzzyNumbers;

   public Step1()
   {
      _fuzzyNumbers = new FuzzyNumber[5];
      _fuzzyNumbers[0] = new FuzzyNumber(1, 1, 3);    // Very Weak or Very Low
      _fuzzyNumbers[1] = new FuzzyNumber(1, 3, 5);    // Weak or Low
      _fuzzyNumbers[2] = new FuzzyNumber(3, 5, 7);    // Average or Medium
      _fuzzyNumbers[3] = new FuzzyNumber(5, 7, 9);    // Good or High
      _fuzzyNumbers[4] = new FuzzyNumber(7, 9, 9);    // Very Good or Very High
   }

   public FuzzyNumber LinguisticToFuzzy(string code)
   {
       // Very Poor or Very Low
      if (code == "Very Weak" || code == "Very Low")
      {
         return _fuzzyNumbers[0];
      }
      // Poor or Low
```

```
        else if (code == "Weak" || code == "Low")
        {
            return _fuzzyNumbers[1];
        }
        // Average or Medium
        else if (code == "Average" || code == "Medium")
        {
            return _fuzzyNumbers[2];
        }
        // Good or High
        else if (code == "Good" || code == "High")
        {
            return _fuzzyNumbers[3];
        }
        // Very Good or Very High
        else if (code == "Very Good" || code == "Very High")
        {
            return _fuzzyNumbers[4];
        }

        throw new System.ArgumentException("Invalid linguistic code used");
    }
}
```

Step 2: Construct the aggregate weights vector and aggregate fuzzy decision matrix

(i) Mathematically aggregate weights of each criterion as follows:

Let K = number of decision makers

Let j = 1,2, ….n representing the criteria

$$\tilde{w}_j^k = (w_{j1}, w_{j2}, w_{j3})$$

$$\text{where: } w_{j1} = \min_k\{w_{jk1}\},\ w_{j2} = \frac{1}{K}\sum_{k=1}^{k} w_{jk2},\ w_{j3} = \max_k\{w_{jk3}\} \quad (1)$$

(ii) Mathematically aggregate weights of each alternative as follows:

Let k = number of decision makers

Let j = 1,2, ….n represent the criteria

Let i = 1,2,….s represent the alternatives

Aggregate weightings $\tilde{x}_{ij} = (l_{ij}, m_{ij}, u_{ij})$ of alternatives (*i*) with respect to each criterion *(j)* based on the fuzzy ratings by the *k* decision maker is expressed by the calculations below:

$$l_{ij} = \min_k\{l_{ij}^k\},\ m_{ij} = \frac{1}{K}\sum_{k=1}^{k} m_{ij}^k,\ u_{ij} = \max_k\{u_{ij}^k\} \quad (2)$$

(iii) Mathematically the aggregate fuzzy decision matrix for the alternatives ($\widetilde{D}$) is expressed below:

$$\widetilde{D} = \begin{array}{c} \\ A_1 \\ A_i \\ A_n \end{array} \begin{array}{c} \begin{array}{cccc} C_1 & C_2 & C_j & C_m \end{array} \\ \begin{bmatrix} \tilde{x}_{11} & \tilde{x}_{12} & \tilde{x}_{1j} & \tilde{x}_{1m} \\ \vdots & \vdots & \vdots & \vdots \\ \tilde{x}_{n1} & \tilde{x}_{n2} & \tilde{x}_{nj} & \tilde{x}_{nm} \end{bmatrix} \end{array} \quad (3)$$

The code for Step 2 is provided below.

```
public class Step2
  {
    Utils _utils = new Utils();

    /// <summary>
    /// Calculates the alternatives aggregate fuzzy values.
    /// </summary>
    /// <returns>The alternatives agg fuzzy values.</returns>
    /// <param name="dMRatings">DMRatings.</param>

    public FuzzyNumber[] CalcAlternativesAggFuzzyValues(string[][] dMRatings)
    {
      FuzzyNumber[] fuzzyValues = new FuzzyNumber[dMRatings[0].Length];
       string[] iCriteriaRatings = new string[dMRatings.Length]; // the criteria ratings as per number of
DMs

      for (int i = 0; i < dMRatings[0].Length; i++)
      {
        for (int j = 0; j < dMRatings.Length; j++)
        {
          iCriteriaRatings[j] = dMRatings[j][i];
```

```
            }
            fuzzyValues[i] = _utils.QualitativeToFuzzy(iCriteriaRatings);
        }
        return fuzzyValues;
    }
}
```

Step 3: Normalise the Fuzzy decision matrix

Let $\tilde{R}$ = normalized decision matrix

Let j = 1,2, ….n represent the criteria

Let i = 1,2,….s represent the alternatives

Let (l_{ij}, m_{ij}, u_{ij}) = aggregated weighting of alternative (i) with respect to criterion (j)

Let $\tilde{r}_{ij}$= triangular fuzzy number of alternative (i) with respect to criterion (j) in matrix r with n rows and s columns

The normalized decision matrix is calculated as follows:

$$\widetilde{(R)} = [\tilde{r}_{ij}]_{nxs}$$

where $\tilde{r}_{ij} = \left(\frac{l_{ij}}{u_j^+}; \frac{m_{ij}}{u_j^+}; \frac{u_{ij}}{u_j^+}\right)$ and $u_j^+ = \max_i u_{ij}$ (benefit criteria) (4)

therefore $u_j^+ = \max_i$ (9,9,9)

The code for the normalisation (Step 3) is provided below.

```
public class Step3
  {
    /// <summary>
    /// Normalize an array of fuzzy numbers
    /// </summary>
    /// <param name="fuzzyValues"></param>
    /// <returns></returns>

    public FuzzyNumber[] NormalizeFuzzyValues(FuzzyNumber[] fuzzyValues)
    {
      FuzzyNumber[] normalizedArray = new FuzzyNumber[fuzzyValues.Length];
      double[] maxArray = new double[fuzzyValues.Length];
```

```
        double iMax;

        // calculate maximum upper value
        for (int j = 0; j < fuzzyValues.Length; j++)
        {
           maxArray[j] = fuzzyValues[j].Max;
        }

        // get the highest upper value
        Array.Sort(maxArray);
        iMax = maxArray[maxArray.Length - 1];

        for (int i = 0; i < fuzzyValues.Length; i++)
        {
           normalizedArray[i] = new FuzzyNumber(fuzzyValues[i].Min / iMax, fuzzyValues[i].Mean /
iMax,
           fuzzyValues[i].Max / iMax);
        }
        return normalizedArray;
     }
  }
```

Step 4: Compute the weighted normalized decision matrix $(\widetilde{V})$

Let j = 1,2, ….n represent the criteria

Let i = 1,2,….s represent the alternatives

The weights of the evaluation criteria, $(\widetilde{w}_j)$ is multiplied by the elements $\tilde{r}_{ij}$ of the normalized fuzzy decision matrix.

$$(\widetilde{V}) = [\tilde{v}_{ij}]_{nxs} \quad (5)$$

where $\tilde{v}_{ij} = \tilde{r}_{ij}$ x $\widetilde{w}_j$

The code for Step 4 is provided below.

```
public class Step4
   {
      /// <summary>
      /// Multiplies an array of weights to an array of FuzzyNumbers
      /// </summary>
      /// <param name="normalizeArray"></param>
      /// <param name="weights"></param>
      /// <returns></returns>

      public FuzzyNumber[] CalculateWeightedValues(FuzzyNumber[] normalizeArray, FuzzyNumber[] weights)
      {
         FuzzyNumber[] weightedArray = new FuzzyNumber[weights.Length];
         FuzzyNumber iFuzzyNumber;
         for (int i = 0; i < weights.Length; i++)
         {
            iFuzzyNumber = normalizeArray[i];
            FuzzyNumber weightedFuzzyNumber = new FuzzyNumber(iFuzzyNumber);

            weightedFuzzyNumber.Max = weightedFuzzyNumber.Max * weights[i].Max;
            weightedFuzzyNumber.Mean = weightedFuzzyNumber.Mean * weights[i].Mean;
            weightedFuzzyNumber.Min = weightedFuzzyNumber.Min * weights[i].Min;
            weightedArray[i] = weightedFuzzyNumber;
         }

         return weightedArray;
      }
   }
```

Step 5: Determine the fuzzy positive-ideal solution (FPIS, A^+) and the fuzzy negative-ideal solution (FNIS, A^-)

Let j = 1,2, ….n represent the criteria

$A^+ = \{ \tilde{v}_1^+, \tilde{v}_j^+, \ldots, \tilde{v}_n^+ \}$ (6)

$A^- = \{ \tilde{v}_1^-, \tilde{v}_j^-, \ldots, \tilde{v}_n^- \}$ (7)

where $\tilde{v}_j^+ = (1, 1, 1)$ and $\tilde{v}_j^- = (0, 0, 0)$

The code for Step 5 is provided below.

```
public class Step5
{
    Utils _utils = new Utils();
    /// <summary>
    /// Calculates the fnis.
    /// </summary>
    /// <returns>The fnis.</returns>
    /// <param name="normalizedValues">Weighted normalized Criteria values.</param>

    public FuzzyNumber CalcFNIS(FuzzyNumber[] normalizedValues)
    {
       // normalizedValues.Length is the number of Alternatives
       FuzzyNumber fnis = null;
       double[] minArray = new double[normalizedValues.Length];
       double minValue;
       // store all the min values
       for (int i = 0; i < normalizedValues.Length; i++)
       {
          minArray[i] = normalizedValues[i].Min;
       }
       // chose the lowest value
       minValue = _utils.GetMinValue(minArray);
       fnis = new FuzzyNumber(minValue, minValue, minValue);
       return fnis;
    }
    /// <summary>
    /// Calculates the fpis.
    /// </summary>
    /// <returns>The fpis.</returns>
```

```
    /// <param name="normalizedValues">Normalized values.</param>


    public FuzzyNumber CalcFPIS(FuzzyNumber[] normalizedValues)
    {
        // normalizedValues.Length is the number of Alternatives
        FuzzyNumber fpis = null;
        double[] maxArray = new double[normalizedValues.Length];
        double maxValue;
        // store all the min values
        for (int i = 0; i < normalizedValues.Length; i++)
        {
            maxArray[i] = normalizedValues[i].Max;
        }
        // chose the lowest value
        maxValue = _utils.GetMaxValue(maxArray);
        fpis = new FuzzyNumber(maxValue, maxValue, maxValue);
        return fpis;
    }
}
```

Step 6: Calculate the distance d_j^+ and d_j^- of each alternative from $\tilde{v}_j^+$ and $\tilde{v}_j^-$ respectively

Let j = 1,2, ….n represent the criteria

Let i = 1,2,….s represent the alternatives

$$d_i^+ = \sum_{j=1}^{n} d_v(\tilde{v}_{ij}, \tilde{v}_j^+) \quad (8)$$

$$d_i^- = \sum_{j=1}^{n} d_v(\tilde{v}_{ij}, \tilde{v}_j^-) \quad (9)$$

Where d represents the distance between two triangular fuzzy numbers. For example the distance $d(\tilde{v}_{ij}, \tilde{v}_j^+)$ where $\tilde{v}_{ij}$ =(l_1,m_1, u_1) and $\tilde{v}_j^+$ =(l_2,m_2,u_2) is expressed as follows:

$$d(\tilde{v}_{ij}, \tilde{v}_j^+) = \sqrt{\frac{1}{3}\left[(l_1 - l_2)^2 + (m_1 - m_2)^2 + (u_1 - u_2)^2\right]} \quad (10)$$

The code for Step 6 is provided below.

```
public class Step6
{
    /// <summary>
    /// Calculates the distance.
    /// </summary>
    /// <returns>The distance.</returns>
    /// <param name="fuzzyNumber">Fuzzy number.</param>
    /// <param name="idealSolution">Ideal solution. This is either the FNIS or FPIS</param>

    public double CalcDistance(FuzzyNumber fuzzyNumber, FuzzyNumber idealSolution)
    {
       double distance = 0;
       double min, mean, max;
       min = Math.Pow((fuzzyNumber.Min - idealSolution.Min), 2);
       mean = Math.Pow((fuzzyNumber.Mean - idealSolution.Mean), 2);
       max = Math.Pow((fuzzyNumber.Max - idealSolution.Max), 2);
       distance = Math.Sqrt(((double)1 / 3.0) * (min + mean + max));
       return distance;
    }
  }
```

Step 7: Obtain the closeness coefficient (CC_i) and rank the order of alternatives

$$CC_i = \frac{d_i^-}{d_i^+ + d_i^-} \tag{11}$$

The code for step 7 is provided below.

```
public class Step7
 {
    /// <summary>
    /// Calculates the cci.
    /// </summary>
    /// <returns>The cci.</returns>
    /// <param name="dNegative">Fnis.</param>
    /// <param name="dPositive">Fpis.</param>
```

```
    public double CalcCCI(double dNegative, double dPositive)
    {
        double cci = (dNegative / (dNegative + dPositive));
        return cci;
    }
}
```

The alternative with the highest closeness coefficient represents the best alternative and is closest to the FPIS and farthest from FNIS. Thereafter the ranking of the alternatives according to the closeness coefficient, CCi, in decreasing order is defined. The best alternative is closest to the FPIS and farthest to the FNIS.

4.2.4 Stage 4: Demonstration

The automated fuzzy tool was demonstrated and tested by software developers using the test website www.fuzzytool.co.za.[1] The domain name was registered by the researcher. The website was hosted on a web server belonging a web application hosting company. The demonstration was validated and verified by showing the construction of all intermediate judgement matrices, distance measurement calculations and how final CCi values were computed. The calculations were shown to stakeholders to prove the correctness of the results given by the tool at the time of the demonstration. These calculations took place programmatically at the backend of the automated fuzzy tool web application.

In this section, the demonstration is illustrated using requirements from a software development project at a study site and the actual values input by the various stakeholders.

The approach for this section therefore, is as follows:

i. Appropriate screen dumps of the automated fuzzy tool are depicted according to the steps of the fuzzy TOPSIS algorithm given in Chapter 2 Section 2.4.2.
ii. Brief explanations accompany the screen dumps.

[1] The online web application is currently live.

iii. Manual mathematical calculations using fuzzy TOPSIS equations are provided to show how the results of intermediate matrices and final CCi values are computed. Detailed calculations are shown to validate the final output of the automated fuzzy tool using the input values given, when the tool was demonstrated.

iv. The fully constructed fuzzy decision matrices and table of CCi values resulting from (iii) above is shown.

4.2.4.1 Login screen

Figure 4.5 depicts the login screen of the web application. A username and password is required to gain access to the system.

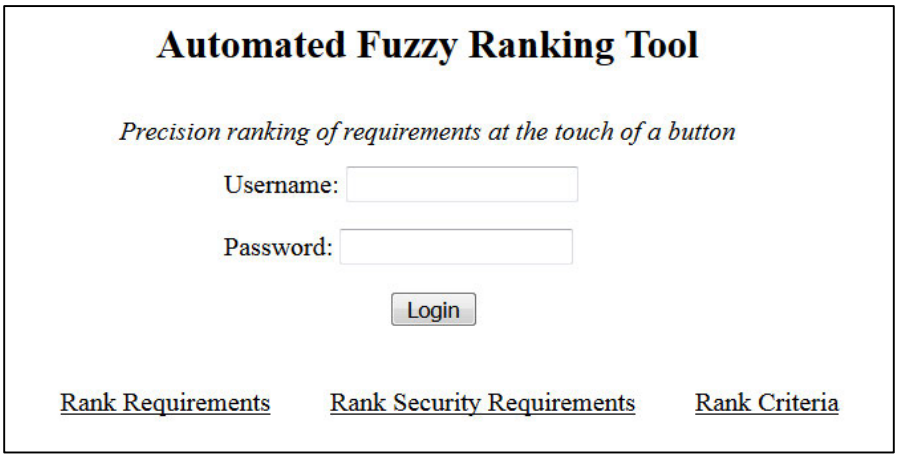


Figure 4.5: Screen Dump of Login Screen

The automated fuzzy tool can rank normal requirements, rank security requirements in the threats and vulnerability table. Further it will allow decision makers to view ratio data on how they prioritised the evaluation criteria as the criteria is ranked as well.

4.2.4.2 Set up parameters

Figure 4.6 depicts screen for inputting project information namely, number of project decision makers, number of criteria and number of requirements of the automated fuzzy tool.

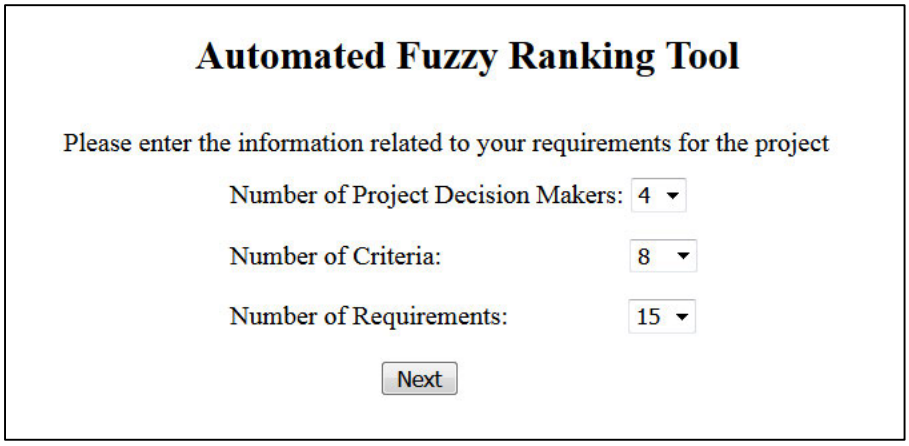


Figure 4.6: **Screen Dump of Set-up Parameter Screen**

The tool was demonstrated with 4 expert project decision makers, 8 project criteria and 15 requirements inclusive of non-functional requirements.

4.2.4.3 Expert decision makers

Figure 4.7 shows the ASD roles of the expert decision makers for the project.

Figure 4.7: Screen Dump of Expert Decision Makers

The expert decision makers who conducted the demonstration of the automated fuzzy tool were the customer (off-site), the product owner (on-site), the business analyst (team-member) and the scrum master (team leader) as shown in Figure 4.7.

4.2.4.4 Input criteria for prioritisation

Decision makers through consensus chose the following criteria by which user requirements were prioritised, namely:

C1 Right personnel are available to implement feature

C2 No dependencies: Dependent requirements get low priority

C3 Less mental effort required to implement

C4 Core: Core to system and must get high priority

C5 Small in size to implement: generates only few user stories

C6 High business value

C7 Low cost (cheap) to implement

C8 Low risk: no chances of volatility/uncertainty

Figure 4.8 is a screen dump of the criteria entered in the criteria screen of the automated fuzzy tool. All criteria entered were benefit criteria.

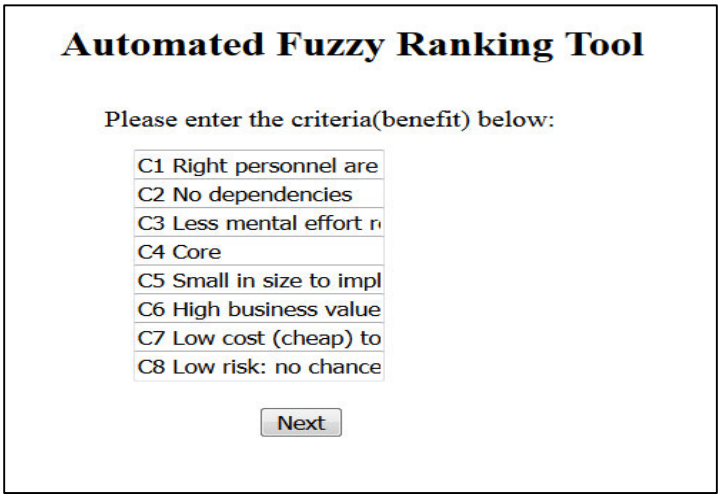


Figure 4.8: **Screen Dump of Input Criteria**

4.2.4.4 User requirements

The Business Requirements Document (BRD) for an online book store specified roles for customers, an administrator, a stock manager and a dispatch clerk. The aim of the development was to deliver features in the form of several releases. The business requirements, that were drafted by the business analyst have been extrapolated and presented by the project manager in a kick off meeting, are listed below:

R1 Access control: Unauthorised Users, Registered Users and Privileged Users

R2 Login, Registration and Home Page

R3 Customizable reports for various roles: customer, administrator, dispatch clerk

R4 Courier costs per contracted vendor (transport companies)

R5 Product module: catalogue of products, services and promotions with terms and conditions

R6 Encryption algorithms to scramble data into unreadable text

R7 Administrator module: add items to catalogue, delete items from catalogue, user accounts

R8 Delivery module: assign courier company, calculate cost of delivery for customer, tracking

R9 Database backup and recovery of customers, shopping carts, orders, inventory, order transaction and delivery.

R10 Inventory module: view, add, delete, edit with automatic purchase and sales updates, alerts and re-order levels

R11 Checkout module: Order, cost of purchases in a shopping cart, process and record payments

R12 Search for a product and service

R13 Dispatch Clerk module: manage customer orders

R14 Display and store promotions

R15 Send out specials and general information via e-mail and SMS.

A screen dump of the demonstration of the user requirements screen in the automated fuzzy tool is shown in Figure 4.9:

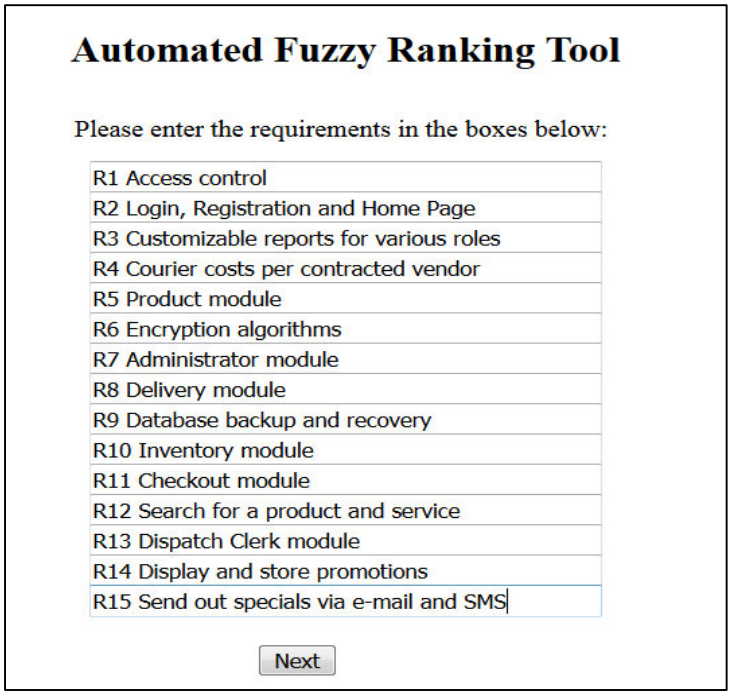


Figure 4.9: Screen ump of User Requirement Screen

4.2.4.5 Decision makers rate criteria

Table 4.2 shows linguistic ratings in column 1 and the corresponding fuzzy rating in column 2 for evaluation criteria. The fuzzy ratings are expressed as a triangular fuzzy number using a scale from 1 to 9. The fuzzy number chosen for a linguistic rating take into consideration the fuzziness of that linguistic rating, for example Medium (M) is represented as (3, 5, 7).

Linguistic term	Membership function
Very Low (VL)	(1,1,3)
Low (L)	(1,3,5)
Medium (M)	(3,5,7)
High (H)	(5,7,9)
Very High (VH)	(7,9,9)

Table 4.2: Fuzzy Numbers for Linguistic Ratings of Criteria

Table 4.3 shows the linguistic weighting given to each criterion by the Customer (DM1), the Product Owner (DM2), the Business Analyst (DM3) and the Scrum Master (DM4).

Criteria	DM1	DM2	DM3	DM4
C1	VL	L	M	M
C2	M	M	VL	L
C3	L	VL	M	M
C4	VH	VH	H	H
C5	M	L	L	VL
C6	M	L	L	H
C7	VL	L	M	VL
C8	M	H	H	H

Table 4.3: Linguistic Ratings for Criteria by Different Decision Makers

Figure 4.10 was extracted from the "rate criteria" screen by all decision makers. The screen dump indicates the choices made by first decision maker namely, the customer in terms of each criterion. The input in Figure 4.10 taken directly from the automated fuzzy tool corresponds with column 1 of the judgement matrix in Table 4.3 above. Similarly column 2 (DM1), column 3 (DM2), column 4 (DM3) and column 5 (DM4) of Table 4.4 was completed by the other expert decision makers namely, Product Owner, Business Analyst and Scrum Master using the automated fuzzy tool with a similar input interface shown in Figure 4.6.

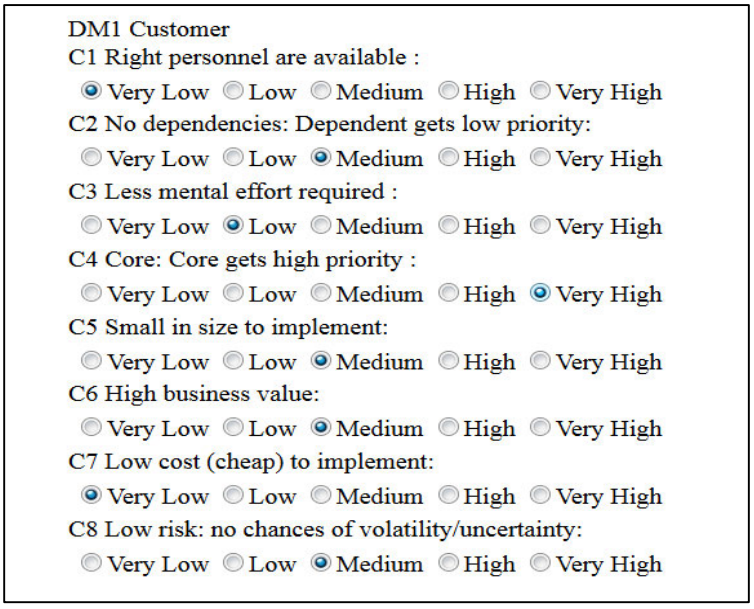


Figure 4.10: Screen Dump of Rate Criteria Screen

Once all inputs were completed by decision makers a fuzzy judgement matrix was created using the fuzzy rating scale as shown in Table 4.3, for example DM3 on criteria 6 was low (L). The

corresponding fuzzy rating for L is (1, 3, 5) as shown in Table 4.4 below. Similarly Table 4.4 was populated with all the other fuzzy weights for criteria.

Criteria	DM1			DM2			DM3			DM4		
C1	(1,	1,	3)	(1,	3,	5)	(3,	5,	7)	(3,	5,	7)
C2	(3,	5,	7)	(3,	5,	7)	(1,	1,	3)	(1,	3,	5)
C3	(1,	3,	5)	(1,	1,	3)	(3,	5,	7)	(3,	5,	7)
C4	(7,	9,	9)	(7,	9,	9)	(5,	7,	9)	(5,	7,	9)
C5	(3,	5,	7)	(1,	3,	5)	(1,	3,	5)	(1,	1,	3)
C6	(3,	5,	7)	(1,	3,	5)	(1,	3,	5)	(5,	7,	9)
C7	(1,	1,	3)	(1,	3,	5)	(3,	5,	7)	(1,	1,	3)
C8	(3,	5,	7)	(5,	7,	9)	(5,	7,	9)	(5,	7,	9

Table 4.4: Fuzzy Weights of Criteria

The fuzzy weights of each criteria were now aggregated to get the aggregated fuzzy weight ($\widetilde{w}_j$) of criterion (C_j) using equation (1):

$$w_{j1} = min\{w_{jk1}\},\ w_{j2} = \frac{1}{K}\sum_{k=1}^{k} w_{jk2},\ w_{j3} = max\{w_{jk3}\} \qquad (1)$$

Table 4.5 shows the aggregated fuzzy matrix for criteria. Taking C3, **Less mental effort required to implement**, as an example, we find from Table 4.5 that the aggregated fuzzy weight is shown as (1, 3.5, 7). It is noted from Table 4.4 above the rating for C3, **Less mental effort required to implement**, was (1, 3, 5) by the Customer represented as DM1; (1,1, 3) by the Product Owner represented as DM2; (3, 5, 7) by the Business Analyst represented as DM3 and (3, 5, 7) by the Scrum Master represented as DM4 in the system. Using equation (I) the lower bound value was obtained by finding the minimum of 1, 1, 3 and 3, the middle number was obtained as follows: (3 + 1 + 5 + 5)/4 = 3.5 and the upper bound was determined by finding the maximum of 5, 3, 7 and 7. Hence the aggregated fuzzy weight for C3, **Less mental effort required to implement** is (1, 3.5, 7) as shown in Table 4.6 below. Similarly aggregate fuzzy weights for C1, C2, C4, C5, C6, C7 and C8 were obtained.

Criteria	Aggregate		
C1	(1,	3.5,	7)
C2	(1,	3.5,	7)
C3	(1,	3.5,	7)
C4	(5,	8,	9)
C5	(1,	3,	7)
C6	(1,	4.5,	9)
C7	(1,	2.5,	7)
C8	(3,	6.5,	9)

Table 4.5: Aggregate fuzzy weights for criteria

4.2.3.5 Decision makers rate requirements based on each criterion

Table 4.6 below shows linguistic ratings in column 1 and the corresponding fuzzy rating for the requirements (alternatives) in column 2. The fuzzy ratings are expressed as triangular fuzzy number using a scale from 1 to 9. The fuzzy number chosen for a linguistic rating takes into consideration the fuzziness of that linguistic rating, for example Very Poor (VP) is represented as (1, 1, 3) in Table 4.6.

Linguistic term	Membership function
Very Poor (VP)	(1,1,3)
Poor (P)	(1,3,5)
Fair(F)	(3,5,7)
Good (G)	(5,7,9)
Very Good (VG)	(7,9,9)

Table 4.6: Fuzzy Numbers for Linguistic Ratings of Requirements

Table 4.7 shows the population of the judgement matrix by the customer for each requirement based on criterion 1 to criterion 8, for example requirement 2 **(R2 Login, Registration and Home Page)** based on criterion 6 **(C6 High business value)** was rated VG by the customer.

		C1	C2	C3	C4	C5	C6	C7	C8
Customer	**R1**	VW	VW	W	A	G	VG	A	G
	R2	VG	G	W	A	G	VG	G	G
	R3	VW	W	A	G	VG	VG	W	VW
	R4	VW	VW	W	A	VW	A	A	VW
	R5	VG	G	W	A	G	VG	G	G
	R6	W	W	A	G	VG	G	W	VW
	R7	VG	VG	W	A	G	W	A	G

	R8	VG	A	A	A	G	VG	G	G
	R9	VG	G	A	G	VG	A	W	VW
	R10	W	VG	W	A	G	VG	A	G
	R11	A	W	W	A	G	VG	G	G
	R12	A	A	A	G	VG	VG	W	VW
	R13	W	W	W	A	G	VG	G	G
	R14	G	G	A	G	VG	VG	W	VW
	R15	VG	VG	A	G	VG	A	W	VW

Table 4.7: Linguistic Rating for Requirements by Customer

Table 4.8 shows the population of the judgement matrix by the Product Owner for each requirement based on criterion 1 to criterion 8, for example requirement 5 **(R5 Product module: catalogue of products, services and promotions with terms and conditions)** based on criterion 1 **(Right personnel are available to implement feature)** was rated W by the Product Owner.

		C1	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**
Product Owner	**R1**	**VG**	**VG**	**W**	**A**	**G**	**A**	**A**	**G**
	R2	A	A	VG	A	G	VG	G	G
	R3	A	W	VG	G	VG	VG	W	VW
	R4	VG	VW	W	A	G	VG	A	G
	R5	W	G	W	A	G	A	G	G
	R6	VG	VG	A	G	A	G	W	VW
	R7	W	A	VG	G	G	W	A	G
	R8	G	W	A	A	G	VG	G	G
	R9	G	VG	A	G	A	A	W	VW
	R10	VG	W	W	VG	G	VG	A	G
	R11	VG	A	G	A	G	VG	G	G
	R12	G	G	VG	G	VG	VG	W	A
	R13	VG	W	G	A	G	VG	G	G
	R14	VG	G	W	G	VG	VG	W	VW
	R15	W	VG	A	G	VG	G	W	VW

Table 4.8: Linguistic rating for requirements by Product Owner

Table 4.9 shows the population of the judgement matrix by the Business Analyst for each requirement based on criterion 1 to criterion 8, for example requirement 8 (**Delivery module: assign courier company, calculate cost of delivery for customer, tracking)** based on criterion 7 **(Low cost to implement)** was rated G by the Business Analyst.

		C1	C2	C3	C4	C5	C6	C7	C8
Business Analyst	**R1**	G	VW	W	A	G	G	A	G
	R2	VG	VG	W	A	VG	VG	G	G
	R3	VW	A	A	G	VG	VG	A	VW
	R4	VW	VG	W	A	G	W	A	G
	R5	VG	G	VG	A	G	G	G	G
	R6	VG	G	A	G	VG	G	W	VW
	R7	VG	G	W	A	G	W	A	G
	R8	VG	W	W	A	G	G	G	G
	R9	G	VG	A	VG	VG	A	W	VW
	R10	W	VG	W	A	VG	VG	A	G
	R11	A	W	W	A	VG	VG	G	G
	R12	A	A	A	VG	VG	VG	W	VW
	R13	G	W	G	A	VG	VG	VG	VG
	R14	G	G	A	VG	VG	VG	W	VW
	R15	VG	VG	A	VG	VG	A	W	VW

Table 4.9: Linguistic Rating for Requirements by Business Analyst

Table 4.10 shows the population of the judgement matrix by the Scrum Master for each requirement based on criterion 1 to criterion 8, for example requirement 15 **(R15 Send out specials and general information via e-mail and SMS)** based on criterion 8 **(Low risk: no chances of volatility/uncertainty)** was rated VW by the Scrum Master.

		C1	C2	C3	C4	C5	C6	C7	C8
Scrum Master	**R1**	G	W	VW	A	G	VG	A	G
	R2	VG	A	VW	A	G	VG	G	G
	R3	VW	W	A	G	VG	A	W	VW
	R4	VW	VW	A	A	G	VG	A	G
	R5	VG	A	VW	A	G	VG	G	G
	R6	W	W	A	G	VG	A	W	VW
	R7	VG	VG	A	A	G	W	A	G
	R8	VG	A	A	A	G	VG	G	G
	R9	VG	G	A	G	VG	A	VW	VW
	R10	W	VG	VW	A	G	A	A	G
	R11	A	W	VW	A	G	VG	G	G
	R12	A	A	A	G	VG	VG	W	VW
	R13	W	W	VW	A	G	VG	G	G
	R14	G	G	A	G	VG	A	VW	VW
	R15	VG	VG	A	G	A	A	VW	VW

Table 4.10: Linguistic Rating for Requirements by Scrum Master

Decision makers ratings from Table 4.7, Table 4.8, Table 4.9 and Table 4.10 were then converted to a fuzzy decision matrix of requirements under each criterion. The table is now represented by fuzzy numbers, for example the rating by DM1, Customer for R2C6 was VW (1,1, 3), DM2, Product Owner for R5C1 was A (3, 5, 7), DM3, Business Analyst for R8C7 was VW (1, 1, 3) and DM 4, Scrum Master for R15C8 was VW (1,1, 3).

The fuzzy numbers are pooled together to get the aggregated fuzzy rating $\tilde{x}_{ij} = (l_{ij}, m_{ij}, u_{ij})$ of requirement R_i under criterion C_j using equation (2):

$$l_{ij} = min\{l_{ij}^k\},\ m_{ij} = \frac{1}{K}\sum_{k=1}^{k} m_{ij}^k,\ u_{ij} = max\{u_{ij}^k\} \quad (2)$$

Table 4.11 shows the Aggregate fuzzy decision matrix for requirements.

	C1			C2			C3			C4			C5			C6			C7			C8		
R1	(1,	6,	9)	(1,	3.5,	9)	(1,	2.5,	5)	(3,	5,	7)	(5,	7,	9)	(3,	7.5,	9)	(3,	5,	7)	(5,	7,	9)
R2	(3,	8,	9)	(3,	6.5,	9)	(1,	4,	9)	(3,	5,	7)	(5,	7.5,	9)	(7,	9,	9)	(5,	7,	9)	(5,	7,	9)
R3	(1,	2,	7)	(1,	3.5,	7)	(3,	6,	9)	(5,	7,	9)	(7,	9,	9)	(3,	8,	9)	(1,	3.5,	7)	(1,	1,	3)
R4	(1,	3,	9)	(1,	3,	9)	(1,	3.5,	7)	(3,	5,	7)	(1,	5.5,	9)	(1,	6.5,	9)	(3,	5,	7)	(1,	5.5,	9)
R5	(1,	7.5,	9)	(3,	6.5,	9)	(1,	4,	9)	(3,	5,	7)	(5,	7,	9)	(3,	7.5,	9)	(5,	7,	9)	(5,	7,	9)
R6	(1,	6,	9)	(1,	5.5,	9)	(3,	5,	7)	(5,	7,	9)	(3,	8,	9)	(3,	6.5,	9)	(1,	3,	5)	(1,	1,	3)
R7	(1,	7.5,	9)	(3,	7.5,	9)	(1,	5,	9)	(3,	5.5,	9)	(5,	7,	9)	(1,	3,	5)	(3,	5,	7)	(5,	7,	9)
R8	(5,	8.5,	9)	(1,	4,	7)	(1,	4.5,	7)	(3,	5,	7)	(5,	7,	9)	(5,	8.5,	9)	(5,	7,	9)	(5,	7,	9)
R9	(5,	8,	9)	(5,	8,	9)	(3,	5,	7)	(5,	7.5,	9)	(3,	8,	9)	(3,	5,	7)	(1,	2.5,	5)	(1,	1,	3)
R10	(1,	4.5,	9)	(1,	7.5,	9)	(1,	2.5,	5)	(3,	6,	9)	(5,	7.5,	9)	(3,	8,	9)	(3,	5,	7)	(5,	7,	9)
R11	(3,	6,	9)	(1,	3.5,	7)	(1,	3.5,	9)	(3,	5,	7)	(5,	7.5,	9)	(7,	9,	9)	(5,	7,	9)	(5,	7,	9)
R12	(3,	5.5,	9)	(3,	5.5,	9)	(3,	6,	9)	(5,	7.5,	9)	(7,	9,	9)	(7,	9,	9)	(1,	3,	5)	(1,	2,	7)
R13	(1,	5.5,	9)	(1,	3,	5)	(1,	4.5,	9)	(3,	5,	7)	(5,	7.5,	9)	(7,	9,	9)	(5,	7.5,	9)	(5,	7.5,	9)
R14	(5,	7.5,	9)	(5,	7,	9)	(3,	4.5,	7)	(5,	7.5,	9)	(7,	9,	9)	(3,	8,	9)	(1,	2.5,	5)	(1,	1,	3)
R15	(1,	7.5,	9)	(7,	9,	9)	(3,	5,	7)	(5,	7.5,	9)	(3,	8,	9)	(3,	5.5,	9)	(1,	2.5,	5)	(1,	1,	3)

Table 4.11: Aggregate Fuzzy Decision Matrix for Requirements

Using requirement 3 **(R3 Customizable reports for various roles: customer, administrator, dispatch clerk)** and Criterion 1 **(C1 Right personnel are available to implement feature)** as an

example it can be illustrated how Table 4.11 was constructed. The rating by DM 1, the customer was VW (1,1, 3), DM2, the Product Owner was A (3, 5, 7), DM3, the Business Analyst was VW (1, 1, 3) and DM4, Scrum Master was VW (1,1, 3) as shown in Table 4.7, 4.8, 4.9 and 4.10 respectively. The aggregated triple was calculated as follows: lower bound = min 1, 3, 1, and 1, middle number = (1+5+1+1)/4= 2 and upper bound = max 3, 7, 3 and 3. Hence an aggregate rating of (1, 2, 7) was obtained for R3 C1 as shown in Table 4.11.

All other values in Table 4.11 were populated similarly.

	C1			C2			C3			C4			C5			C6			C7			C8		
R1	(0.11,	0.67,	1)	(0.11,	0.39,	1)	(0.11,	0.28,	0.56)	(0.33,	0.56,	0.78)	(0.56,	0.78,	1)	(0.33,	0.83,	1)	(0.33,	0.56,	0.78)	(0.56,	0.78,	1)
R2	(0.33,	0.89,	1)	(0.33,	0.72,	1)	(0.11,	0.44,	1)	(0.33,	0.56,	0.78)	(0.56,	0.83,	1)	(0.78,	1,	1)	(0.56,	0.78,	1)	(0.56,	0.78,	1)
R3	(0.11,	0.22,	0.78)	(0.11,	0.39,	0.78)	(0.33,	0.67,	1)	(0.56,	0.78,	1)	(0.78,	1,	1)	(0.33,	0.89,	1)	(0.11,	0.39,	0.78)	(0.11,	0.11,	0.33)
R4	(0.11,	0.33,	1)	(0.11,	0.33,	1)	(0.11,	0.39,	0.78)	(0.33,	0.56,	0.78)	(0.11,	0.61,	1)	(0.11,	0.72,	1)	(0.33,	0.56,	0.78)	(0.11,	0.61,	1)
R5	(0.11,	0.83,	1)	(0.33,	0.72,	1)	(0.11,	0.44,	1)	(0.33,	0.56,	0.78)	(0.56,	0.78,	1)	(0.33,	0.83,	1)	(0.56,	0.78,	1)	(0.56,	0.78,	1)
R6	(0.11,	0.67,	1)	(0.11,	0.61,	1)	(0.33,	0.56,	0.78)	(0.56,	0.78,	1)	(0.33,	0.89,	1)	(0.33,	0.72,	1)	(0.11,	0.33,	0.56)	(0.11,	0.11,	0.33)
R7	(0.11,	0.83,	1)	(0.33,	0.83,	1)	(0.11,	0.56,	1)	(0.33,	0.61,	1)	(0.56,	0.78,	1)	(0.11,	0.33,	0.56)	(0.33,	0.56,	0.78)	(0.56,	0.78,	1)
R8	(0.56,	0.94,	1)	(0.11,	0.44,	0.78)	(0.11,	0.5,	0.78)	(0.33,	0.56,	0.78)	(0.56,	0.78,	1)	(0.56,	0.94,	1)	(0.56,	0.78,	1)	(0.56,	0.78,	1)
R9	(0.56,	0.89,	1)	(0.56,	0.89,	1)	(0.33,	0.56,	0.78)	(0.56,	0.83,	1)	(0.33,	0.89,	1)	(0.33,	0.56,	0.78)	(0.11,	0.28,	0.56)	(0.11,	0.11,	0.33)
R10	(0.11,	0.5,	1)	(0.11,	0.83,	1)	(0.11,	0.28,	0.56)	(0.33,	0.67,	1)	(0.56,	0.83,	1)	(0.33,	0.89,	1)	(0.33,	0.56,	0.78)	(0.56,	0.78,	1)
R11	(0.33,	0.67,	1)	(0.11,	0.39,	0.78)	(0.11,	0.39,	1)	(0.33,	0.56,	0.78)	(0.56,	0.83,	1)	(0.78,	1,	1)	(0.56,	0.78,	1)	(0.56,	0.78,	1)
R12	(0.33,	0.61,	1)	(0.33,	0.61,	1)	(0.33,	0.67,	1)	(0.56,	0.83,	1)	(0.78,	1,	1)	(0.78,	1,	1)	(0.11,	0.33,	0.56)	(0.11,	0.22,	0.78)
R13	(0.11,	0.61,	1)	(0.11,	0.33,	0.56)	(0.11,	0.5,	1)	(0.33,	0.56,	0.78)	(0.56,	0.83,	1)	(0.78,	1,	1)	(0.56,	0.83,	1)	(0.56,	0.83,	1)
R14	(0.56,	0.83,	1)	(0.56,	0.78,	1)	(0.33,	0.5,	0.78)	(0.56,	0.83,	1)	(0.78,	1,	1)	(0.33,	0.89,	1)	(0.11,	0.28,	0.56)	(0.11,	0.11,	0.33)
R15	(0.11,	0.83,	1)	(0.78,	1,	1)	(0.33,	0.56,	0.78)	(0.56,	0.83,	1)	(0.33,	0.89,	1)	(0.33,	0.61,	1)	(0.11,	0.28,	0.56)	(0.11,	0.11,	0.33)

Table 4.12: Normalised Aggregate Fuzzy Decision Matrix

Table 4.12 shows the constructed normalized fuzzy decision matrix. This matrix is constructed with equation (3)

$$\widetilde{(R)} = [\tilde{r}_{ij}]_{mxn}$$

$$\tilde{r}_{ij} = \left(\frac{l_{ij}}{u_j^+}; \frac{m_{ij}}{u_j^+}; \frac{u_{ij}}{u_j^+}\right) \text{ and } u_j^+ = \max_i u_{ij} \quad \text{(benefit criteria)}$$

Therefore $u_j^+ = \max_i$ (9,9,9)

Consider R3C1 (1, 2, 7) and R15C5 (3, 8, 9) from Table 4.12 as examples. The normalized values were obtained from Table 4.11 as follows (1/9,2/9,7/9) and (3/9, 8/9, 9/9) to obtain (0.111,0.222,0.778) and (0.111, 0.889, 1) respectively, as shown in Table 4.12 above.

Similarly other normalized values in Table 4.12 were obtained using values from the Aggregate fuzzy decision matrix for requirements in Table 4.11.

	C1			C2			C3			C4			C5			C6			C7			C8		
R1	(0.11,	2.33,	7)	(0.11,	1.36,	7)	(0.11,	0.97,	3.89)	(1.67,	4.44,	7)	(0.56,	2.33,	7)	(0.33,	3.75,	9)	(0.33,	1.39,	5.44)	(1.67,	5.06,	9)
R2	(0.33,	3.11,	7	(0.33,	2.53,	7)	(0.11,	1.56,	7)	(1.67,	4.44,	7)	(0.56,	2.5,	7)	(0.78,	4.5,	9)	(0.56,	1.94,	7)	(1.67,	5.06,	9)
R3	(0.11,	0.78,	5.44)	(0.11,	1.36,	5.44)	(0.33,	2.33,	7)	(2.78,	6.22,	9)	(0.78,	3,	7)	(0.33,	4,	9)	(0.11,	0.97,	5.44)	(0.33,	0.72,	3)
R4	(0.11,	1.17,	7)	(0.11,	1.17,	7)	(0.11,	1.36,	5.44)	(1.67,	4.44,	7)	(0.11,	1.83,	7)	(0.11,	3.25,	9)	(0.33,	1.39,	5.44)	(0.33,	3.97,	9)
R5	(0.11,	2.92,	7)	(0.33,	2.53,	7)	(0.11,	1.56,	7)	(1.67,	4.44,	7)	(0.56,	2.33,	7)	(0.33,	3.75,	9)	(0.56,	1.94,	7)	(1.67,	5.06,	9)
R6	(0.11,	2.33,	7)	(0.11,	2.14,	7)	(0.33,	1.94,	5.44)	(2.78,	6.22,	9)	(0.33,	2.67,	7)	(0.33,	3.25,	9)	(0.11,	0.83,	3.89)	(0.33,	0.72,	3)
R7	(0.11,	2.92,	7)	(0.33,	2.92,	7)	(0.11,	1.94,	7)	(1.67,	4.89,	9)	(0.56,	2.33,	7)	(0.11,	1.5,	5)	(0.33,	1.39,	5.44)	(1.67,	5.06,	9)
R8	(0.56,	3.31,	7)	(0.11,	1.56,	5.44)	(0.11,	1.75,	5.44)	(1.67,	4.44,	7)	(0.56,	2.33,	7)	(0.56,	4.25,	9)	(0.56,	1.94,	7)	(1.67,	5.06,	9)
R9	(0.56,	3.11,	7)	(0.56,	3.11,	7)	(0.33,	1.94,	5.44)	(2.78,	6.67,	9)	(0.33,	2.67,	7)	(0.33,	2.5,	7)	(0.11,	0.69,	3.89)	(0.33,	0.72,	3)
R10	(0.11,	1.75,	7)	(0.11,	2.92,	7)	(0.11,	0.97,	3.89)	(1.67,	5.33,	9)	(0.56,	2.5,	7)	(0.33,	4,	9)	(0.33,	1.39,	5.44)	(1.67,	5.06,	9)
R11	(0.33,	2.33,	7)	(0.11,	1.36,	5.44)	(0.11,	1.36,	7)	(1.67,	4.44,	7)	(0.56,	2.5,	7)	(0.78,	4.5,	9)	(0.56,	1.94,	7)	(1.67,	5.06,	9)
R12	(0.33,	2.14,	7)	(0.33,	2.14,	7)	(0.33,	2.33,	7)	(2.78,	6.67,	9)	(0.78,	3,	7)	(0.78,	4.5,	9)	(0.11,	0.83,	3.89)	(0.33,	1.44,	7)
R13	(0.11,	2.14,	7)	(0.11,	1.17,	3.89)	(0.11,	1.75,	7)	(1.67,	4.44,	7)	(0.56,	2.5,	7)	(0.78,	4.5,	9)	(0.56,	2.08,	7)	(1.67,	5.42,	9)
R14	(0.56,	2.92,	7)	(0.56,	2.72,	7)	(0.33,	1.75,	5.44)	(2.78,	6.67,	9)	(0.78,	3,	7)	(0.33,	4,	9)	(0.11,	0.69,	3.89)	(0.33,	0.72,	3)
R15	(0.11,	2.92,	7)	(0.78,	3.5,	7)	(0.33,	1.94,	5.44)	(2.78,	6.67,	9)	(0.33,	2.67,	7)	(0.33,	2.75,	9)	(0.11,	0.69,	3.89)	(0.33,	0.72,	3)

Table 4. 13: Weighted Normalized Fuzzy Decision Matrix

Now the weighted normalized fuzzy decision matrix was constructed as shown in Table 4.13. The values were computed by multiplying the weights $\tilde{w}_j$ of evaluation criteria (vector) with the normalized fuzzy decision matrix $\tilde{r}_{ij}$ using the equation $\tilde{v}_{ij} = \tilde{r}_{ij}(.)\tilde{w}_j$. Still taking R3C1 and R15C5 as examples, it is shown how the values for Table 4.13 were computed. In Table 4.12 it was shown that the normalized values for R3C1 was (0.111, 0.222, 0.778) and R15C5 was (0.333, 0.889, 1). Using values from Table 4.5 the weight for C1 was (1, 3.5, 7) and C5 was (1, 3 , 7). Therefore using equation (10) R3C1 was obtained as follows: lower bound = 0.111 x 1, middle = 0.2222 x 3.5, upper bound = 0.778 x 7 and R15C5 was obtained as follows: lower bound = 0.33x1, middle =0.89x3, upper= 1x7. Hence the values for R3C1 is (0.111, 0.778, 5.44) and R15C5 is (0.33, 2.67, 7) as shown in Table 4.13 were obtained. Similarly, other values were computed and Table 4.13 shows the values after these computations were completed.

Now the Fuzzy Negative-Ideal Solution (FNIS) and Fuzzy Positive-Ideal Solution (FPIS) are calculated. The Fuzzy Positive Ideal Solution (FPIS, A+) and the Fuzzy Negative Ideal Solution (FNIS, A-) are defined according to the following equations:

$$A^{+} = \{ \tilde{v}_1^{+}, \tilde{v}_j^{+}, \dots, \tilde{v}_m^{+} \} \quad (8)$$

$$A^{-} = \{ \tilde{v}_1^{-}, \tilde{v}_j^{-}, \dots, \tilde{v}_m^{-} \} \quad (9)$$

Where $\tilde{v}_j^{+}$ = (1, 1, 1) and $\tilde{v}_j^{-}$ = (0, 0, 0)

Table 4.14 shows the FNIS(A-) and FNIS(A+) values. These values are obtained by looking for the maximum and minimum values under each criterion in Table 4.13.

Criteria	**FNIS(A-)**			**FNIS(A+)**		
C1	0.111	0.111	0.111	7.000	7.000	7.000
C2	0.111	0.111	0.111	7.000	7.000	7.000
C3	0.111	0.111	0.111	7.000	7.000	7.000
C4	0.111	0.111	0.111	9.000	9.000	9.000
C5	0.111	0.111	0.111	7.000	7.000	7.000
C6	0.111	0.111	0.111	9.000	9.000	9.000
C7	0.111	0.111	0.111	7.000	7.000	7.000
C8	0.333	0.333	0.333	9.000	9.000	9.000

Table 4. 14: FNIS(A-) and FNIS(A+)

The distance d which represents the distance between two triangular fuzzy numbers, namely the weighted normalized triple (l_1,m_1, u_1) and FNIS or FPIS (l_2,m_2,u_2) are calculated using equation (12) below.

$$d(\tilde{x}, \tilde{z}) = \sqrt{\frac{1}{3}\left[(l_1 - l_2)^2 + (m_1 - m_2)^2 + (u_1 - u_2)^2\right]} \quad (12)$$

Table 4.15 and Table 4.16 shows distance values from FPIS and FNIS respectively. For example in order to find the distance between R5 C8 (1.67, 5.06, 9) and A+ (9, 9, 9) the following calculation was completed:

D(R5,A+) = $\sqrt{\frac{1}{3}[(1.67-9)^2 + (5.06-9)^2 + (9-9)^2]}$ = 4.80, as shown in Table 4.15 below.

	C1	C2	C3	C4	C5	C6	C7	C8		d+
d(R1,A+)	4.804	5.14	5.582	5.116	4.594	5.85	5.11	4.808		41
d(R2,A+)	4.456	4.635	5.069	5.116	4.538	5.412	4.729	4.808		38.76
d(R3,A+)	5.434	5.218	4.698	3.934	4.271	5.777	5.361	7.738		42.43
d(R4,A+)	5.212	5.212	5.218	5.116	4.972	6.112	5.11	5.785		42.74
d(R5,A+)	4.624	4.635	5.069	5.116	4.594	5.85	4.729	4.808		39.42
d(R6,A+)	4.804	4.868	4.913	3.934	4.591	6.005	5.632	7.738		42.48
d(R7,A+)	4.624	4.514	4.933	4.854	4.594	7.101	5.11	4.808		40.54
d(R8,A+)	4.289	5.148	5.081	5.116	4.594	5.594	4.729	4.808		39.36
d(R9,A+)	4.346	4.346	4.913	3.837	4.591	6.36	5.683	7.738		41.81
d(R10,A+)	5.001	4.624	5.582	4.734	4.538	5.777	5.11	4.808		40.17
d(R11,A+)	4.698	5.218	5.14	5.116	4.538	5.412	4.729	4.808		39.66
d(R12,A+)	4.764	4.764	4.698	3.837	4.271	5.412	5.632	6.738		40.11
d(R13,A+)	4.868	5.513	5.001	5.116	4.538	5.412	4.68	4.712		39.84
d(R14,A+)	4.405	4.466	4.981	3.837	4.271	5.777	5.683	7.738		41.16
d(R15,A+)	4.624	4.122	4.913	3.837	4.591	6.169	5.683	7.738		41.68

Table 4.15: Distance from FPIS (A+)

The distances d_j^+ as shown in Table 4.15 and d_j^- as shown in Table 4.16 of each alternative is computed from respectively $\tilde{v}_j^+$ and $\tilde{v}_j^-$ according to equations (10) and (11).

$$d_i^+ = \sum_{j=1}^{n} d_v(\tilde{v}_{ij}, \tilde{v}_j^+) \quad (10)$$

$$d_i^- = \sum_{j=1}^{n} d_v(\tilde{v}_{ij}, \tilde{v}_j^-) \quad (11)$$

	C1	C2	C3	C4	C5	C6	C7	C8		d-
d(R1,A-)	4.179	4.042	2.237	3.472	4.187	5.547	3.169	5.75		32.58
d(R2,A-)	4.34	4.217	4.064	3.472	4.217	5.736	4.124	5.75		35.92
d(R3,A-)	3.103	3.163	4.181	5.025	4.33	5.603	3.119	1.556		30.08
d(R4,A-)	4.024	4.024	3.163	3.472	4.1	5.443	3.169	5.427		32.82
d(R5,A-)	4.294	4.217	4.064	3.472	4.187	5.547	4.124	5.75		35.65
d(R6,A-)	4.179	4.146	3.259	5.025	4.244	5.444	2.221	1.556		30.07
d(R7,A-)	4.294	4.296	4.116	4.625	4.187	2.934	3.169	5.75		33.37
d(R8,A-)	4.392	3.19	3.221	3.472	4.187	5.667	4.124	5.75		34
d(R9,A-)	4.346	4.346	3.259	5.164	4.244	4.212	2.207	1.556		29.33
d(R10,A-)	4.088	4.294	2.237	4.734	4.217	5.603	3.169	5.75		34.09
d(R11,A-)	4.181	3.163	4.042	3.472	4.217	5.736	4.124	5.75		34.69
d(R12,A-)	4.148	4.148	4.181	5.164	4.33	5.736	2.221	3.902		33.83
d(R13,A-)	4.146	2.265	4.088	3.472	4.217	5.736	4.145	5.852		33.92
d(R14,A-)	4.302	4.261	3.224	5.164	4.33	5.603	2.207	1.556		30.65
d(R15,A-)	4.294	4.449	3.259	5.164	4.244	5.355	2.207	1.556		30.53

Table 4.16: Distance from FNIS (A-)

For example the d- value for R3 is computed as 3.103 + 3.163 + 4.181 + 5.025 + 4.33 + 5.603 + 3.119 + 1.556 = 30.08 as shown in Table 4.16.

4.2.3.6 The defuzzified 0utput

The closeness coefficient (CC_i) is computed according to equation (13).

$$CC_i = \frac{d_i^-}{d_i^+ + d_i^-}$$

The alternative with the highest closeness coefficient represents the best alternatives and is closest to the FPIS and farthest from FNIS. For example, the CCi value for R3 can be calculated using equation (13) above is as follows: CCi = 30.08/(30.08+42.43) = 0.41484. Similarly all other CCi values were calculated as shown in Table 4.17.

Requirement	CCi	Ranking
R2	0.48097	1
R5	0.47489	2
R11	0.46656	3
R8	0.4635	4
R13	0.45989	5
R10	0.45907	6
R12	0.45751	7
R7	0.45153	8
R1	0.44278	9
R4	0.43438	10
R14	0.42682	11
R15	0.4228	12
R3	0.41484	13
R6	0.41448	14
R9	0.41229	15

Table 4.17: Closeness Coeffiecients and Ranking of Requirements

Defuzzification of the Best Non fuzzy Performance value (BNP) for a criteria weighting j, is calculated using the following equation (Safari et al. 2012):

$$BNP_{wj} = [(Upper\ bound_{wj} - lower\ bound_{wj}) + (Middle\ bound_{wj} - lower\ bound_{wj})]/3 + lower\ bound_{wj}$$

for example, the aggregated fuzzy value for criteria 1 is (1, 3.5, 7). The calculation for the BNP of criteria 1 is as follows:

$BNP_{w1} = [(Upper\ bound_{w1} - lower\ bound_{w1}) + (Middle\ bound_{w1} - lower\ bound_{w1})]/3 + lower\ bound_{w1}$

BNP = [(7-1) + (3.5-1)]/3 + 1

BNP = 3.83333

Similarly other BNP values were calculated.

Figure 4.11 shows the output of the automated fuzzy ranking tool. The values of Table 4.17 are identical to Figure 4.11 proving the correctness of the demonstration and the accuracy of the automated fuzzy tool.

Automated Fuzzy Ranking Tool

(Defuzzified Output)

Backlog: Closeness Coefficients (CCi) and Ranking:

R2 Login, Registration and Home Page	CCi:	0,481
R5 Product module	CCi:	0,4749
R11Checkout module	CCi:	0,4666
R8 Delivery module	CCi:	0,4635
R13 Dispatch Clerk module	CCi:	0,4599
R10 Inventory module	CCi:	0,4591
R12 Search for a product and service	CCi:	0,4575
R7 Administrator module	CCi:	0,4515
R1 Access control	CCi:	0,4428
R4 Courier costs per contracted vendor	CCi:	0,4344
R14 Display and store promotions	CCi:	0,4262
R15 Send out specials via e-mail and SMS	CCi:	0,4228
R3 Customizable reports for various roles	CCi:	0,4148
R6 Encryption algorithms	CCi:	0,4145
R9 Database backup and recovery	CCi:	0,4123

Ranked Criteria: Best Non Fuzzy Performance (BNP):

C4 Core: Core to system and must get high priority	BNP:	7,3333
C8 Low risk: no chances of volatility/uncertainty	BNP:	6,1667
C6 High business value	BNP:	4,8333
C1 Right personnel are available to implement feature	BNP:	3,8333
C2 No dependencies: Dependent requirements get low priority	BNP:	3,8333
C3 Less mental effort required to implement	BNP:	3,8333
C5 Small in size to implement: generates only few user stories	BNP:	3,6667
C7 Low cost (cheap) to implement	BNP:	3,5

Figure 4.11: Output of Automated Fuzzy Tool

The results of the tool were verified and shown to be valid.

4.2.5 Stage 5: Evaluation

The artifact was evaluated individually by three representatives of intended users of the web application tool. The evaluators represented different Agile Software Development roles namely, customer, business analyst and project manager. The evaluation was conducted at the study sites using requirements from various types of projects. For example, large projects with many requirements, small projects with few requirements, projects with detailed requirements and projects with a mix of detailed and high level requirements. As discussed in the literature review (Chapter 2), researchers emphasized the importance of the prioritisation tool being appropriate to the software development methodology. Therefore questions crafted for the evaluation assesses suitability of the automated fuzzy tool for use in Agile Software Development. The evaluation questionnaire is attached in Annexure F. The evaluation framework focused on the following themes as synthesized and referenced from the literature review (Chapter 2), namely:

- ease of use
- total time taken
- scalability
- suitability to high level requirements
- suitability to detailed requirements
- correctness
- scale of measurements

Each theme is presented below from the data analysis of evaluators' responses.

4.2.5.1 Ease of use

This evaluation criterion describes end users experience on how easy it was to use the automated fuzzy tool. The informants indicated that the tool had simple instructions that were fairly easy to follow. Informants indicated that the use of the drop down list boxes and radio buttons ensured that user input was swift and error free.

However the following were cited as potential improvements:

- Increase text field sizes to accommodate longer inputs and outputs.
- Provide better hints to users during input.

- Provide a graphical output for the end user.

4.2.5.2 Total time taken

This evaluation theme focused on the time taken to complete the process. In general evaluators indicated that the process was very quick.

The time delays experienced were based on how long it took users to make a decision on a requirement.

4.2.5.3 Scalability

There were no issues reported on the ability of the tool to handle projects of various sizes.

4.2.5.4 Suitability to high level requirements

High level requirements are not specified with a great deal of detail and is elicited in the early stages of a project. The consensus reached was that the automated fuzzy tool is efficient for high level requirements.

4.2.5.5 Suitability to detailed requirements

Detailed requirements are specified generally after the analysis of requirements. Generalised requirements become more clearly defined as the project progresses. All participants indicated that the tool is suitable for detailed requirements as the prioritisation was completed just as efficiently as high level requirements.

4.2.5.6 Correctness

The correctness of the tool was illustrated to evaluators through a stepwise walkthrough of the manual calculations using their live input. The nature of the illustration was as per the demonstration in Section 4.2.4. This was presented at the feedback session. All three evaluators were satisfied with the explanations and accepted the results of the tool as accurate.

4.2.5.7 Scale of measurements

Scale of measurement is the scale on which the ranking is based. According to Karlsson et al. (1998) the ratio scale is the strongest scale of measurement. The automated fuzzy tool is based on fuzzy TOPSIS which presents output on a ratio scale using mathematical operations. All evaluators were impressed with the scale of measurement as it was able to present the output with the magnitude of the rating and not just output an ordered list of requirements. Evaluators found information on the magnitude extremely useful because if the differences between two requirements were marginal, more practical decisions can be taken during implementation, for example, a requirement ranked slightly lower could be implemented immediately if resources for the higher rated requirement were not available for some reason. Magnitude also gives the developers a general idea on the importance of a requirement.

It was suggested that it would be more meaningful to the end users if the magnitude was expressed graphically. This will be most valuable as differences in CCi values of requirements can be easily detected at a glance rather than inspecting the numerical CCi values. All informants were in agreement that this modification will enhance the presentation of the output as well as make the tool easier to interpret.

Conclusion of evaluation

The overall results yielded by the tool are trustworthy and correct. The tool worked efficiently for a mix of high level as well as detailed requirements. The tool can be used at any stage in Agile RE, for example at the initial planning stage when requirements are more high level or prior to the iterations, after requirements analysis when more detailed requirements are specified or during the security risk analysis stage when ranking security requirements.

There were few general suggestions for improvement of the tool. This is summarised as follows:

- A larger area for input of long winded requirements.
- A screen at the end summarizing how all decision makers ranked requirements to allow decision makers an opportunity for self-evaluation. It will also be used as a reference to improve their judgements in future prioritisation activities.

- The tool must be enhanced to support remote login for administrators and decision makers for more convenience.
- Finally the numerical output for the magnitude of the requirements ratings (CCi values) and magnitude of criteria ratings (BNP values) must be converted to a graphical output to make the presentation appealing for the end users and the results much easier to interpret.

4.2.6 Stage 6: Communication

This thesis, conference presentations, and academic publications fulfil the communication requirement of the Design Science process. Additionally, the researcher will hold workshops with the participating organisations to demonstrate the final product. Further research on the use of the software tool is envisaged.

4.3 Chapter Summary

The automated fuzzy tool was presented and designed using the Design Science Research Methodology. DSRM was utilised both as a paradigm and a lens in the tool development. An account of how the tool was designed and constructed at each DSRM stage was outlined in the chapter. The demonstration of the tool was conducted by end users. The output of the tool was validated by showing detailed calculations of intermediate steps. Finally, the fuzzy automated tool was evaluated and showed positive results by stakeholders. In the next chapter the data analysis of the survey results is presented.

CHAPTER FIVE: PRESENTATION OF SURVEY RESULTS AND FINDINGS

5.0 Review of Study Design

The previous chapter outlined the design, development and evaluation of the new automated fuzzy tool for requirements prioritisation. The research design discussed in Chapter 3 is an *Explanatory Sequential* Mixed Methods research study. Figure 5.1 below shows the order of the data collection and analysis in the study. Data was collected and analysed in two phases.

The results and findings of the extent that secure RE processes are implemented in Agile RE practices in industry are presented in Chapter 5 and Chapter 6. Chapter 5 deals with the presentation of the results and findings of phase 1 of the study, namely, the field survey. Chapter 6 deals with the presentation of the results and findings of phase 2, namely, the qualitative study as depicted in Figure 5.1.

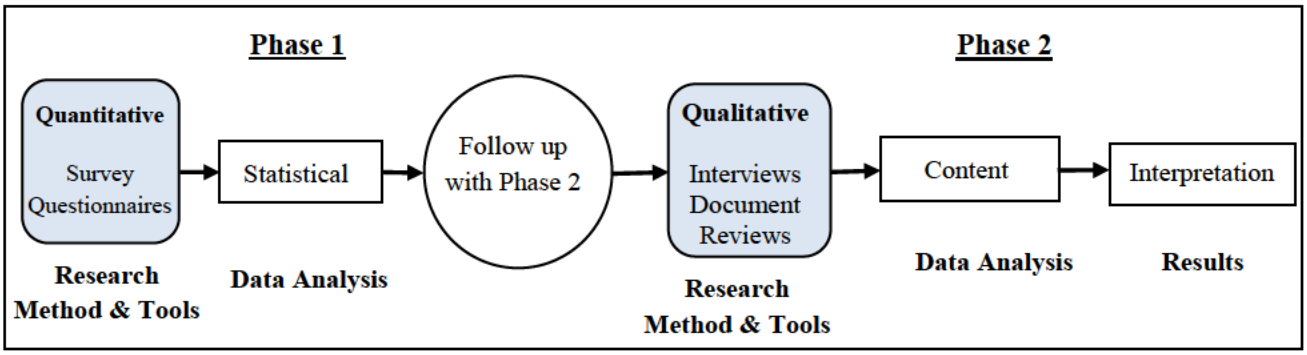


Figure 5.1: Data collection, analysis and interpretation

In phase 1 of the study, the tool used in the quantitative study was the survey questionnaire. Data analysis was statistical. After examining the results of the quantitative study, the qualitative study was undertaken in phase 2, to seek further in-depth explanations and clarifications around requirements engineering practices and security in ASD projects in industry. The research tools employed in the field work were interviews and document reviews. Data was analysed using content analysis. Thereafter, the results of both the quantitative and qualitative studies were interpreted and are presented in Chapter 7.

5.1 Introduction

Chapter 5 presents the survey results on requirements engineering and security practices in ASD and discusses the findings obtained from the survey questionnaire in this study. The questionnaire was distributed to seventeen software development companies. The data collected from the respondents were analysed with SPSS version 24.0. The results are presented using descriptive statistics in the form of tables, graphs and cross tabulations for the quantitative data that was collected. Inferential techniques included the use of chi square test values and correlations; which were interpreted using the p-values. The presentation is ordered according to the sections in the survey questionnaire as follows: reliability statistics and factor analysis; Section A: biographical details, Section B: requirement engineering processes and Section C: requirements engineering practices. This is followed by hypothesis testing and correlation analysis that was conducted on the data. Finally, a regression model is presented.

5.2 The sample

Questionnaires were distributed to a sample of 100 Agile Software Developers from 17 software development companies. There were 78 questionnaires returned to the researcher. The response rate was 78%. Respondents were asked to base their responses on the most recently completed project that they were involved in.

5.3 The research instrument

The objectives of the questionnaire were twofold, namely:

- Evaluate the extent that secure RE approaches are implemented in Agile RE practices in Industry;
- Establish how software engineers analyse client requirements.

The research instrument contained 90 items, with a nominal or an ordinal level of measurement. The questionnaire was divided into 3 sections which measured themes as illustrated below (see Annexure D for copy of survey questionnaire):

(i) Biographical data (A1-A8)

(ii) Requirements Engineering processes in ASD (B1-B43)

(iii)Secure RE practices in ASD projects (C1-C12)

5.4 Reliability statistics

Reliability is measured by taking several assessments on the same subjects. A reliability coefficient of 0.600 or higher for a newly developed construct is considered as "acceptable" (Crow 2006). Table 5.1 reflects the Cronbach's alpha score for all the items that constituted the questionnaire.

		Number of Items	Cronbach's Alpha
B1 - B8	Requirements elicitation process in Agile Software Development	8	0.826
B11 - B15	Degree of difficulty to elicit requirements	5	0.604
B17 - B20	Requirements elaboration	4	0.799
B21 - B23	Analysis of requirements	3	0.807
B25 - B30	Requirements negotiation with the client[A]	6	0.718
B35 - B43	Constraints to secure requirements engineering in Agile[B]	9	0.830
C1 - C12	Secure Agile RE practices in projects[C]	12	0.921

Table 5.1: Reliability Statistics

A: (B25-B30) namely, time-to-market, trade-off between functional and non-functional requirements, cost, conflicting requirements, security requirements and prioritisation of requirements.

B: (B35-B43) namely, large scope, limited budget, limited time, change in requirements, non-secuity risks, limited human resources, limited security knowledge of the team, poor management support for security and lack of interest in security by the customer.

C: (C1-C12) namely, requirements elicitation, Identification of security goals, requirements analysis and modelling, requirements estimation efforts, system for requirements traceability to work products, the trade-off between functional and non-functional requirements, the valuation of assets and resources of the software being developed, requirements inspection to identify potential threats, security risk analysis, security requirements identification, requirements validation methods and system for requirements management.

Table 5.1 shows that the reliability scores exceed the recommended Cronbach's alpha value of 0.700 for all sections besides Section B11-B15 of the newly developed construct. This indicates a degree of acceptable, consistent scoring for these sections of the research instrument. A reliability

score of 0.604 was obtained for Section B11-B15. The researcher attempted to do a reduction of items for this section but this did not reveal significantly stronger results.

5.5 Factor analysis

This section is approached by first presenting the results of the tests required before factor analysis can be implemented and then presenting the results after implementing the factor analysis procedure. Table 5.2 is a summarised table that reflects the results of KMO and Bartlett's Test.

		Kaiser-Meyer-Olkin Measure of Sampling Adequacy	Bartlett's Test of Sphericity		
			Approx. Chi-Square	df	Sig.
B1 - B8	Requirements elicitation process in Agile Software Development	0.820	188.522	28	0.000
B11 - B15	Degree of difficulty to elicit requirements	0.626	50.661	15	0.000
B17 - B20	Requirements elaboration	0.736	118.214	6	0.000
B21 - B23	Analysis of requirements	0.698	78.158	3	0.000
B25 - B30	Requirements negotiation with the client	0.668	92.909	15	0.000
B35 - B43	Constraints to secure requirements engineering in Agile RE	0.710	306.206	36	0.000
C1 - C12	Secure Agile RE practices in projects	0.877	595.556	66	0.000

Table 5.2: KMO and Bartlett's Test

Table 5.2 shows that the Kaiser-Meyer-Olkin statistic is large since the KMO measure of sampling adequacy value is greater than 0.500 and the Bartlett's Test of Sphericity is statistically significant since the sig. value is less than 0.05. This means that all of the conditions are satisfied for factor analysis.

Results after implementing the factor analysis procedure

Before presenting the results of the factor analysis in the form of a rotated component matrix, it is important to note the following (Woolford 2015):

- Principal component analysis was used as the extraction method, and the rotation method was Varimax with Kaiser Normalization. This is an orthogonal rotation method that minimizes the number of variables that have high loadings on each factor. It simplifies the interpretation of the factors.

- Factor analysis/loading show inter-correlations between variables.
- Items of questions that loaded similarly imply measurement along a similar factor. An examination of the content of items loading at or above 0.5 (and using the higher or highest loading in instances where items cross-loaded at greater than this value) effectively measured along the various components.

5.5.1 Factor Analysis Results

The Likert scale items from the survey questionnaire, namely sections, B1-B8, B11-16, B17-20, B21-B23, B26-B30, B35-B43 and C1-C12 were analysed. The results show that variables that constituted the question sections were loaded on 1, 2 or 3 components. The statements that constituted sections B1 – B8, B17 – B20 and B21 – B23, loaded perfectly along a single component. The extraction method used was principal component analysis. This implies that the statements that constituted these sections perfectly measured what it set out to measure.

The results suggest the loading of a question section (construct) along two or more components means that the question section can be divided into finer components (sub-themes). This is explained by the tables below, showing results of the rotated component matrix.

Requirements Engineering Processes (B1 - B8)	**Component**
	1
Objectives of the web application are identified	0.735
All stakeholders are identified	0.660
All viewpoints are established	0.727
Assets of the system are identified	0.699
Security experts are identified	0.710
Non-security goals identified	0.578
Normal requirements are identified	0.587
Non-functional requirements are identified	0.685

Table 5.3: Component Matrix Section B1-B8

Constructs in Table 5.3 perfectly correlated with the construct being measured namely, ‘requirements engineering processes’ measuring Agile RE activities.

Difficulty to elicit requirements (B11 - B16)		Component	
		1	2
B11	When project goals are unclear	0.702	-0.196
B12	When stakeholders priorities differ	0.657	0.364
B13	When people have unspoken assumptions	0.585	0.324
B14	When stakeholders interpret meanings differently	0.719	0.083
B15	When requirements are stated in a way that makes it difficult to verify	0.249	0.473
B16	When the customer is unavailable	-0.111	0.896

Table 5.4: Rotated Component Matrix Section B11-B16

It is observed from Table 5.4 that two components (sub themes) can be identified from the variables that constituted Section B11-B16. This means that respondents identified different trends within the section. The rotation converged in 3 iterations. The sub-theme for the first component (column 1, B11-B14) can be described as 'problems experienced by the technical team to elicit requirements'. B15 and B16 are based on the verification of requirements as this process has an impact on the quality of requirements being elicited. Therefore the sub-theme for the second component (column 2, B15-B16) can be described as 'difficulty verifying requirements'.

It is also observed from Table 5.4 that in component 2, B15 had a score of 0.473 and is widely different from B16 with a score of 0.896. These items are weakly linked to one another and future research is necessary to determine if a requirement that needs to be verified the customer must be available.

Requirements Elaboration (B17 - B20)	Component
	1
Generating Use Cases	0.553
UML activity diagrams	0.860
Class diagrams	0.844
State diagrams	0.881

Table 5.5: Component Matrix Section B17-B20

The concepts that constituted this section perfectly measured 'requirements elaboration methods' used.

Requirements Analysis (B21 - B23)	Component
	1
Structured Requirements Definition	0.849
Object Oriented Analysis	0.822
Structured Analysis and Design	0.885

Table 5.6: Component Matrix Section B21-B23

The statements that constituted this section perfectly measured the 'methods used for requirements analysis'.

Requirements Negotiation (B25 - B30)		Component	
		1	2
B25	Time-to-market	0.158	0.796
B26	Tradeoff between functional and non-functional requirements	0.119	0.872
B27	Cost	0.694	0.043
B28	Conflicting requirements	0.768	0.269
B29	Security requirements	0.695	0.297
B30	Prioritisation of requirements	0.667	0.027

Table 5.7: Rotated Component Matrix Section B25-B30

Table 5.7 shows that the variables that constituted section B25-B30 loaded along 2 components (sub-themes). Separate dimensions were identified for the theoretical construct of 'requirements negotiation'. The rotation converged in 3 iterations. The first sub-theme that can be identified in component 1 (column 1, B27-B30) is factors that 'raise the cost of development' Therefore they can be grouped together. The second sub-theme that can be identified in component 2 (column 2, B25-B26) is 'Time'.

Constraints to secure RE (B35 - B43)		Component		
		1	2	3
B35	Large scope	0.363	0.193	0.554
B36	Limited budget for project	-0.024	0.189	0.884
B37	Limited time to complete	0.565	-0.012	0.684
B38	Change in requirements	0.885	0.104	0.099
B39	Non-security risks in the project	0.729	0.233	0.034
B40	Limited human resources	0.765	0.144	0.333
B41	Limited security knowledge of team	-0.011	0.812	0.334
B42	Poor Management support for security	0.235	0.879	0.094
B43	Lack of interest in security by the customer	0.222	0.810	0.022

Table 5.8: Rotated Component Matrix Section B35-B43

It is noted that the variables that constituted the remaining sections loaded along 3 components (sub-themes). This means that respondents identified three different trends within the section. The rotation converged in 5 iterations. The first sub-theme that can be identified in component 1 (column 1, B38-B40) is related to 'project risks'. The second sub-theme that can be identified in component 2 (column 2, B41-B43) is related to 'knowledge and information'. The third sub-theme that can be identified in component 3 (column 3, B35-B38) is related to 'project scope'.

Secure requirements engineering practices (C1 - C12)	Component	
	1	2
Identification of security goals	0.855	0.237
The trade-off between functional and non-functional requirements	0.535	0.381
The valuation of assets and resources of the software being developed	0.603	0.409
Requirements inspection to identify potential threats	0.688	0.441
Security Risk Analysis	0.892	0.164
Security requirements identification	0.904	0.192
Requirements gathering	0.394	0.699
Requirements analysis and modelling	0.496	0.589
Requirements estimation efforts	0.128	0.841
System for requirements traceability to work products	0.284	0.766
Requirements validation methods	0.465	0.537
System for requirements management (changes, tracking and control of requirements)	0.171	0.758

Table 5.9: Rotated Component Matrix Section C1-C12

It is noted that the variables that constituted section C1-C12 loaded along 2 components (sub-themes). This means that respondents identified two separate underlying constructs within this section. The rotation converged in 3 iterations. The first sub-theme that can be identified in component 1 (column 1) is related to 'security risk assessment practices'. The second sub-theme that can be identified in component 2 (column 2) is related to 'conventional requirements engineering practices'.

5.6 Results: Section A- Biographical Data

In order to understand the characteristics of respondents questions, regarding biographical data were asked. This section presents the results for the biographical data of the study.

5.6.1 Age (in completed years)

Table 5.10 shows the spread of ages in the sample.

	Frequency	Percent
18 - 20	2	2.6
21 - 25	26	33.3
26 - 30	19	24.4
31 - 45	29	37.2
46+	2	2.6
Total	78	100.0

Table 5.10: Spread of Age Groups in Sample

The majority of the respondents were under the age of 30. More than 60% of the responses in the sample population came from respondents who were under the age 30. This is in keeping with global statistics on the age of software developers that show that developers worldwide tend to be younger. In a global survey conducted by Stack Overflow, an online question and answer site for programmers, the average age of developers reported were under age 30 (Heath 2016).

5.6.2 Gender

The table below describes the overall gender distribution.

	Frequency	Percent
Male	63	80.8
Female	15	19.2
Total	78	100.0

Table 5.11: Gender distribution

Overall, the ratio of males to females is approximately 4:1 (80.8%: 19.2%). This percentage is in keeping with global trends that suggest that women are vastly under-represented in the software development industry, for example at Google women make up 17% of technical staff and at Facebook women make up 15% (Chern 2017).

5.6.3 Agile Roles

Table 5.12 indicates the various roles that respondents held in the Agile Software Development team.

	Frequency	Percent
Project Manager	7	9.0
Team Leader	12	15.4
Team member	52	66.7
Business Analyst	6	7.7
Product owner	1	1.3
Total	78	100.0

Table 5.12: Agile Roles

A diverse range of survey participants are reflected. Two thirds of the respondents (66.7%) were team members, 15.4% were team leaders and with 1.3% of product owners also being involved in the sample population. Team members are the largest represented group and include programmers, testers, architects, user interface designers and quality assurance members (Kavitha & Thomas 2011). The views and beliefs of members representing the various roles in an ASD team represented in the sample were important to gain a broad perspective of Agile RE practices.

5.6.4 Education Levels

Figure 5.2 indicates the education levels of the respondents.

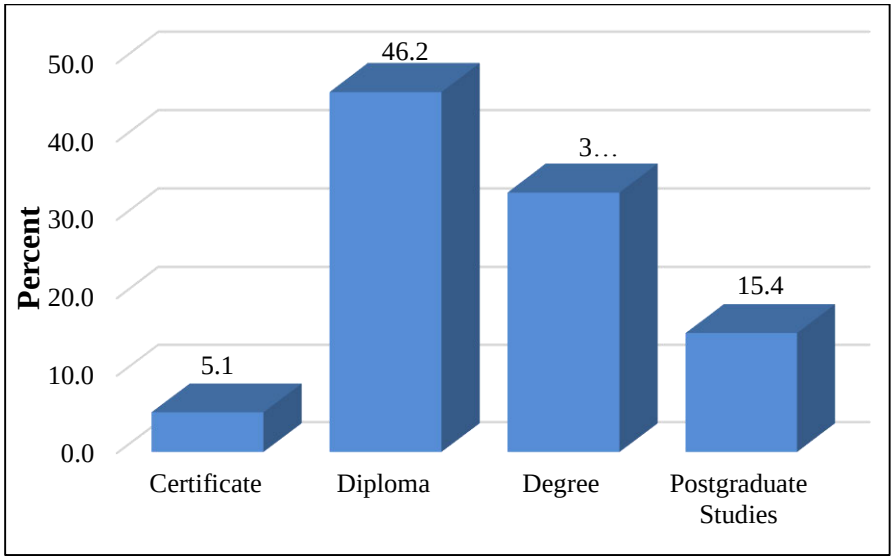


Figure 5.2: Education level of respondents

All of the respondents had a post-school qualification, with the majority (79.5%) having a university degree or diploma. The reason for employment of highly qualified individuals is that

ASD requires highly skilled individuals (De Lucia & Qusef 2010). The Global Information Technology Report (2016) ranked South Africa in 65th position for technology readiness. The advanced education levels reported in this study is much better than IT countries such as China with a slightly better global ranking (59th) in terms of technology readiness (World Economic Forum 2016). In a survey study conducted in Chinese organisations by Elahi et al. (2011) to gain an understanding of security RE practices in industry showed that 287 of 374 (77%) participants had post-school education.

5.6.5 Nature of employment

The nature of the employment is reflected below.

	Frequency	Percent
Permanent	73	93.6
Contract	5	6.4
Total	78	100.0

Table 5.13: Nature of employment

Table 5.13 shows that 93.6% of the respondents are employed in a permanent capacity. Permanent employees are more reliable and tend to have greater commitment to an organisation (Nortje 2013).

5.6.6 Years of experience

Figure 5.3 indicates the length of service of the respondents.

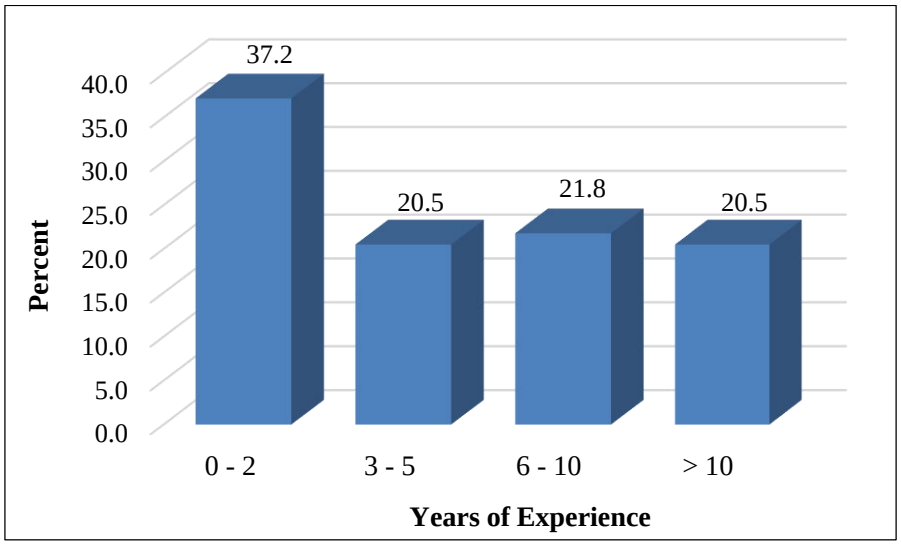


Figure 5.3: Years of experience

Approximately 42% of the respondents had been in employment for more than 5 years. The percentage that represented the 3-5 year work experience category was 20.5%. Figure 5.3 shows that more than 60% of the responses were from employees who had 3 or more years of experience. Agile Software Development was first introduced in 2001 and is in use for the last 16 years in industry (Kavitha & Thomas 2011). This indicates that the responses are from professionals with sufficient experience in ASD to provide relevant feedback in the survey.

5.6.7 Application security training received

Table 5.14 shows the type of application security training received in the last 12 months.

TYPE	Frequency	Percent
Security requirements/modeling related	2	2.6
Code related	22	28.2
General application threats and vulnerabilities	7	9.0
Security metrics tools	1	1.3
No, I did not receive security training in the last 12 months	46	59.0
Total	**78**	**100.0**

Table 5.14: Type of application security received

The majority (59%) of the respondents did not receive security training in the last 12 months. This is an alarmingly high percentage and was flagged for in-depth explanations in phase 2 of the data collection, namely, in the qualitative study. Only 28.2% of respondents received code related security training. This percentage is aligned to the Elahi et al. (2011) study that showed 27% of the participants received security training in employment. The security training the teams are receiving is insufficient. Ge et al. (2007) recommend that all participants of a team including the customer receive security training to ensure that the new system satisfy critical security requirements. They suggest the following training at a minimum: understanding of common vulnerabilities, writing of security stories, awareness of poor programming practices in terms of security and security testing.

5.6.8 Value of application security training received

Figure 5.4 indicates the value of application security training received by developers.

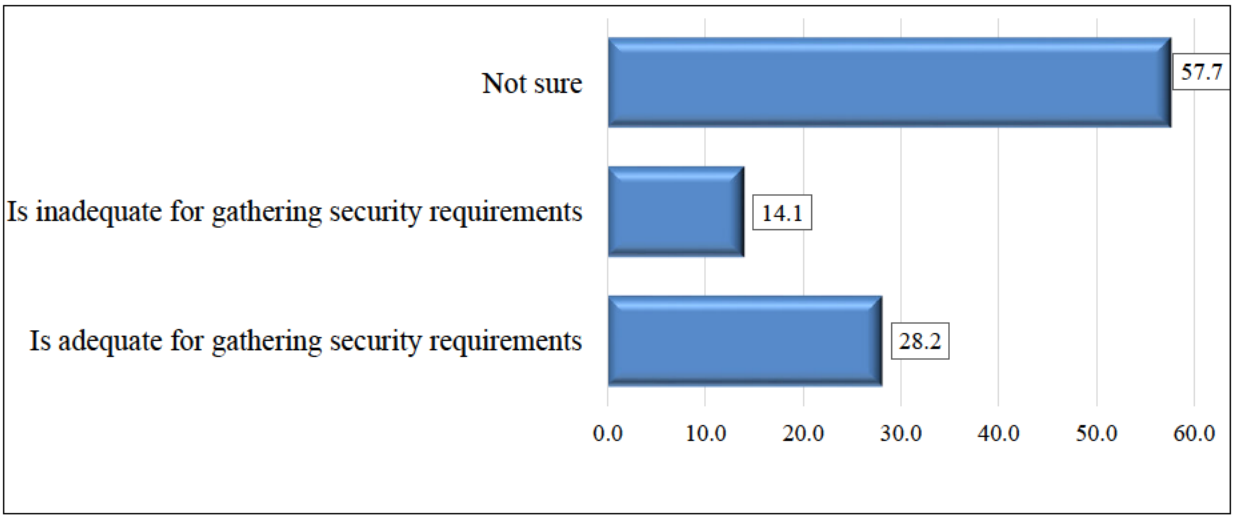


Figure 5.4: Value of application security training

According to Figure 5.4 the highest percentage, 57.7%, is in the "Not sure" category. Table 5.14 indicates that the majority (59%) of the respondents did not receive security training in the last 12 months. This accounts for why the majority responded in the "Not sure" category about the value of application security received by developers. Those respondents that received security training, 14.1%, believed that security training received, was inadequate for gathering security requirements.

5.7 Section analysis

The section that follows analyses the scoring patterns of the respondents per variable per section. The results are first presented using summarised percentages for the variables that constitute each section. Results are then further analysed according to the importance of the statements.

5.7.1 Results: Section B- Requirements Engineering

This section presents the results for Agile RE in practice. It comprises of the following sub-sections: Agile RE Elicitation practices, Agile RE Elicitation Techniques, Agile RE Elicitation Challenges, Agile RE Elaboration of requirements and Agile RE Analysis of requirements

5.7.1.1 Agile RE Elicitation practices

This section deals with the requirements elicitation practices in Agile Software Development. The results presented are for questions B1-B8 in the survey questionnaire. The table below summarises the scoring patterns.

		Disagree	Not Sure	Agree	Chi Square p-value
Objectives of the web application are identified	**B1**	12.82	10.26	76.92	0.000
All stakeholders are identified	**B2**	10.26	10.26	79.49	0.000
All viewpoints are established	**B3**	12.82	14.10	73.08	0.000
Assets of the system are identified	**B4**	11.54	17.95	70.51	0.000
Security experts are identified	**B5**	30.77	34.62	34.62	0.007
Non-security goals identified	**B6**	11.54	35.90	52.56	0.000
Normal requirements are identified	**B7**	1.28	8.97	89.74	0.000
Non-functional requirements are identified	**B8**	10.26	17.95	71.79	0.000

Table 5.15: Requirements Engineering Processes

In Table 5.15 it can be observed that most statements show significantly higher levels of agreement (column 4 and column 5) whilst other levels of agreement are lower but still greater than levels of disagreement. A Chi Square test was completed to determine whether the scoring patterns per statement were statistically significantly different per question option. The p-values obtained were less than 0.05 (the level of significance) in all cases. This implies that the differences between the way respondents scored (agree, not sure, disagree) were significant.

Majority of the respondents were in agreement that standard requirement engineering processes are practiced in Agile RE, namely, B1(76%), B2(79%), B3(73%), B4(70%), B6(52%), B7(89%) and B8(71%). This was very encouraging considering that ASD processes have flexiblity. However, on the question of the identification of security experts, B5, there were significantly more respondents who were not sure. Grouping the "Not sure" category with the "Disagreement" category implies that 65.4% of respondents disagree or are not sure that security experts are identified. This high percentage raises questions around security experts and further clarity will be sought in the indepth qualitative study.

5.7.1.2 Agile RE Elicitation Techniques

The results below correspond to question B9 of the survey questionnaire. Figure 5.5 illustrates the elicitation techniques used for the functional requirements. The responses have been ranked.

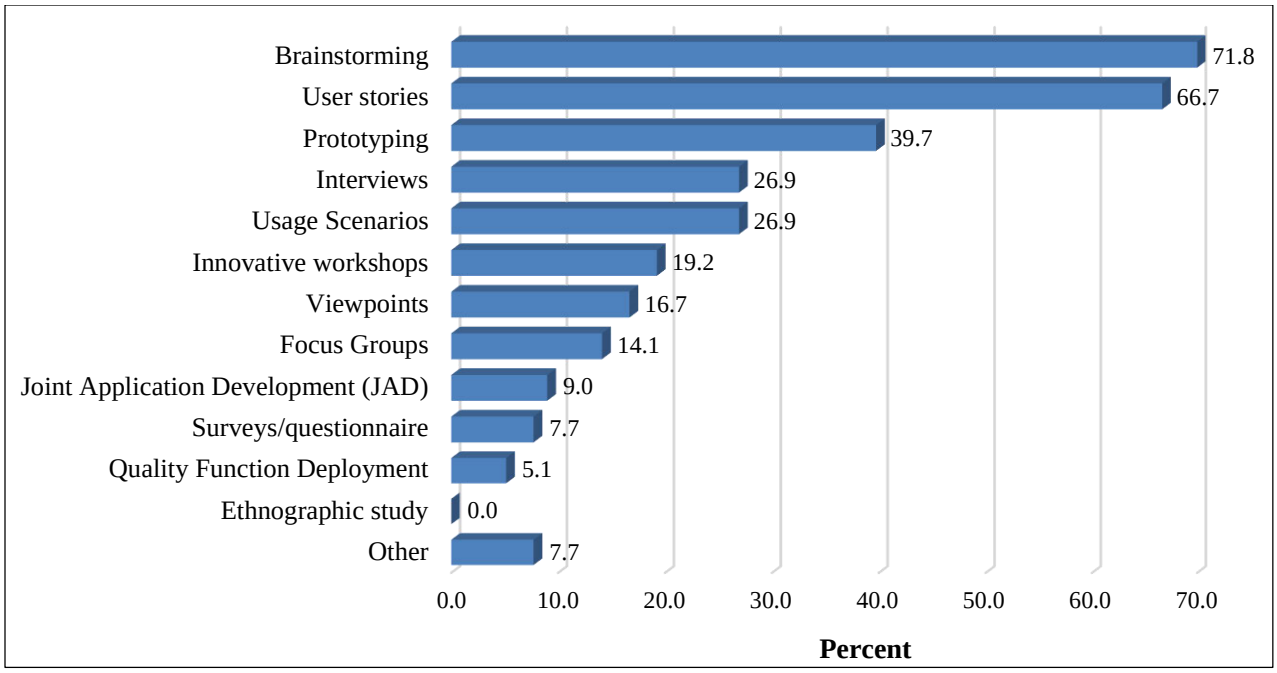


Figure 5.5: Elicitation Techniques for Functional Requirements

In this question respondents could choose more than one option, which is why the total does not sum to 100%. The predominant elicitation technique that is used for functional requirements in ASD practice is Brainstorming (71.8%) and User stories (66.7%) was ranked as the second most common approach used. These results concur with the results obtained by Kassab (2014) who conducted a web based survey with 247 respondents from 23 different countries on the current state of RE practice for ASD practitioners. In the survey conducted by Kassab (2014) Brainstorming followed by User Stories were the two most popular elicitation techniques.

The results in Figure 5.6 correspond to question B10 of the survey questionnaire which the elicitation techniques used for non-functional requirements.

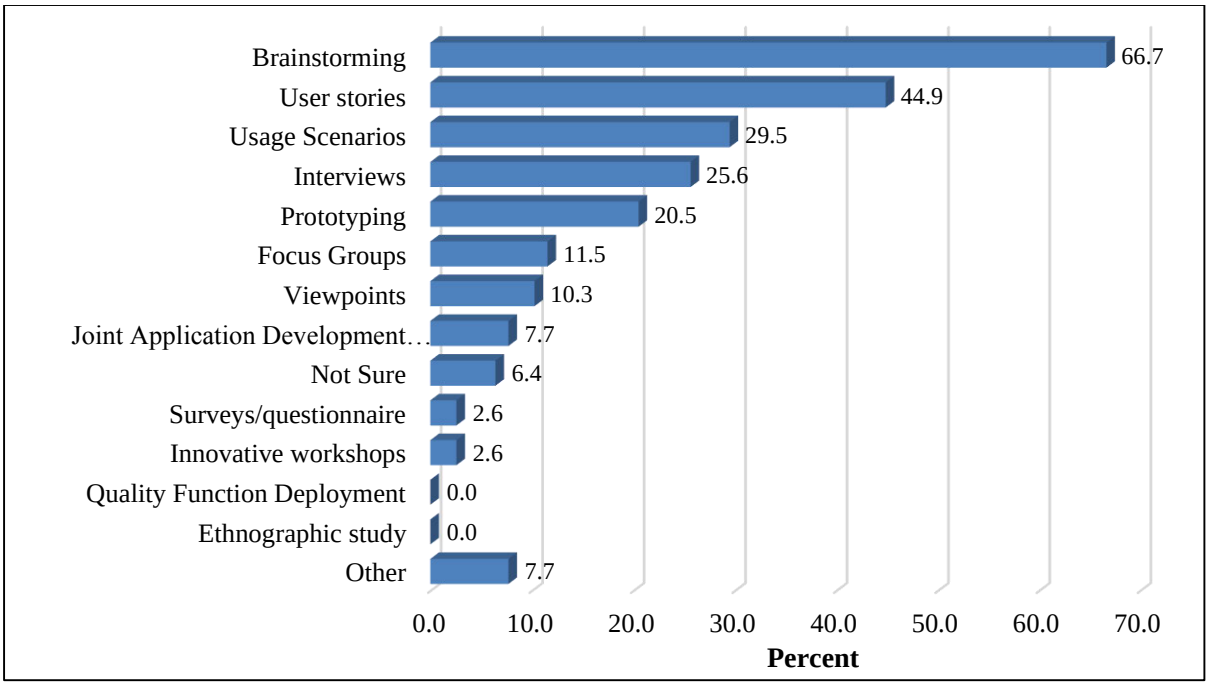


Figure 5.6: Elicitation Techniques for Non-Functional Requirements

The predominant elicitation techniques that are commonly used for non-functional requirements is Brainstorming (66.7%) and User stories (44.9%). This is not surprising, considering that these techniques were predominantly used in other ASD projects as shown by Kassab (2014) study.

5.7.1.3 Agile RE Elicitation challenges

		Not a Factor	Mildly Difficult	Somewhat Difficult	Difficult	Very Difficult	Most Difficult
When project goals are unclear	**B11**	0.00	1.28	6.41	16.67	30.77	44.87
When stakeholders priorities differ	**B12**	0.00	0.00	10.26	17.95	48.72	23.08
When people have unspoken assumptions	**B13**	0.00	2.60	14.29	20.78	36.36	25.97
When stakeholders interpret meanings differently	**B14**	0.00	0.00	11.54	20.51	34.62	33.33
When requirements are stated in a way that makes it difficult to verify	**B15**	1.28	1.28	5.13	16.67	38.46	37.18
When the customer is unavailable	**B16**	0.00	1.28	7.69	20.51	21.79	48.72

Table 5.16: Degree of difficulty to Elicit Requirements

When gathering user requirements several challenges are experienced by requirements engineers. Table 5.16 presents the results for question B11-B16 of the survey questionnaire. The section (B11-B16) deals with the difficulty experienced with eliciting requirements. The majority of the

respondents (more than 67%) rated questions B11-B16 in the 'very difficult' and 'most difficult' category. This implies that software developers experienced difficulty eliciting requirements in all categories of Table 5.16. By summing the results in the very difficult' and 'most difficult' columns of Table 5.16 it can be concluded that the biggest problem posed to software developers related to eliciting requirements is 'when requirements are stated in a way that makes it difficult to verify' (B15) and 'when project goals are unclear' (B11). The percentages are 75.7% and 71.8% respectively.

5.7.1.4 Agile RE Elaboration of requirements

The table below presents the results for questions B17-B20 of the survey questionnaire. This section (B17-B20) deals with the techniques used by software developers to elaborate requirements.

		Disagree	Not Sure	Agree
Generating Use Cases	**B17**	14.10	8.97	76.93
UML activity diagrams	**B18**	26.92	16.67	56.41
Class diagrams	**B19**	29.49	16.67	53.85
State diagrams	**B20**	37.18	25.64	37.18

Table 5.17: Elaboration of Requirements

The use case diagram was the most popular method among software developers to elaborate requirements with 76.93% 'agreeing'. Use Case is a popular choice in ASD because it provides developers with a high level view of the features developers are constructing.

5.7.1.5 Agile RE Analysis of requirements

Table 5.18 presents the results for questions B21-B24. This section (B21-B24) deals with the methods of analysing requirements.

		Disagree	Not Sure	Agree
Structured Requirements Definition	**B21**	15.39	20.51	64.10
Object Oriented Analysis	**B22**	64.10	17.96	17.94
Structured Analysis and Design	**B23**	87.18	3.85	8.97
No methodology	**B24**	89.74	1.27	8.99

Table 5.18: Analysis of requirements

Requirements analysis is a process used to check the requirements for consistency, completeness and feasibility (Kassab 2014). Structured requirements definition was the most preferred method

for requirements analysis with 64.11% of the respondents in agreement with its use. Traditional techniques such as structured analysis is in the decline.

Table 5.19 presents the results for questions B25-B27. This section (B25-B27) deals with requirements negotiation.

		Not a Factor	Very Weak Factor	Weak Factor	Mildly Strong Factor	Strong Factor	Very Strong Factor
Time-to-market	**B25**	2.56	2.56	8.97	20.51	24.36	41.03
Trade-off between functional and non-functional requirements	**B26**	5.13	3.85	7.69	39.74	28.21	15.38
Cost	**B27**	2.56	2.56	6.41	17.95	28.21	42.31
Conflicting requirements	**B28**	0.00	5.19	7.79	27.27	32.47	27.27
Security requirements	**B29**	7.69	3.85	14.10	26.92	24.36	23.08
Prioritisation of requirements	**B30**	0.00	1.28	2.56	14.10	41.03	41.03

Table 5.19: Requirements negotiation

By combining the percentages in 'Strong Factor' and 'Very Strong Factor' columns it was concluded that the strongest factor for requirements negotiation was the 'prioritisation of requirements' (B30), with a total of 82.06% obtained, followed by 'cost' (B27), with a total of 70.52 % obtained as indicated in Table 5.19. The inference here is that customer involvement in the prioritisation of requirements plays an important role.

5.7.1.6 Agile RE security requirements practices

Table 5.20 presents the results for question B32. The table below shows the percentages for the methods used for *security requirements specification*.

	Frequency	Percent
Security Specifications Language (eg. CLASP, Secure TROPOS, etc.)	10	12.8
Semi-Formal Notations (UML, class, sequence diagram)	13	16.7
Informal language (User stories/scenarios)	55	70.5
Total	78	100.0

Table 5.20: Requirements specification

Table 5.20 shows that the majority of the respondents (70.5%) indicated that security requirements are specified using an informal language such as user stories or scenarios of their company. This means that when security requirements are elicited they appear in a natural language.

Table 5.21 presents the results for question B33 and describes the teams that are responsible for identifying security requirements.

	Frequency	Percent	Cumulative Percent
Development team	52	66.7	66.7
Dedicated security team	14	17.9	84.6
Not sure	12	15.4	100.0
Total	**78**	**100.0**	

Table 5.21: Security requirements identification

Table 5.21 shows that 66.7% of respondents indicated that the software development team was responsible for identifying security requirements. This is an interesting finding as the development team excludes project managers, requirements engineers and/or business analysts. These excluded individuals do not play a major role during coding and will not be involved with security. The natural conclusion here is that ultimately, security is left to be built into the system during coding. The findings are not dissimilar to the Elahi et al. (2011) who reported that security was left to be achieved by the implementation team. The researcher used phase 2 of the data collection (qualitative) to seek further explanations on whether the approach to security was "penetrate and patch", "secure software development at coding" or "secure software development during RE".

Table 5.22 presents the results for question B34 that requests whether *dedicated security expert/s brought into the team to validate of requirements.* The table shows percentages of groups responsible for validating requirements in a project.

	Frequency	Percent	Cumulative Percent
Yes	16	20.5	20.5
No-developers validate all requirements	62	79.5	100.0
Total	78	100.0	

Table 5.22: Requirements validation

Table 5.22 illustrates that the majority of respondents (79.5%) indicated that all validation of requirements take place by developers. The inference here is that there is no dedicated security team operating at most companies. The role of the customer in the validation of requirements was explored further in phase 2 of the data collection (qualitative study).

Table 5.23 presents the results for questions B35-B43. This section (B35-B43) deals with what software engineers perceive as *constraints to secure requirements engineering* in practice.

		Not a Factor	Very Weak Factor	Weak Factor	Mildly Strong Factor	Strong Factor	Very Strong Factor
Large scope	**B35**	8.97	6.41	8.97	34.62	32.05	8.97
Limited budget for project	**B36**	6.41	3.85	14.10	32.05	29.49	14.10
Limited time to complete	**B37**	3.85	0.00	7.69	21.79	35.90	30.77
Change in requirements	**B38**	5.13	1.28	11.54	19.23	38.46	24.36
Non-security risks in the project	**B39**	11.54	8.97	12.82	38.46	21.79	6.41
Limited human resources	**B40**	7.69	5.13	8.97	37.18	23.08	17.95
Limited security knowledge of team	**B41**	5.13	3.85	10.26	32.05	29.49	19.23
Poor Management support for security	**B42**	7.69	3.85	11.54	30.77	33.33	12.82
Lack of interest in security by the customer	**B43**	10.26	5.13	11.54	37.18	20.51	15.38

Table 5.23: Constraints to secure requirements engineering

In Table 5.23, summing the columns 'Strong Factor' and 'Very Strong Factor', 62.82% it can be concluded that 'change in requirements' (B38) was the biggest constraint to secure requirements engineering. Similarly 'limited time to complete the project' (B37) was concluded to be the second biggest factor hindering secure requirements engineering with a percentage of 61.67 % respondents in agreement. In phase 2 of the data collection these results were triangulated.

5.7.2 Results: Section C- Secure Requirements Engineering practices

Table 5.24 shows the mean scores in the range [0-5] for ASD requirements engineering practices in the 17 companies. Agile RE practices were rated by team members of a project based on 12 requirements engineering processes as indicated in Section C of the survey questionnaire. Team members were required to rate (0-5) secure requirements engineering practices based on the most recently completed project that the group was assigned. Responses were obtained from one project team per company. Therefore, the project numbers represent the company. Hence, the words "project" and "company" are used interchangeably. The results are presented for 17 companies.

Ranking	1	2	3	4	5	6	7	8	9
Project	**12**	**5**	**8**	**4**	**3**	**6**	**11**	**7**	**2**
Score	2.03	2.71	2.77	2.81	2.99	3.15	3.19	3.21	3.23
Ranking	10	11	12	13	14	15	16	17	
Project	**1**	**14**	**9**	**10**	**16**	**13**	**15**	**17**	
Score	3.25	3.27	3.47	3.56	3.57	3.77	3.83	4.10	

Table 5.24: Mean scores for Agile RE practices at 17 companies

The rankings of the projects are given from low ("not well established secure RE") to high ("well established secure RE"). The project from company 12 was rated the lowest (2.03) for its requirements engineering practices. This is a small software development company with only one development team functioning. The project from company 17 was rated the highest (4.10) for their requirements engineering practices. The profile of this company is that of a large international software development company operating in a multi-project and multi-team environment. This company had individual teams for security, architecture, RE and research functioning independently from the multiple development teams.

Table 5.25 shows means scores for projects on constraints experienced to secure requirements engineering.

Ranking	1	2	3	4	5	6	7	8	9
Project	**14**	**10**	**9**	**11**	**16**	**3**	**8**	**17**	**2**
Score	1.76	2.26	2.56	2.96	3.19	3.24	3.31	3.37	3.38
Ranking	10	11	12	13	14	15	16	17	
Project	**6**	**4**	**13**	**12**	**7**	**1**	**15**	**5**	
Score	3.42	3.43	3.44	3.52	3.53	3.56	3.78	3.83	

Table 5.25: Mean scores for constraints for Secure RE at 17 companies

Project 14 experienced the least problems (1.76) for secure requirements engineering whilst Project 5 experienced the most problems (3.83) for secure requirements engineering. This correlates with the requirements engineering practices in Table 5.23 since Project 14 (3.57) was rated much higher in terms of requirements engineering practices compared to Project 5 (2.71).

5.8 Hypothesis testing

A statement of statistical significance is a traditional approach to reporting a result. A p-value is generated from a test statistic. When the results of a Pearson's Chi-square statistical test has a p-value that is less than 0.05 then we conclude that the result is statistically significant (Wellman et al. 2005). Table 5.26 gives the results of Chi-square tests showing significant results. These values are highlighted with an *. A second Chi-square test was performed to determine whether there was a statistically significant relationship between the variables (rows vs columns). "The null hypothesis states that there is no association between the two. The alternate hypothesis indicates that there is an association. All values less than 0.05 imply that the distributions are skewed in one direction" (Wellman et al. 2005). Table 5.26 shows the results of the Pearson Chi-square tests that helped the researcher discover trends and relationships in secure RE.

Pearson Chi-square tests		What is your current role in the Agile Software Development team?	What type of qualification do you have?	Employment type	Indicate your age group?	Gender	What type of application security training did you receive in the last 12 months?	The value of application security training received by developers
Objectives of the web application are identified	Chi-square	11.308	7.047	1.636	27.929	2.891	9.778	9.773
	df	16	12	4	16	4	16	8
	Sig.	0.79	0.854	0.802	.032*	0.576	0.878	0.281
When project goals are unclear	Chi-square	21.010	22.107	1.873	20.313	1.554	8.922	13.000
	df	16	12	4	16	4	16	8
	Sig.	0.178	.036*	0.759	0.206	0.817	0.917	0.112
When stakeholders priorities differ	Chi-square	16.681	5.348	2.200	14.764	1.865	21.558	16.426
	df	12	9	3	12	3	12	6
	Sig.	0.162	0.803	0.532	0.255	0.601	.043*	.012*
Security requirements in projects are generated	Chi-square	10.205	1.691	17.808	25.272	7.548	11.545	10.741
	df	12	9	3	12	3	12	6
	Sig.	0.598	0.995	.000*	.014*	0.056	0.483	0.097
Please indicate how security requirements are specified	Chi-square	10.028	3.795	0.334	14.993	3.930	18.493	2.171
	df	8	6	2	8	2	8	4
	Sig.	0.263	0.704	0.846	0.059	0.14	.018*	0.704
In the requirements engineering phase of projects security requirements are identified by	Chi-square	19.426	2.373	1.175	4.332	5.966	18.782	6.124
	df	8	6	2	8	2	8	4
	Sig.	.013*	0.882	0.556	0.826	0.051	.016*	0.19
	Chi-square	9.997	1.022	1.244	1.720	0.431	13.392	3.734

Is the dedicated security expert/s brought into the team to validate of requirements?	df	4	3	1	4	1	4	2
	Sig.	.040*	0.796	0.265	0.787	0.511	.010*	0.155
Limited security knowledge of team	Chi-square	14.789	17.602	3.557	16.503	4.910	46.224	15.331
	df	20	15	5	20	5	20	10
	Sig.	0.788	0.284	0.615	0.685	0.427	.001*	0.12
Identification of security goals	Chi-square	12.721	16.745	4.072	9.541	3.488	26.506	7.250
	df	16	12	4	16	4	16	8
	Sig.	0.693	0.159	0.396	0.889	0.48	.047*	0.51
Security Risk Analysis	Chi-square	29.867	24.196	6.203	7.962	6.893	25.836	4.331
	df	16	12	4	16	4	16	8
	Sig.	.019*	.019*	0.185	0.95	0.142	0.056	0.826
Security requirements identification	Chi-square	9.979	21.762	7.516	5.713	5.919	19.555	3.527
	df	16	12	4	16	4	16	8
	Sig.	0.868	.040*	0.111	0.991	0.205	0.241	0.897

Table 5.26 : Pearson Chi Square Test Results

There were many significant relationships to report as indicated by an asterisk (*****) in Table 5.26. The results are shown as cross tabulations between rows and columns and a few relevant hypothesis tests are discussed below:

The p-value for the cross tabulation between "What is your current role in the Agile Software Development team?" and "In the requirements engineering phase of projects security requirements are identified by" is 0.013 (p-value < 0.05). This means that there is a relationship between the variables. That is, the current role played in ASD will play a role in terms of identification of security requirements in a project. Here it is clear that not all respondents can identify security requirements.

There is also a significant relationship between "What type of application security training did you receive in the last 12 months?" and "Limited security knowledge of team" as the p-value is 0.001(p-value < 0.05). This relationship suggests that for secure software development, security training of personnel is necessary.

Another noteworthy trend is the relation between "What type of application security training did you receive in the last 12 months?" and "Identification of security goals". The table indicates a p-value of 0.047 (p-value < 0.05). This means that there is a relationship between the variables. This

relationship suggests that the type of application security training received in the last 12 months by the respondent will play a role in the identification of security goals. This relationship is important to secure RE practices. It suggests that those who did not receive security training will most likely not identify security goals.

Finally, another important trend for secure Agile RE is revealed in the p-value for the cross tabulation between "What is your current role in the Agile Software Development team?" and "Security Risk Analysis" is 0.019 (p-value < 0.05). This relationship suggests that the current role of the respondent plays a significant role on whether security risk analysis is conducted. Project managers and team leaders by the very nature of their roles will be more likely to be motivated to conduct security risk analysis as a secure RE practice.

5.9 Correlation analysis

Bivariate correlation was also performed on the (ordinal) data in this study. The results indicate the following patterns (Wellman et al. 2005).

- "Positive values indicate a directly proportional relationship between the variables
- Negative values imply an inverse relationship. That is, the variables have an opposite effect on each other".

The results of correlation tests are shown in Table 5.28. All significant relationships are indicated by ***** or ****.** There are too many to discuss and only those that are related to the study are discussed below:

The correlation value between "The tradeoff between functional and non-functional requirements" and "Identification of security goals" is 0.499^{**}. This is a directly related proportionality. Respondents indicated that the more there is trade-off between functional and non-functional requirements, the more likely there is to be the identification of security goals as an RE practice.

The Chi-square test shows significant correlation value between "The valuation of assets and resources of the software being developed" and "Security requirements". The correlation coefficient is 0.345^{**}. Developers who perform valuation of assets and resources of the software

being developed are more likely to identify security requirements. Given the relationship it is surprising that the correlation coefficient is not stronger.

An example of negative correlation is the relationship between "Requirements inspection to identify potential threats" and "Lack of interest in security by the customer". The correlation coefficient is -0.357**. The negative value implies an inverse relationship. That is, the variables have an opposite effect on each other. The more requirements are inspected by the customer to identify potential threats the less likely there will be a lack of interest in security by the customer. This relationship has important implications for secure RE.

Another significant correlation relationship is between "Security Risk Analysis" and "The valuation of assets and resources of the software being developed". The correlation value is 0.521**. Respondents who consider security risk analysis are more likely to conduct the valuation of assets and resources of the software being developed. This is because risk analysis includes valuation of assets. Table 5.27 shows that only 23.1% and 6.4% of respondents have good and excellent risk analysis processes. This means in this study only 29.5% of participants will be more likely to conduct a valuation of assets and resources in their projects. This result is concerning for secure Agile RE as security risk analysis practices are not widespread, implying that practitioners are producing insecure software. A confirmation of this was sought in phase 2 of the data collection.

	Frequency	Percent	Cumulative Percent
Very Weak	11	14.1	14.1
Weak	19	24.4	38.5
Satisfactory	25	32.1	70.5
Good	18	23.1	93.6
Excellent	5	6.4	100.0
Total	78	100.0	

Table 5.27: Security Risk Analysis Practices

In the final analysis, the correlation value between "Requirements inspection to identify potential threats" and "Security experts are identified" is 0.522**. This is a directly related proportionality. Respondents who are more likely to inspect requirements to identify potential threats are more likely to identify security experts.It can be concluded from these relationships, direct or inverse,

that secure requirements engineering is likely and the outcome is dependent on the effort of ensuring secure RE practices are taking place by stakeholders.

			Assets of the system are identified	Security experts are identified	When project goals are unclear	Tradeoff between functional and non-functional requirement	Security require ment	Prioritisation of requirements	Limited security knowledge of team	Poor Manage support for security	Lack of interest in security by the customer	Requirements gathering	Identification of security goals	The valuation of assets and resources of the software being developed	Requirements inspection to identify potential threats	Security Risk Analysis	Security requirements identification
Spearman's rho	Requirements estimation efforts	Correlation Coefficient	.308**	0.143	-0.020	0.143	0.106	-0.090	0 174	0.140	-0.014	.530**	.294**				
		Sig. (2-tailed)	0.006	0.212	0.862	0.212	0.357	0.436	0 128	0.222	0.903	0.000	0.009				
		N	78	78	78	78	78	78	78	78	78	78	78				
	System for requirements Traceability to work products	Correlation Coefficient	.367**	0.143	0.044	.240*	0.119	0.126	0.083	0.026	-0.193	.519**	.404**				
		Sig. (2-tailed)	0.001	0.211	0.701	0.035	0.298	0.271	0.471	0.819	0.091	0.000	0.000				
		N	78	78	78	78	78	78	78	78	78	78	78				
	The trade-off between functional and non-functional requirements	Correlation Coefficient	.274*	.250*	0.088	0.217	0.159	0.007	0 126	0.190	0.163	.435**	.499**				
		Sig. (2-tailed)	0.015	0.028	0.442	0.057	0.164	0.949	0 270	0.096	0.153	0.000	0.000				
		N	78	78	78	78	78	78	78	78	78	78	78				
	The valuation of assets and resources of the software being developed	Correlation Coefficient	0.210	.239*	0.098	0.203	.345**	0.098	0 132	0.056	0.010	.393**	.495**	1.000			
		Sig. (2-tailed)	0.065	0.035	0.395	0.075	0.002	0.394	0 250	0.628	0.932	0.000	0.000				
		N	78	78	78	78	78	78	78	78	78	78	78	78			
	Requirements inspection to identify potential threats	Correlation Coefficient	.323**	.522**	0.133	0.105	.429**	0.189	-0.012	-0.086	-.357**	.572**	.686**	.571**	1.000		
		Sig. (2-tailed)	0.004	0.000	0.246	0.360	0.000	0.097	0 915	0.455	0.001	0.000	0.000	0.000			
		N	78	78	78	78	78	78	78	78	78	78	78	78	78		
	Security Risk Analysis	Correlation Coefficient	.308**	.559**	0.139	0.161	.423**	0.035	0.046	0.061	-0.157	.415**	.734**	.521**	.679**	1.000	
		Sig. (2-tailed)	0.006	0.000	0.224	0.159	0.000	0.763	0.691	0.594	0.169	0.000	0.000	0.000	0.000		
		N	78	78	78	78	78	78	78	78	78	78	78	78	78	78	
	Security requirements identification	Correlation Coefficient	.299**	.542**	0.141	.290*	.401**	0.089	0.080	0.154	-0.044	.432**	.754**	.576**	.631**	.884**	1.000
		Sig. (2-tailed)	0.008	0.000	0.218	0.010	0.000	0.440	0.487	0.177	0.705	0.000	0.000	0.000	0.000	0.000	
		N	78	78	78	78	78	78	78	78	78	78	78	78	78	78	78

* Correlation is significant at the 0.05 level (2-tailed) and ** Correlation is significant at the 0.01 level (2-tailed).

Table 5.28: Correlations between Variables in Agile RE

5.10 Regression models

Two regression models have been built for secure requirements engineering practices in Agile Software Development. The first model took into consideration impacting factors obtained from the research on secure requirements engineering practices. The regression model in Section 5.10.2 with fewer impacting factors is the researcher's contribution to the body of knowledge on secure requirements engineering in Agile RE.

5.10.1 Regression Model with 11 Impacting Factors

A regression model is a statistical linear model that is used to analyse the relationship between dependent and independent variables. It is a mathematical way of determining which factors matters the most, which factors we can ignore and how these factors interact with each other (Harvard Business School 2017). The aim of the study is to determine the impact of ASD requirements engineering practices on the security of the ASD product. Therefore, it is critical to assess which RE activities impact secure Agile RE the most.

In this study the dependent variable is *secure Agile RE practices* as this is what the researcher is trying to understand. The independent variables are the factors that the researcher suspects will have an impact on the dependent variable. In this study there are 11 independent variables, namely, *requirements elicitation; Identification of security goals; requirements analysis and modelling; requirements estimation efforts; system for requirements traceability to work products; the trade-off between functional and non-functional requirements; the valuation of assets and resources of the software being developed; requirements inspection to identify potential threats; security risk analysis; security requirements identification and requirements validation methods.*

Table 5.29 depicts a summary model that indicates whether independent variables (predictors) predict the dependent variables. An explanation of the Table 5.29 is given in points (a)-(d):

Model Summary				
[b]Model	[c]R	[d]R Square	Adjusted R Square	Std. Error of the Estimate
1	.996[a]	.992	.990	.07649
a. Predictors: (Constant), Requirements validation methods, The trade-off between functional and non-functional requirements, Requirements estimation efforts, Security Risk Analysis, Requirements analysis and modelling, The valuation of assets and resources of the software being developed, System for requirements traceability to work products, Requirements elicitation, Requirements inspection to identify potential threats, Identification of security goals, Security requirements identification				

Table 5.29: Model Summary

a. Predicted values of dependent variables.

b. This column shows the number of the model.

c. R is the square root of R-Squared and is the correlation between the observed and predicted values of the dependent variable.

d. This is the proportion of variance in the dependent variable (*secure RE practice*) which can be explained by the independent variables (*requirements elicitation; Identification of security goals; requirements analysis and modelling; requirements estimation efforts; system for requirements traceability to work products; the trade-off between functional and non-functional requirements; the valuation of assets and resources of the software being developed; requirements inspection to identify potential threats; security risk analysis; security requirements identification and requirements validation methods).* The R Square value indicates that 99% of the variation in the dependent variable can be explained by changes to the independent variables (UCLA Institute for Digital Research and Education 2017).

Table 5.30 indicates if independent variables can reliably predict dependent variables. The independent variables are listed in point (b) of Table 5.30.

ANOVA[a]						
Model		**Sum of Squares**	**df**	**Mean Square**	**F**	**Sig.**
1	Regression	45.916	11	4.174	713.373	.000[b]
	Residual	.386	66	.006		
	Total	46.302	77			
a. Dependent Variable: Secure RE Practices						
b. Predictors: (Constant), Requirements validation methods, The tradeoff between functional and non-functional requirements, Requirements estimation efforts, Security Risk Analysis, Requirements analysis and modelling, The valuation of assets and resources of the software being developed, System for requirements traceability to work products, Requirements gathering, Requirements inspection to identify potential threats, Identification of security goals, Security requirements identification.						

Table 5.30: ANOVA

F-statistic- is the Mean Square (Regression) divided by the Mean Square (Residual): 3.468/0.359 = 9.658.
Significance (p-value)-A significance value less than 0.05 means that predictors can be used to give a good indication of secure RE practices. The model predicts the outcome significantly as the p-value of 0.00 is less than the level of significance of 0.05 and that there is evidence that the independent variables predict the dependent variable (UCLA Institute for Digital Research and Education 2017). The next step is to create the coefficient table. This generates information on each predictor value. The table indicates the relationship between independent variables and the dependent variable.

Coefficients[a]						
Model		**Unstandardised Coefficients**		**Standardised Coefficients**	**t**	**Sig.**
		B	Std. Error	Beta		
1	(Constant)	0.008	.041		.197	.844
	Requirements gathering (X_1)	0.104	.013	.142	8.109	.000
	Identification of security goals (X_2)	0.073	.014	.110	5.299	.000
	Requirements analysis and modelling (X_3)	0.067	.013	.092	5.225	.000
	Requirements estimation efforts (X_4)	0.112	.013	.143	8.496	.000
	System for requirements traceability to work products (X_5)	0.090	.014	.109	6.261	.000
	The tradeoff between functional and non-functional requirements (X_6)	0.081	.013	.093	6.115	.000
	The valuation of assets and resources of the software being developed (X_7)	0.096	.012	.131	8.236	.000
	Requirements inspection to identify potential threats (X_8)	0.116	.014	.156	8.228	.000
	Security Risk Analysis (X_9)	0.077	.018	.112	4.174	.000
	Security requirements identification (X_{10})	0.078	.020	.109	3.912	.000
	Requirements validation methods (X_{11})	0.107	.012	.143	8.989	.000
a. Dependent Variable: secure RE Practices						

Table 5.31: Relationship between independent variables and the dependent variable

B – These are the values for the regression equation for predicting the dependent variable from the independent variables. Expressed in terms of the variables used in this project, the regression equation is (UCLA Institute for Digital Research and Education 2017):

$Y_{predicted} = b_0 + b_1X_1 + b_2X_2 + b_3X_3 + b_4X_4 + b_5X_5 + b_6X_6 + b_7X_7 + b_8X_8 + b_9X_9 + b_{10}X_{10} + b_{11}X_{11}$

From column B in Table 5.31 we derive the unstandardised coefficients (b0, b1 , b2 , b3 , b4 ,b5 ,b6 ,b7 ,b8 ,b9 ,b10 ,b11) and represent the regression equation, as follows:

Secure RE Practices *= 0.008 + (0.104 × Requirements gathering) + (0.073 × Identification of security goals) + (0.067 × Requirements analysis and modelling) +(0.112 × Requirements estimation efforts) + (0.090 System for requirements traceability to work products ×) + (0.081 × The trade-off between functional and non-functional requirements) + (0.096 × The valuation of assets and resources of the software being developed) +(0.116 × Requirements inspection to identify potential threats) + (0.077 ×Security Risk Analysis) + (0.078 × Security requirements identification) + (0.107 ×Requirements validation methods)*

If we consider the right hand side of the regression equation we can now describe the relationship between independent and dependent variables, for example if all other variables are held constant, 11.6 % of the influence in *Secure RE practice* is contributed by *Requirements inspection to identify potential threats* (UCLA Institute for Digital Research and Education 2017). Similarly this will apply to other predictors as well. What this tells us is that secure Agile RE Practices are most significantly impacted by *Requirements gathering, Requirements estimation efforts, Requirements inspection to identify potential threats, and Requirements validation methods.*

5.10.2 Regression model with impacting factors reduced

The regression model presented in Section 5.10.1 consistent with current literature showed 11 independent variables impacting secure requirements engineering. In order to make the model more applicable and focused a new regression model was developed. This entailed reducing the number of impacting factors and constructing a new model with fewer impacting factors. In constructing the new regression model only impacting factors over 10% were considered (*Requirements gathering, Requirements estimation efforts, Requirements inspection to identify potential threats, and Requirements validation methods)* from the model presented in Section 5.10.1. The following reasons are advanced for including *Security Risk Analysis* and *Security requirements identification* as independent variables with impacting values of lower than 10% in the new model, namely:

- The factor analysis results (Table 5.9) on secure RE practice showed two factor groupings for secure requirements practices. The first sub-theme identified was ‘security risk assessment practices’ and the second sub-theme identified was ‘conventional requirements engineering practices’. Both *Security Risk Analysis* and *Security requirements identification* are associated with ‘security risk assessment practices’.

- A combination of *Security Risk Analysis* and *Security requirements identification* impact values is higher than any single impacting factor.
- The dependent variable (secure RE practices) influenced the decision to consider all other independent variables related to security.

Table 5.32 depicts a summary model that indicates whether independent variables (predictors) predict the dependent variables. The independent variables are listed in point (a) of Table 5.32.

Model Summary				
Model	**R**	**R Square**	**Adjusted R Square**	**Std. Error of the Estimate**
1	.976[a]	0.953	0.949	0.17568
a. Predictors: (Constant), Requirements validation methods, Requirements estimation efforts, Security Risk Analysis, Requirements gathering, Requirements inspection to identify potential threats, Security requirements identification				

Table 5.32: Model Summary

The R^2 value (which is the coefficient of determination) indicates that the amount of variation that is explained in Y when changes are made to X. When all the points lie on the line, the R^2 value = 1 (100%). The smaller the R^2 value, the less accurate the explanations for the variations in Y becomes when changes are made in X (UCLA Institute for Digital Research and Education 2017). Table 5.33 indicates if independent variables can reliably predict dependent variables.

ANOVA[a]						
Model		**Sum of Squares**	**df**	**Mean Square**	**F**	**Sig.**
1	Regression	44.111	6	7.352	238.209	.000[b]
	Residual	2.191	71	0.031		
	Total	46.302	77			
a. Dependent Variable: Secure RE Practices						
b. Predictors: (Constant), Requirements validation methods, Requirements estimation efforts, Security Risk Analysis, Requirements gathering, Requirements inspection to identify potential threats, Security requirements identification						

Table 5.33: ANOVA

Table 5.34 indicates the relationship between independent variables and the dependent variable.

Coefficients[a]						
Model		**Unstandardized Coefficients**		**Standardized Coefficients**	**t**	**Sig.**
		B	**Std. Error**	**Beta**		
1	(Constant)	0.225	0.085		2.642	0.010
	Requirements gathering	0.166	0.026	0.226	6.347	0.000
	Requirements estimation efforts	0.183	0.025	0.234	7.328	0.000
	Requirements inspection to identify potential threats	0.158	0.030	0.212	5.232	0.000
	Security Risk Analysis	0.124	0.040	0.182	3.074	0.003
	Security requirements identification	0.146	0.041	0.205	3.553	0.001
	Requirements validation methods	0.162	0.026	0.216	6.338	0.000
a. Dependent Variable: Secure RE Practices						

Table 5.34: Relationship between independent variables and the dependent variable

The new regression model described in Table 5.34 is as follows:

Secure RE Practices = 0.225 + (0.166 × Requirements gathering) + (0.183 × Requirements estimation efforts) + (0.162 × Requirements validation methods) + (0.158 × Requirements inspection to identify potential threats) + (0.124 × Security Risk Analysis) + (0.146 × Security requirements identification).

The impacting factors are consistent with the two sub-themes identified in Table 5.9 showing the results of the factor analysis. The new regression model is intended to contribute to the existing body of knowledge.

5.11 Structural Equation Modelling (SEM)

The researcher used structural equation modelling to validate that concepts from RE literature and Security literature namely, *security approach, security knowledge and training, requirements elicitation activities* and *prioritisation of security requirements* (independent variables) are associated and can impact the level of *secure requirements engineering* (dependent variable*).*

The goodness of fit statistics is shown in the Table 5.35 with *security approach, security knowledge and training, requirements elicitation activities* and *prioritization of security requirements* as exogenous (independent) variables and *secure requirements engineering* as the endogenous variable.

Standardized	Coef.	OIM Std. Err.	z	P>|z|	[95% Conf. Interval]	
Requirements_activities	.6226133	.0649755	9.58	0.000	.4952637	.7499629
Security_Approach	.3538432	.0769586	4.60	0.000	.2030071	.5046792
Prioritisation_security_require	.38102	.0723871	5.26	0.000	.2391439	.522896
Security_Knowledge_and_Training	.1741485	.0758623	2.30	0.022	.0254611	.3228359
_cons	-3.708832	.6518664	-5.69	0.000	-4.986467	-2.431197
var(e.Secure_RE)	.4141728	.060365			.3112579	.5511157

Table 5.35: Structural Equation Model showing Goodness of fit Statistics

In Table 5.35 the p-values show all significant interaction between the independent and dependent variables.

The path diagram depicted in Figure 5.7 shows the relationship among dependent and independent variables and their coefficient values. The rectangle denotes variables that are directly measured. The lines with the arrow heads represent the direct effect of one variable on another. This means that the first variable affects the second variable, for example the absence of a security approach would affect the capability to conduct secure requirements engineering.

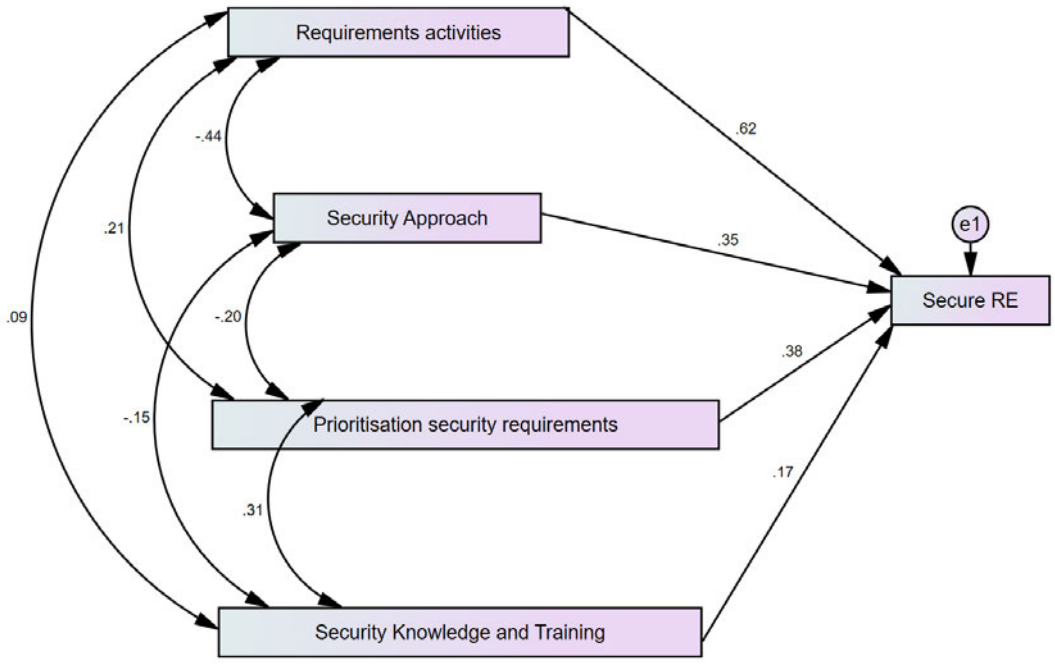


Figure 5.7: SEM Path Diagram for secure requirements engineering

Analysing the goodness of fit statistics in Table 5.35 and the Path Diagram in Figure 5.7 it can be concluded that *security approach, security knowledge and training, requirements elicitation activities and prioritisation of security requirements* predict *secure requirements engineering* practices. Figure 5.7 also shows the relationships between the exogenous variables. It is noted that the direct relationship between the endogenous and exogenous variables is more than 100% owing to associations between exogenous variables.

Hence the model for secure requirements engineering is given by the following structural equation:

> Secure requirement engineering = (α * *security approach)* + *(β * security knowledge and training)* + *(δ * normal requirements activities)* + *(γ * prioritisation of security requirements)* + *(error1).*

Where the maximum value of α is 0.35, β is 0.17, δ *is 0.62* and γ *is 0.38*. The max values are applicable when all other independent variables are omitted or excluded.
The independent variables and their tested association with secure requirements engineering were used to construct the product model for secure requirements engineering that is presented in Chapter 7 as an output of the study.

5.12 Dynamic Analysis Security Test Results

In order to validate the hypothesis that the security of the ASD product is dependent on the RE score obtained, albeit on face value as secure software development can be considered at any other phase in development, the researcher conducted some dynamic analysis security tests on selected projects (DAST).

The Acunetix web vulnerability scanner was used to conduct DAST on 4 of the 17 projects. The vulnerabilities obtained were classified in terms of Common Weakness Enumeration (CWE) list. The CWE list consists of comprehensive community developed software weakness types that are identified by a CWE number, for example CWE-200 is "information exposure". The vulnerabilities identified by the security scanner for each selected project is shown in the table

below under the column “CWE-No”. The description of the vulnerability type is written under column “CWE Vulnerability” as extracted from the CWE (2017) website.

Table 5.36 shows the column “RE ranking” which indicates the RE ranking of the project (from low to high) in terms of the 17 projects. A low ranking means “not well established practices for secure RE” and high means “mature practices for secure RE”; the column “RE score” shows the RE score given to the project by team members. The values for “RE ranking” and “RE score” were obtained from Table 5.24.

RE Ranking		RE Score	Project No.	CWE-No	CWE Vulnerability
LOW TO HIGH	**2**	**2.71**	**5**	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
				CWE-538	File and Directory Information Exposure
				CWE-200	Information Exposure
				CWE-16	Configuration
				CWE-352	Cross-Site Request Forgery (CSRF)
				CWE-307	Improper Restriction of Excessive Authentication Attempts
				CWE-910	Use of Expired File Descriptor
				CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
				CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
				CWE-310	Cryptographic Issues
				CWE-113	Improper Neutralization of CRLF Sequences in HTTP Headers ('HTTP Response Splitting')
	3	**2.77**	**8**	CWE-119	Improper Restriction of Operations within the Bounds of a Memory Buffer
				CWE-352	Cross-Site Request Forgery (CSRF)
				CWE-310	Cryptographic Issues
				CWE-16	Configuration
				CWE-521	Weak Password Requirements
				CWE-200	Information Exposure
				CWE-538	File and Directory Information Exposure
				CWE-307	Improper Restriction of Excessive Authentication Attempts
	5	**2.99**	**3**	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
				CWE-200	Information Exposure
				CWE-352	Cross-Site Request Forgery (CSRF)
				CWE-310	Cryptographic Issues
				CWE-693	Protection Mechanism Failure
				CWE-16	Configuration

	13	**3.56**	**10**	CWE-16	Configuration
				CWE-200	Information Exposure
				CWE-693	Protection Mechanism Failure
				CWE-400	Uncontrolled Resource Consumption ('Resource Exhaustion')

Table 5.36: Results of DAST

Source: Extracted from CWE (2017)

Table 5.36 shows that all 4 tested projects had software vulnerabilities. On closer examination of the table, it shows that projects 5,8 and 3 all had vulnerabilities related to the OWASP top ten (2013), namely, CWE-22 ("Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')") is *Insecure Direct Object References*; CWE-352 is *Cross-Site Request Forgery* (CSRF); CWE-89 ("Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')") is *SQL Injection* and CWE-310 ("Cryptographic Issues") is sensitive data exposure. Table 5.32 shows that project 5 had 4 vulnerabilities (CWE-22,CWE-352,CWE-89,CWE-310) associated with OWASP top ten (2013); project 8 had 2 OWASP top ten vulnerabilities (CWE-352, CWE-310); project 3 had 3 OWASP top ten vulnerabilities (CWE-89, CWE-352, CWE-310) and project 13 had none.

Analysis of the results in the table does show some support for the premise that Agile RE practices (RE score) provides an indication of the security of the ASD product. The assumption here naturally is that secure software development practices are not conducted at any other phase during development. Finally, the vulnerabilities detected by DAST could have been prevented through more secure RE practices.

5.12 Chapter summary

In this chapter the results obtained from phase 1 of the data collection namely, the field survey were presented. The results were presented in the form of frequency distribution tables and graphs. Hypothesis tests and correlation analysis were also completed to discover trends and significant relations in secure Agile RE. The typical respondent to the study was a permanently employed male, with some level of post-secondary education. He did not receive application security in the last 12 months. In a typical project at his company the requirements were specified in an informal language. Brainstorming and user stories were the RE elicitation techniques utilised. The software

development team was responsible for identifying security requirements in the project. Requirements for the project were validated by the developers and not specialists. Changing requirements was confirmed as the biggest challenge to secure requirements engineering in the project.

Furthermore, project 12 had the lowest mean rating (2.03) for secure requirements engineering whilst project 17 had the highest mean rating (4.10). Project 12 came from a small company with only one development team functioning. Project 12 came from a company that is a large international software development company operating in a multi-project and multi-team environment. The survey reflects favourable results for RE processes in Agile RE practice however more analysis was required on the extent of secure RE practices in ASD. This was conducted in phase 2 of the study.

The regression model on secure RE practices was presented with eleven impacting factors. As a refinement of this model a new regression model was presented as the researcher unique contribution. "Requirements inspection to identify potential threats" had the most impact on secure requirements engineering. Finally, Dynamic Analysis Security Testing (DAST) was conducted to determine the security of the ASD product. A major finding here was that three of the four projects scanned, had OWASP top ten vulnerabilities. The analysis of the qualitative results and findings are presented in the next chapter.

CHAPTER SIX: PRESENTATION OF QUALITATIVE RESULTS AND FINDINGS

6.1 Introduction

The ASD methodology is a lightweight process model that embraces flexibility in processes and allows for changes in requirements. Traditional software development processes are minimised or replaced with ad hoc processes in this approach. Therefore, it was critical to assess the extent to which mainstream RE processes from the software engineering discipline was implemented in Agile RE practice in industry without compromising the quality of the output. Specifically, the researcher explored how much prominence was given to non-functional requirements, specifically security.

The literature review (Chapter Two) reflected that well recognised software engineering processes, for example, requirements prioritisation and security requirements engineering are two important RE processes that impacted the security of an application. As such, the qualitative fieldwork explored Agile RE practices with an in-depth focus on these processes, namely, requirements prioritisation and the extent of secure requirements engineering in the software development industry. The approach for the discussion of the data analysis is as follows:

- Themes and sub-themes were identified
- Emergent data patterns from the content analysis for current state of practice within each theme are presented
- Activity Theory (AT) is used as an analytical lens through which Agile RE practices are presented.
- Narratives from the field work, where applicable, are provided as supporting evidence

6.2 Recap of AT Concepts

Activity Theory is discussed extensively in Chapter Three and the main aspects of AT are recapped in this section. AT is used as a lens through which Agile RE practices in real life are analysed. Agile RE practices are exposed and unravelled in the fieldwork through the themes using data collection methods such as interviews and document reviews. The data was analysed using each theme as an activity system on its own. The interplay between the components of the activity

system, namely, *subject-community-object* within the software development environment is used to inform our understanding of Agile RE practices. *Subject* refers to the individual or group participating in the activity system. *Community* refers to participants of the activity system who share the same object. It is important to have an understanding of the techno-social interactions surrounding their activities. *Object* is the expected outcome of the activity. It motivates the activity (Murphy & Rodriguez-Manzanares 2008).

Tools, division of labour and *rules* serve as mediators in the activity system. *Tools* are the physical/psychological tools required to perform the activity. The *division of labour* involves division of tasks or assigning roles amongst members of the community. *Rules* are norms that regulate actions. They can be formal or informal rules that guide the activities. The relationship between *subject-object* is mediated by tools, the relationship between *subject-community* is mediated by rules and the relationship between *community-object* is mediated by the division of labour (Uden 2006).

The three relationships denoted as *subject-tool-object*; *subject-rule-community* and community-*division of labour-object* must be understood broadly in an activity system as they represent the interaction between elements of the activity system (Kuuti 1995). Agile RE practices in this study was analysed according to the themes above using these three relationships. Each relationship describes the activity towards an object in order to achieve the outcome, namely, of generating highly refined requirements to support secure systems development.

Figure 6.1 represents a simulated activity system constructed for the software development industry.

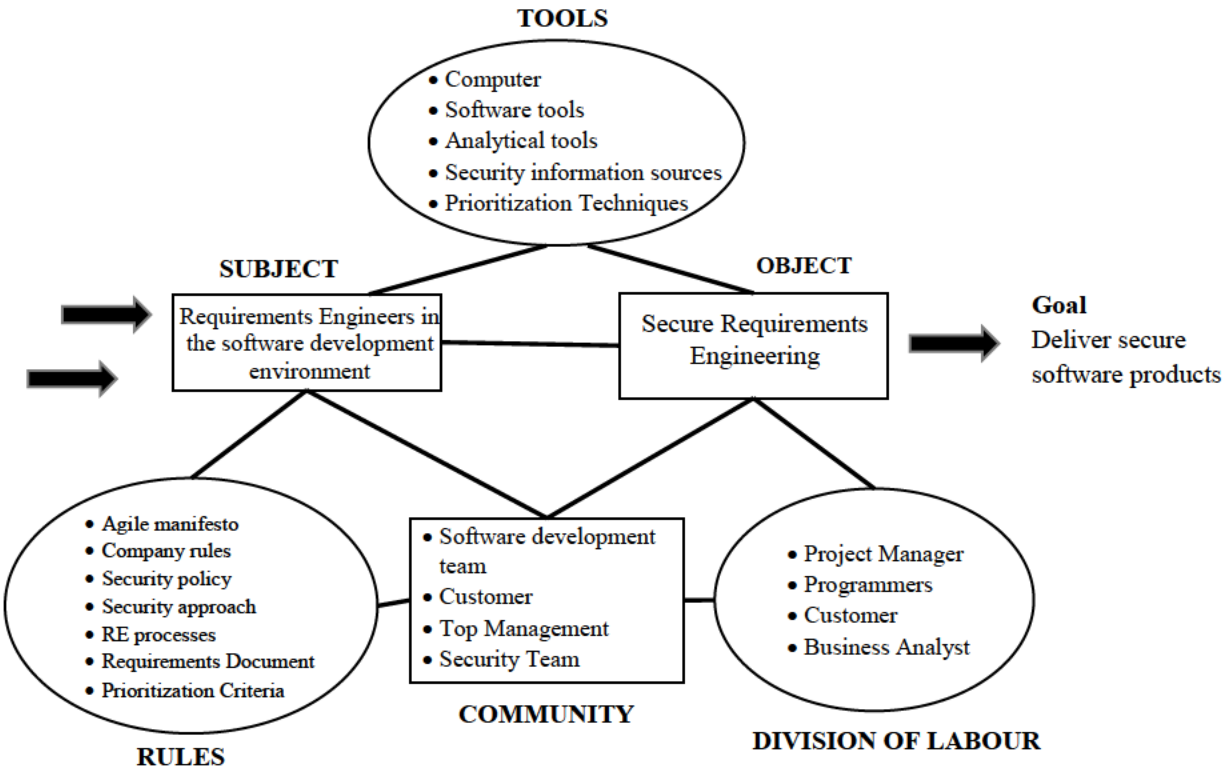


Figure 6.1: Activity System for Agile RE
Source: Researcher's own construction

6.3 The fieldwork

The fieldwork represented phase two of the data collection. The data was collected from seventeen software development companies implementing Agile Software Development. The companies were concentrated in KwaZulu-Natal and Gauteng. Local and international software development companies were chosen to be part of the study. The core business of these companies were medical solutions, educational solutions, casino and gaming applications, telecommunication solutions, call centre solutions, media software solutions and general purpose solutions. The aims and objectives of the research were already presented to CEOs and project managers prior to the fieldwork. One the first day of the fieldwork the researcher was introduced to a randomly chosen team of developers by the CEO or project manager. While the project managers at most of the companies respected the need to conduct the research they requested that the researcher's intervention should cause minimum disruption to the work day of the developers. Interviews were the primary data collection tool. The software development teams were the main informants, providing data on their most recently completed project. Interviews were conducted on an

appointment basis. Interviews were scheduled during lunch breaks, tea breaks, very early in the morning or very late in the afternoon. The interviews were 45 minutes to 120 minutes in duration. Interviews were recorded on the researcher's iPad.

Secondary data were collected at study sites by the researcher who was granted access to release documentation. The researcher was also allowed access into the developers' open plan work space to view artefacts such as current project documentation, white boards for project planning such as current and proposed UML models, business requirements documentation, product backlogs and minutes of meetings. The researcher was also able to observe informal discussions between developers and view software tools such as jira and pivotal tracker on their work stations.

Data analysis is presented thematically. The themes within secure Agile RE emerged preceding the extensive literature review and the findings from both quantitative and qualitative data analysis. The themes are as follows:

- Theme 1: Requirements Elicitation
- Theme 2: Establish Viewpoints
- Theme 3: Security Requirements Elicitation
- Theme 4: Security Approach
- Theme 5: Security Training
- Theme 6: Customer Involvement
- Theme 7: Analysis and Prioritisation of Requirements
- Theme 8: ASD RE Satisfaction
- Theme 9: Constraints to Secure Requirements Engineering
- Theme 10: Best Practices for Secure Requirements Engineering

The ten themes do not necessarily coincide with distinct knowledge areas in RE but are critical aspects selected by the researcher based on the findings of the study. A software tool called NVivo pro was used to assist with coding and analysis.

6.4 Theme 1: Requirements Elicitation

Requirements elicitation in RE focuses on ensuring high quality requirements are extracted from the user. The document review in this study showed that requirements generated for projects are highly refined, lacking complexity and ambiguity. The data analysis indicated the following:

6.4.1 Business analyst elicited requirements

It is important to note that across companies, in Agile RE practice requirements elicitation did not involve the entire software development team. The business analyst emerged as the most important stakeholder involved in meetings with the customers, to elicit requirements.

Community-division of labour-object: In software development teams (community) requirements elicitation (object) was conducted by the business analyst in consultation with the project manager (division of labour). Developer 1 explained that the user requirements were derived by the business analysts. They visited the client and got the specifications for the system. They brought the requirements back and did further brainstorming with the project manager. They proposed the cost and estimates.

6.4.2 Business requirements document

Business analysts generated a business requirement document with inputs from project managers. When this document of requirements is completed, it gets signed off by the client. Thereafter, the project managers take responsibility for constructing the system in several releases.

Subject-rule-community: Business analyst (subject) generated a requirements document with requirement specifications (rule) for the software development team (community) to implement.

6.4.3 Business analyst on the customer side

Some important clients have a business analyst in their employment. These clients make the process of requirements elicitation very easy for the software developer as the client's business analyst captures the critical customer requirements on the client side. This reduces the overall RE process time for the business analyst since the client's business analyst submits a completed business requirements document to the software development business analyst.

Subject-tool-object: The business analyst (subject) received a completed document of requirement specifications (tool) from the client's business analyst for the new system (object).

In summary, common practices of Agile RE in industry has shown that requirements elicitation is the responsibility of the business analyst. Software development companies refer to a business analyst or team of business analysts by different names, for example, product specialist, sales team or exploration team. The business analyst or team of business analysts collect requirements for all projects and is part of an independent team. This member or team is not directly part of the ASD team and provides support to all project managers and individual teams during the development.

6.5 Theme 2: Establish viewpoints

Several users of the system must be identified and each user's viewpoint must be considered for the new system when establishing viewpoints. The data analysis on the establish viewpoints process in Agile RE practices revealed the following:

6.5.1 Business Analyst establishes viewpoints

The business analyst conducted meetings with clients in order to establish the user's viewpoints. The team was not involved. The business analyst also used informal communication such as e-mails and telephonic conversations to establish viewpoints. These viewpoints were captured in the business requirements documentation.

Subject-tool-object: The business analysts (subject) captured user's viewpoints (object) from end user meeting memos (tool) after conducting several meetings.

Subject-rule-community: The business analyst (subject) used business rules (rules) to ensure that all viewpoints of the customer (community) are established.

6.5.2 Viewpoints captured in business requirements document

Users' viewpoints were captured in the business requirements document (BRD) by the business analysts. The document was presented at a project 'kick-off' (JAD) meeting involving all stakeholders. Developers could ask questions at this meeting regarding users' viewpoints.

Subject-tool-object: The software developer (subject) used the business requirements document (tool) to consider the users' viewpoints (object).

6.5.3 Clarification of users' viewpoints

During development, when developers were unsure of a requirement, the business analyst or project manager got involved. If they could not provide explanations, then an e-mail was sent to the customer or the customer is called in to a meeting. Prototypes for functional requirements were recommended because the client very often did not know what they wanted.

Community-division of labour-object: team members (community) requiring clarification of user viewpoints (object) engaged with the project manager or business analyst (division of labour) as they have expert knowledge of the system.

In summary the common practice that emerged from the data is that the business analyst plays the most important role in establishing viewpoints. The business analyst liaises with the client to ensure that all viewpoints are established. When developers need clarification on requirements, the business analyst, who is a subject specialist, provides the first level of support.

6.6 Theme 3: Security requirements identification

Security requirements focus on protecting assets of the system and prevent unauthorised access and violation of system assets. These requirements are identified at the requirements engineering stage as part of system planning. The data analysis for security requirements identification in Agile RE practices indicated the following:

6.6.1 Insufficient security requirements identified

The business analyst as an extension of the software development team was responsible for identification of security requirements. Security experts were not involved. Insufficient security requirements were elicited for projects. Inspection of the business requirements documentation from several projects revealed that very little or in some cases no security requirements were identified. Only at one large international software development company, informants indicated that they had an independent security team that was responsible for identification of all security requirements. Review of their business requirements document confirmed that security requirements were in place for the project.

At the beginning of the project it is the business analyst's responsibility to elicit and document security requirements from the client. It was evident that these requirements are kept to a minimum.

Subject-rule-community: The business analyst (subject) used the security policy (rule) of the customer (community) to elicit security requirements.

6.6.2 Security requirements identified at coding

The researcher wanted to gain better insight to determine from informants who, in particular, identifies security requirements for specific projects. Responses from the small to medium software development companies indicated that the development team was responsible for identifying security requirements. The customer was not involved.

Subject-tool-object: The software developer (subject) used the requirements document (tool) to identify security requirements (object) at coding.

6.6.3 Security requirements identified are generic (not customised)

The researcher sought clarification on the type of customised security requirements identified at requirements engineering phase. The common Agile RE practice indicated that security requirements were restricted to generic security requirements such as authentication and access control.

Community-division of labour-object: The project manager (community) assigned a limited number of security requirements (object) to specific team members for development (division of labour).

6.6.4 Security requirements identified after system violation

Informants indicated that customised security requirements were only specified when there was a violation of the system and the customer or team leader requested it at implementation. A reactive rather than a proactive approach was then adopted. In this case, a security requirement came in as a system enhancement.

Subject-rule-community: Developers (subject) are requested to identify customised security requirements by the project manager (community) only when there is a violation of system assets (rule).

In summary, common practices in Agile RE are as follows: large companies have dedicated security teams. They have established processes to ensure that security requirements are identified. Small to medium companies rely on the discretion of the software development team such as the business analyst or team leader for identification of security requirements during requirements engineering. Generally, in these companies security requirements are limited in number and are generic. Security requirements are identified as system enhancements when there is a violation or when the customer specifically requests it.

6.7 Theme 4: Security approach

A security approach in requirements engineering is to deploy a systematic method to unravel and identify security requirements. In software engineering literature, this RE process is known as security requirements engineering. The data analysis indicated the following common practices in Agile RE:

6.7.1 No formal security methodologies

The data analysis indicated that there are no known formal security methodologies for eliciting security requirements used. Security risk assessment was not conducted at most companies.

Subject-tool-object: Developers (subject) did not normally conduct formal security risk assessments (tool), except at the request of customers for secure requirements engineering (object).

6.7.2 Ad hoc approach on security at coding

Subject-rule-community: Under special circumstances the team lead (community) might request that security risks are mitigated by the developer (subject) during the coding (rule). Developer 2 explained that at the team meeting everything is broken down in terms of all the technical aspects. Thereafter each person is assigned tasks. A developer must analyse all the concerns for his task. The team member does his own risk assessment for example he might identify a feature must be accessible only to some users. The team member decides what vulnerability he wants to mitigate against. Risk assessment is not done collectively. It is done in isolation with the team lead and the developer who is assigned that feature to code. For example if cross site scripting is a concern of the team member, he takes that concern to the team lead. The team lead then makes the technical decisions.

6.7.3 Business Analyst to ensure security requirements

The business analyst ensured that the client's security policy was implemented in the application.

Community-division of labour-object: Customers (community) security policies might be considered in the requirements document by the business analyst (division of labour) within the security approach (object).

In summary, the data patterns from the interviews, business requirements documents and informal discussions with developers showed that small to medium companies have not adopted any known approaches to security. It was the duty of the business analyst (a subject matter and domain expert) through consultation with the client to specify security requirements at requirements engineering phase. Owing to the business analyst's limited knowledge, very few to no security requirements were elicited. Security approaches in general may be factored into the design and development under the following circumstances, namely:

(i) Developer requests for security approaches during coding;

(ii) Customer requests security during requirements engineering,

(iii) Business analyst includes security through consultation with the client and client security policies during requirements engineering.

However in regard to point (iii) above, the researcher noted after viewing the requirements document, that only a few critical security requirements such as authentication and access control were factored in.

6.8 Theme 5: Security training

Security training in application development assists developers to combat known vulnerabilities. The data analysis indicated the following:

6.8.1 Application security training not prioritised

Data analysis showed that very little or no security training is provided to developers within the organisation. See Annexure J. Owing to poor training in security, developers often confused architectural security requirements with application security requirements at interviews. Developers indicated that they use social media sites like 'You Tube' when they want to learn something in application security.

Subject-rule-community: Project managers (community) have not prioritised application security training needs (rule) for developers (subject).

Community-division of labour-object: Large companies (community) with dedicated security teams (division of labour) provided application security training (object) for specialist staff.

6.8.2 Poor awareness of security knowledge sources

Very few developers, especially business analysts had any awareness of security knowledge sources. Few informants indicated that they received regular updates from security knowledge like Open Web Application Security Project *(*OWASP) and Common Weakness Enumeration (CWE), but this was their personal initiative.

Subject-tool-object: The software development team (subject) did not consult security knowledge sources (tool) for awareness in security trends (object).

In summary, two distinguished practices for application security training emerged. In large companies security training is provided for the affected employees whilst in smaller companies security training is not prioritised. See Annexure J confirming this. Further, it was apparent that team members are unaware that the latest trends in terms of application security can be sourced from security knowledge sources such as OWASP and CWE.

6.9 Theme 6: Customer involvement

Customer involvement in ASD projects is an important ASD principle and in order to reap the benefits of ASD, the customer must be involved in the software development.

6.9.1 Poor customer involvement

The content analysis showed that customer involvement in ASD was poor across all organisations. The customer came to meetings at the beginning of the ASD project. They were not working with the development team on-site. They came to meetings when they were requested to attend. If they could not attend, the meeting got re-scheduled. They were only involved in very crucial meetings.

Community-division of labour-object: Customer involvement (community) in identifying security requirements (division of labour) in ASD must be improved.

Subject-tool-object: The business analyst (subject) very rarely received requests in the business requirements document (tool) from the customer for security requirements (object).

6.9.2 Customers technical team elicits requirements

Some customers had their own in-house team of business analysts. They elicited all the requirements including security.

Subject-rule-community: In cases where customers had the support of their own technical teams (community), these teams used their security policy (rule) to include security, which the development team (subject) had to implement.

In summary, the common practice is that there is insufficient involvement from the customer. This is in contradiction to the ASD methodology. The customer is only interested in the functionality of the system in terms of the business rules and very rarely makes requests for security to be included.

6.10 Theme 7: Analysis and prioritisation of requirements

Requirements prioritisation is a very important process in Agile RE. Requirements with high priority are implemented first to add business value to the customer. Incorrect prioritisation can lead to a loss of value eventually resulting in project failure. When security requirements are lowly ranked, they don't get implemented. Prioritisation is an important component of requirements analysis. The data analysis showed the following common patterns.

6.10.1 Insufficient team involvement in requirements prioritisation

ASD literature supports that multiple stakeholders, for example the business analysts, the project manager and the customer using multiple decision making criteria should prioritise requirements. The study indicated that, in Agile RE practice, only the project manager decided the priority of the requirement using multiple criteria. They consulted with technical documents and business analysts when necessary. They prioritised based on their vast experience in the field or reached consensus with other process owners such as the business analyst.

Subject-tool-object: The project manager (subject) used multiple criteria (tool) to prioritise user requirements.

6.10.2 No proper prioritisation techniques used

Companies do not implement any requirement prioritisation techniques. No software application is used to assist with the prioritisation of requirements.

Subject-rule-community: Project managers (subject) held meetings with business analysts (community) to rank requirements based on multiple criteria (rule).

6.10.3 Lack of direct involvement by customer

The project manager completed the ranking process on behalf of the customer. He ranked the customer's preference first followed by ranking according to various other criteria. This meant that the customer was not directly involved in the process.

Community-division of labour-object: The project manager (community) ranks requirements (division of labour) with little customer input to prioritise the product backlog (object).

The general ranking process at an established international software development company was summarized by Developer 3. All functional requirements are ranked. If the customer has 10 requirements they inform the project manager what features they want first. Customers normally advise on the basis of core requirements. According to a senior developer at the company the ranking occurs in the following approach. Developers analyse the requirements, for example if feature XX is released and the development time is 4 months and if feature YY is released and the development time is two months. When feature XX is live the turnover (revenue) for the customer in one month is more than feature YY in one month. XX has higher revenue for the customer and is selected for implementation. Developers consider the development time and revenue to the customer. This is known as opportunity cost. As a rule developers prioritise where they can make more money for the customer.

Sometimes the feature is simple and the development time is long. In this case the project manager informs the customer of the time taken to release the feature. If the release of the feature has a huge impact on the turnover this will be communicated by the customer. Despite a lengthy development time it is chosen for implementation. Also there can be interdependence of requirements, for example feature 1 and 2 must be done before feature 5. In this case, there is no negotiation with the customer. Feature 1 and 2 are both selected together for implementation. Other important criteria that developers consider are business value and weight of the customer in a multi project environment.

Non-functional requirements were not prioritised at many companies. There were standard ways of doing things and non-functional requirements were brought in from past projects. Developers

did not rank non-functional requirements. Once the frameworks were in place for non-functional requirements, it was easy for the development team to implement at coding stage. For example, if they wanted to create a login screen, the encryption mechanism was taken from the security framework. It was considered beneficial for a security framework and a scalability framework to be added to the new project. The need to code complex repetitive functions was removed. Existing projects were often used as a source of information. This saved time, resources and money.

Common practice in Agile RE is summarised as follows: All stakeholders are not involved in the prioritisation process. The customer does not play a physical role in the prioritisation of requirements. There is no specific prioritisation technique utilised in practice and requirements are more often than not prioritised by the project manager in conjunction with the business analyst based on their combined knowledge and experience. Important criteria to consider are opportunity cost for client, business value, dependence and weight of the customer (multi-project multi-team environment).

6.11 Theme 8: Agile RE Satisfaction

The researcher explored team member's satisfaction with Agile RE practices. Agile RE practices in this research is to assess the extent that mainstream RE processes are applied to Agile RE. The data analysis indicated that team members did not explicitly express their dissatisfaction with Agile RE practices but rather raised the following concerns for improvement:

6.11.1 More customer involvement

Common practice in Agile RE suggested that developers are not inclusive of customers in RE processes, as in the case of elicitation of non-functional requirements such as security.

Subject-tool-object: Developers (subject) did not use any form of communication (tool) with the customer to get input from the customer on security (object).

6.11.2 Competence of business analyst

An observation made by developers is that the business analyst were not sufficiently knowledgeable about the actual customer requirements. There were more often than not changes to the requirements that come from the business analyst. These changes were not about the client

changing requirements but it comes from a lack of understanding by the business analyst of what the client really wanted. This hinders the development process.

Community-division of labour-object: The business analyst did not get input from the team members (community) when generating the technical specifications document (division of labour) hence compromising the quality of the requirements (object).

Subject-rule-community: The business analyst (subject) must liaise with the software development team (community) for technical specifications (rule) before finalizing the technical specifications document. Developer 4 in agreement explained:

"The business analyst always misses out things. There are always loopholes and we seldom get a perfect requirements document".

6.11.3 High level view of new system

The document review showed that requirements are mainly in the form of user stories. These requirements may be a task or further broken down into many tasks that are managed using project management software such as pivotal tracker or jira. Many developers were of the opinion that the user stories do not give the big picture of the system under development as compared to a UML diagram. When a developer gets to implement a requirement they need to have a high level view of the system.

Subject-tool-object: A developer (subject) needs to view UML diagrams (tool) to have a high level view of user requirements (object).

6.11.4 Team work

Many of the teams felt that the personality of the requirements engineers can impact the success of the project. Personnel involved in collecting requirements must have a good temperament to deal with queries and share information with the development team. Other than the customer they have the best understanding of the system. When the customer is not on site there must be no hesitation to contact the customer.

Subject-rule-community: Developers (subject) believed that they are not taken seriously due to certain biases (rule) of senior stakeholders (community). Developer 5 complained that the pecking order of the team is very important. His view was that in a team if you are not senior in ranking nobody listens to you even if you make good points. Senior developers don't take your views into account. It becomes very difficult and frustrating to reason with higher authority.

Developer 6 (scrum master) was in agreement:
"In scrum meetings difference in opinion can lead to serious personality clashes which can be detrimental to the project and your job security."

6.11.5 Staff retention

There were developers that felt that a requirements engineer is an important stakeholder in the software development. He is the one who generally has product knowledge not the developer. The concern was that software development is a competitive industry and should the requirements engineer leave, it creates lots of issues for the programmers.

Subject-rule-community: The business analyst (subject) has expert knowledge in the field (rule) and supports all stakeholders (community). Developer 7 explained that the biggest problem encountered is when staff who gathers the requirements leave. He was working on a financial application and was struggling with understanding the requirements from an incomplete requirements document. He could not consult with anyone. Lots of information is lost when the staff member is lost to the company.

Overall, informants did express satisfaction of RE processes used in their ASD projects however requested the following adjustments: finding a way to be more open with clients on non-functional requirements such as security; Business Analysts must be more thorough about the requirements documents and seek clarification from the customer and technical staff when needed. This will prevent confusion; Developers need a high-level view of the system; Social interactions and personality traits play an important role in requirements engineering and more often than not it must be improved within the team and finally, staff retention of requirements engineers poses a threat to the success of the development process.

6.12 Theme 9: Challenges to secure requirements engineering

The researcher explored the constraints to secure requirements engineering in ASD practice under this theme. The constraints were seen as hindrances to ensuring that security is included in the requirements engineering phase of projects. The data analysis indicated the following:

6.12.1 Insufficient training

One of the biggest challenges to secure requirements engineering is that developers received very

little or no training in application security. In order to ensure that security is included in the system, requirements engineers must attend security training.

Subject-tool-object: *Developers (subject) received no training in application security (tool) and therefore did not know every situation where security was necessary (object).*

6.12.2 Access to security knowledge sources

Another problem cited in the data analysis was that developers were not accessing security knowledge sources. Senior management did not support developers through budget or time to access the relevant knowledge sources applicable to projects.

Community-division of labour-object: Developers are not supported by top management (community) to consult security knowledge sources (division of labour) for security requirements in their projects (object).

6.12.3 High workload

Software developers had to multi-task in their day. In regular companies the high workload ensured that the focus was on functional requirements. The situation degenerated in a company operating in a multi-project and multi-team environment. The common complaint was that workload issues prevented developers from considering unrequested security issues of the system.

Community-division of labour-object: The project manager (community) distributed high workloads (division of labour) to developers to ensure that functional features are delivered (object) within deadlines.

6.12.4 Other critical learning

Another constraint was that a great deal of the developer's time, other than attending meetings, was spent on learning new technology. They were compelled to learn new technology as seniors requested that they implement new technology in projects. Much of this learning took place through self-study. This shifted the focus away from security in Agile RE. Even if security requirements are specified, it was lowly ranked and did not get implemented.

Subject-tool-object: A developer (subject) must learn new technology (tool) in order to develop high quality products.

6.12.5 Fixation on functional requirements

To add to this there was a constant demand for new features from the customer. These features

added business value to the customer. It brought in revenue and developers were compelled to meet the customer's demands. Once the system was developed the customer did not view security as adding value. They only dealt with security issues after core features of the system were released and the system was live.

Subject-rule-community: Developers (subject) must add business value (rule) to the client (community) by releasing functional features of the system, not security features.

6.12.6 Constantly changing requirements

Another impeding factor is that in the real world there are clients who change requirements frequently. According to a project manager, in order to satisfy the customer, the development team accedes to implementing changes because contracts are worth millions. Constantly changing requirements reduces the focus on secure requirements engineering.

Community-division of labour-object: Developers contended with changing requirements (division of labour) from the customer (community) which made it impossible to consider security requirements (object).

6.12.7 High paced ASD environment

Developers are complaining about not having enough time for development. The sprints are very short and demand high productivity. Every day begins with a daily stand up meeting. These meetings place a lot of pressure on the developer and many feel that they are being micro-managed.

Subject-rule-community: Developers (subject) attended daily stand-up meetings (rule) with the software development team (community) which created a high paced ASD environment. Developer 8 explained that ASD requires daily scrum meetings for 15min. Developers must answer the questions such as what they did yesterday and what are they going to do today? The downside of this is that people sometimes feel that they are being micro-managed. He also expressed that it's too much for the manger as well. Every morning scrums are daunting because it enforces productivity. His view was that the purpose gets defeated when people come to the meeting and lie about what did. The customer pays for features not security. According to a project manager they are not prepared to pay for non-functional requirements. Hence cost is another factor that hinders secure requirements engineering. Top management do not have budget for security features.

Subject-rule-community: Customers (community) due to high security costs (rule) do not want

developers (subject) to include security in the system.

The challenges from the data analysis can be summarised as follows: Requirements engineers are not trained in security; requirements engineers are uninformed about security knowledge sources and are not updated on vulnerabilities from these sources; there is a high workload for development teams; learning new technology takes precedence over security; constant fixation on new features not security; changing requirements; not enough time given for development and finally, the high costs to implement security.

6.13 Theme 10: Best practices for Secure Requirements Engineering in ASD

Current Agile RE practices suggest that security does not feature prominently in requirements engineering processes. The reasons for this have been considered in the data analysis results under Theme 9. Data was gathered on this theme based on evidence from practitioner's best practice and Informants opinions on what they considered important for secure Agile RE. The findings from the analysis are presented below:

6.13.1 Customer involvement

Firstly developers want more client involvement in generating security requirements. Developers must be more open and educate clients on security issues. Clients must be able to specify in the requirements engineering phase what security requirements should be in the system.

Community-division of labour-object: Client involvement (community) in eliciting security requirements (division of labour) can ensure that security is included in requirements documentation (object).

6.13.2 Security sprint

A dissenting view from a senior developer that can contribute to best practice was that the only way to give security prominence is to have a dedicated sprint for security before sprint planning at every iteration. This is a more upfront and proactive approach to security.

Subject-rule-community: Requirements engineers (subject) after gathering normal requirements must call for a security sprint (rule) involving all stakeholders (community).

6.13.3 Avoid using an SQL database

Some developers felt that lots of security problems are caused by using an SQL database. If

developers avoid using a SQL database much of the problems around hacking can be alleviated. The type of database should be explicitly stated in requirements documentation.

Subject-tool-object: Some *d*evelopers (subject) used a hierarchical database (tool), like an object oriented database with chronological files, to eradicate SQL security concerns (object).

6.13.4 Outsource application security

A dissenting view from a developer was that companies can outsource security requirements to dedicated security companies. This will reduce the time constraints that developers experience in implementing security. Added costs will be incurred but this must be offset by the customer. Therefore customer involvement is very important.

Subject-rule-community*:* project managers (subject) must outsource (rule) security to specialist security companies (community) that provide dedicated application security services.

6.13.5 Security frameworks

Problems around security can, to an extent, be fixed through the use of security patterns and security frameworks in ASD. Specifics of patterns and frameworks must be included in the requirements document.

Subject-tool-object: Some developers (subject) used security patterns and security frameworks (tool) to ensure a basic level of security in the system (object). Developer 2 explained that Microsoft publishes prescribed security patterns that developers can use. Google and Facebook also have templates based on the application that you are constructing. His view was that applications are going cloud (Azure) and indicated that non-functional requirements will fall away in future. He pointed out that as soon as you create a project in Azure, security templates are already in place and confirmed that security is sealed in the Azure framework.

6.13.6 Dedicated work time for security

Some developers agreed on a very novel and practical solution around security i.e. create time for requirements engineers in the work day to get up to speed with latest trends in security. This will force them into security.

Subject-rule-community*:* Some project managers (community) ensured that there was time dedicated in the work day (rule) for requirements engineers (subject) to consider security in projects.

Best practice for secure requirements engineering from the data analysis can be summarised as follows: Client involvement in security must be mandatory; dedicate a sprint to address security requirements; avoid using an SQL database; outsource security requirements to dedicated security companies; use security patterns and security frameworks and finally, during the work day dedicate time for security.

6.14 Thematic Summary

Table 6.1 summarises the major findings of the qualitative study according to the themes identified.

Theme	Agile RE practices
Requirements Elicitation	• Business Analyst elicited all requirements. Team not involved. • Business Requirements Document generated by business analyst. • Some customers have a Business Analyst on their side. • Business Analyst works independently of the ASD team.
Establish Viewpoints	• Business Analyst establishes viewpoints. • Viewpoints captured in business requirements document. • Business Analyst clarifies users' viewpoints when necessary.
Security Requirements Identification	• Identified by software development team, namely business analyst • Insufficient security requirements identified. • Security requirements identified at coding when there is a glaring issue. • Security requirements identified are generic (not customised). • Customer requirements are only specified when the company experienced a security violation in another project and want to mitigate upfront in the new project. • Security requirements are identified after a system violation.
Security Approach	• No formal security methodologies used. • No security experts used. • Ad hoc approaches on security: • Developers request security approach during coding. • Business Analyst to ensure security requirements are elicited through consultation with the client and client security. These are restricted to authentication and access control. • Customer can request security during requirements engineering. Rarely happens.
Security Training	• Application security training not prioritised • Poor awareness of security knowledge sources
Customer Involvement	• Poor customer involvement. • Some customers have their own team of Business Analysts that elicit requirements on their behalf.
Analysis and Prioritisation of requirements	• No team involvement in requirements prioritisation. • No proper prioritisation techniques used. • Lack of direct involvement by customer.

	• There is no specific prioritisation technique. • Prioritisation generally takes place through consensus. • Important criteria: opportunity cost, business value and dependence.
Agile RE satisfaction	• More customer involvement especially for non-functional requirements. • Competence of the business analyst in question. • No high level view of new system. • Team dynamics an issue. • Staff retention for key staff.
Challenges to secure requirements engineering	• Insufficient Training. • Access to security knowledge sources. • High workload. • Other critical technology innovation learning. • Fixation on functional requirements. • Constantly changing requirements. • High paced ASD environment. • Small companies do not have budget for security
Best practices for secure requirements engineering in ASD	• Customer involvement. • Security sprint. • Avoid using an SQL database. • Outsource application security during RE. • Security Frameworks.

Table 6.1: Summary of findings in Agile RE

6.15 Chapter summary

In this chapter the results obtained from phase 2 of the data collection namely, the qualitative study were presented. The data was thematically analysed and presented from the interview question responses and document reviews, using AT. The exploratory qualitative study sought further explanations needed from the field survey as summarized in Table 6.1. In addition, the qualitative study triangulated and strengthened the survey results. The following major findings in the survey study were confirmed in the qualitative study: the business analyst who is an extension of the software development team was responsible for identifying security requirements and dedicated security experts were not involved; requirements for projects were validated by the team; changing requirements was a major challenge to secure requirements engineering; large companies have dedicated security teams while smaller companies with a single team have no security experts and finally a large percentage of developers did not receive security training. In the Chapter 7, the interpretation and discussion of results will be presented.

CHAPTER SEVEN: DISCUSSION & INTERPRETATION OF FINDINGS

7.1 Introduction

In this chapter the results from both the quantitative and qualitative study are collectively interpreted. Recommendations for improvement are provided based on the researcher experiencing best practice in the field work as well as evidence from Agile Software Development (ASD) literature. The chapter begins with a review of the emergent Soft Activity Model (SAM). The model was constructed to serve as a lens and provide a framework to interpret and discuss study results. The results were interpreted under 8 themes. In addition to various recommendations provided under each theme from the body of knowledge, a three step security approach, a process model and product model is proposed as unique contributions, to guide secure requirements engineering in a constrained ASD environment. The chapter begins by reviewing the validated Soft Activity Model (SAM).

7.2 Review of Soft Activity Model (SAM)

The Soft Activity Model (SAM) is the emergent conceptual framework, introduced in Chapter 3, for the interpretation of the results from the data analysis. The model provides the following structured steps to interpret and make meaningful conclusions from the results:

Step 1: Problem situation expressed (Real-world problem)

AT was used, in Chapter 6, as a lens in the qualitative data analysis to express the problem situation. SAM uses AT to uncover and express tensions that exist as dualities within the problem situation. Tensions within an activity system create constraints towards achieving the goal of the activity system. It is important to unearth and understand these tensions to determine what changes must be made to achieve the object and hence the transformation to the outcome of the activity system.

Step 2: Create one or more root definitions for the problem expressed (Ideal situation)

Root definitions created describe how tensions can be resolved. In other words root definitions are solutions to the problem. Root definitions follow the form: A system must do XXX (what it does), by means of YYY (how it does it), in order to achieve ZZZ (the longer term aim).

Step 3: Construct a conceptual model (Activities to be performed to achieve ideal situation)

Each root definition is associated with a conceptual model ('ideal' situation). A conceptual model comprises activities that must be performed to achieve the solution expressed in the root definition.

7.3 Interpretation of results

The results are discussed and interpreted according to 8 themes, namely:

- Theme 1: Requirements Elicitation
- Theme 2: Establish Viewpoints
- Theme 3: Security Requirements Elicitation
- Theme 4: Security Training and Awareness
- Theme 5: Customer Involvement
- Theme 6: Security Approach (methodology)
- Theme 7: Analysis and Prioritisation of Requirements

- Theme 8: Agile RE Satisfaction

The 8 themes above have been obtained by reducing the 10 emergent themes in Chapter 6. The results for themes "Challenges to Secure Requirements Engineering" and "Best Practices for Secure Requirements Engineering in ASD" of Chapter 6 have been synthesised and analysed under the theme "Security Approach" in this chapter. The synthesis of these three themes into one theme is apt as these themes are related. As such the researcher considered it appropriate for them to be combined into one theme. The theme numbers are not necessarily the same as Chapter 6. Each theme is interpreted according to the steps of SAM described in Section (7.2) above.

Section A: Recommendations provided from best practice in ASD research

This section comprises a discussion of Theme 1 to Theme 5, namely: Theme 1: Requirements Elicitation; Theme 2: Establish Viewpoints; Theme 3: Security Requirements Elicitation; Theme 4: Security Training and Awareness; Theme 5: Customer Involvement. In Section A the recommended activities for the problems was obtained by researching best practice in ASD literature and presenting it.

7.3.1 Theme 1: Requirements Elicitation

Step 1: Problem situation

Requirements Engineering (RE) elicitation involves defining the project scope, identifying all stakeholders of the project, extracting requirements from the customer using various elicitation techniques and recording them in a requirements document (Sutcliffe & Sawyer 2013). In the survey analysis it can be inferred from Table 5.16 of Chapter 5, that the majority of ASD projects are using the conventional RE approach to elicit requirements. This may be attributed to mature RE processes over the years that are now streamlined into Agile RE practices. However the following tensions in the relationship (*subject-rule-community*) between software developers (subject) and customers (community) were uncovered from the survey analysis. The majority of software developers (75.7%) (subject) believed that the greatest difficulties experienced in eliciting requirements was when the project goals (rule) established by stakeholders (community) were unclear. De Lucia and Qusef (2010) provide an explanation of the tension experienced by software developers (subject) by suggesting that they are possibly unclear of the scope and hence what is the expected business value to the customer (community). This tension must be resolved.

Also related to difficulties experienced in eliciting requirements, the same percentage of software developers (75.7%) (subject) believed that difficulty can be experienced in elicitation, when requirements were stated in a way by customers (community) who make it difficult to verify (rule). Verification is aimed at ensuring that the system is built correctly. Possible tensions can arise between software developers (subject) and the customer (community) when developers find that the requirements elicited, lack details or are conflicting and must be verified by the customer (De Lucia & Qusef 2010). These tensions between the software developer and the customer must be resolved.

In the qualitative study through deeper probing further tensions surfaced. It was revealed that in Agile RE practice, requirements elicitation was the responsibility of the business analyst or a team of business analysts who work independently from the software development team. This finding is contrary to the ASD principles in the manifesto that states "business people and developers must work together daily throughout the project" (De Lucia & Qusef 2010). The business analyst (subject) who elicited the requirements must provide support (rule) to the software development team (community) during development. This created socio-technical tension in the relationship (*subject-rule-community)* between the developers (community) and the business analyst (subject) (Inayat & Salim 2015). Developers perceived the business analyst as someone lacking technical competence whilst the business analyst felt that they held the expert knowledge of the system and other than the customer they have the most important voice.

The problem situation for elicitation of requirements can be summarized as follows: Difficulty expressed by developers to elicit requirements when project goals were unclear or when elicited requirements were difficult to verify. In the case of the latter, missing requirements or requirements that were mistaken by the development team caused delays in projects and overrun the project budget. Secondly, socio-technical tensions were identified between the business analyst and software development team.

Step 2: Root definition

Requirements engineers in conjunction with the team must collect refined requirements that satisfy very clear project goals and are easy to verify by closer alignment to ASD values outlined in the manifesto, in order to improve Agile RE elicitation practices.

Step 3: Conceptual model

Recommended activities to improve Agile RE elicitation practice

Acceptance testing and test driven development were considered standard practice at most companies. Improvement of these activities could not be suggested as in-depth introspection of these processes were not in the scope of this study. In addition to these 2 activities the researcher recommends the use of prototypes. This approach was not widely used in companies because of the time taken to develop the prototype. However it is important to consider, as prototyping can improve the elicitation process in the long term.

A prototype is used to implement customer requirements giving them an opportunity to experiment with the system thereby revealing any omissions or errors (Sutcliffe & Sawyer 2013). In the qualitative field study, one company can be used as an example for best practice where developers spent a lot of time on building prototypes on functional requirements simply because in their opinion, the customers did not know what they wanted. The shared understanding of what is required, is so much greater when customers interact with the prototype than having an open dialogue (Sutcliffe & Sawyer 2013). Prototypes will ensure that project goals are met and they can be used to verify that the right system is built.

Another practical approach to improve elicitation is to increase Joint Application Development sessions (JAD) during the Agile RE phase. JAD sessions increase customer involvement and resolve conflicting requirements (De Lucia & Qusef 2010). This will ensure that highly refined requirements are documented.

Finally, the problem of socio-technical tension between requirements engineers and developers expressed in the problem situation requires attention. According to the social motivation theory, the interactions between team members can have an impact on the success of the team. Negative members are not motivated towards achieving the team's goals and can hinder the success of the

team. Hence core team members, such as the business analyst, must have positive attitudes and must be willing to share knowledge and information to ensure that the team is successful (Licorish & MacDonell 2014). ASD companies must ensure that team composition strategies considers that the personalities of requirements engineers must be aligned to the principles and values of ASD outlined in the manifesto namely, "Projects are built around motivated individuals who should be trusted" and "self-organising teams" (Kavitha &Thomas 2011).

7.3.2 Theme 2: Establish viewpoints

Step 1: Problem situation

Establishing viewpoints means to receive the different users' viewpoints of the system. The survey analysis Table 5.15 of Chapter 5 illustrated that respondents' beliefs reflected that the majority (73.1 %) were in agreement that users' viewpoints were established for projects. No glaring tensions were reported in this area.

In the qualitative data analysis, through deeper exploration it was found that the business analyst liaised with the client to ensure that all viewpoints are established. When developers (community) needed clarification on viewpoints (rule), the business analyst (subject) was requested to provide the support. There was tension in the relationship (*subject-rule-community)* between the business analyst and the developers as in ASD frequent face-to-face communication is required between team members and the customer. The customer must be incorporated as a team member (Inayat et al. 2015). There should not be intermediates between the software development team and the customer.

Step 2: Root definition

Top management must ensure that members, representing different ASD roles, are involved with customers to establish viewpoints, by adhering to the principles and values in the ASD Manifesto, in order to improve the establish user viewpoints RE practice.

Step 3: Conceptual model

Recommended activities to improve the RE Establish Viewpoints practice in Agile RE

In order to optimize the process of establishing viewpoints, ASD principles written in the manifesto, namely, "face-to-face conversation is the best form of communication" and "close daily

cooperation between business people and developers" is necessary (Schön et al. 2015). To elaborate, top management must ensure that the development team is directly involved with the users in meetings to establish viewpoints. According to Schön et al. (2015) it is advantageous to include developers in meetings because when they have a good understanding of the product they will be able to make better implementation decisions. This also ensures rapid knowledge sharing and allows more stakeholders to have a better understanding of the system. Overall, this measure will assist in resolving conflicts expressed by the problem situation whereby the business analyst is the main source of customer information.

Another recommendation to improve the establish viewpoints RE activity is the use of JAD sessions. Globally the use of JAD sessions are successful in establishing users viewpoints (Kumar et al. 2013). The customers and development team working on one platform will unburden the business analyst, who feels the pressure of being the custodian of knowledge. It will foster an environment of openness and transparency among stakeholders. This ensures closer adherence to manifesto values and principles. Hence companies will derive the benefits of ASD such as on-time delivery and customer satisfaction (Schön et al. 2015).

7.3.3 Theme 3: Customer involvement

Step 1: Problem situation

In the survey analysis the sum of the last three columns of Table 5.23 of Chapter 5 illustrated that 73.07% of respondents indicated that a lack of interest in security by the customer was a hindrance to secure requirements engineering. Tension was created in the relationship (*community-division of labour-object*) between the customer (community) and the security of the application (object). This tension was confirmed in the qualitative study in phase 2 of the study.

The qualitative data analysis showed that there was insufficient involvement from the customer in all Agile RE practices. The customer was consulted only periodically by the business analyst. Customers attended meetings when they were requested. They were not on site as a part of the ASD team. This practice is in conflict with the ASD methodology that prefers an on-site customer (Inayat et al. 2015). The customer (community) is only interested in the functionality of the system in terms of the business rule (rules) and very rarely made requests to developers (subject) for

security to be included. This outlined a significant tension in the relationship (*subject-rule-community*) between developers (subject) and customers (community). Developers do not educate customers on security issues and customers did not show an interest in non-functional requirements such as security. This duality prevented security and other project goals from being achieved.

Step 2: Root definition

Management must ensure active involvement by the customer in Agile RE activities, by creating awareness of ASD principles and values prescribed in the Manifesto, in order to achieve higher business value for the customer and get greater satisfaction from the customer.

Step 3: Conceptual model

Recommended activities to ensure customer involvement

Project success depends on customer involvement (Schön et al. 2017). Although in ASD face-to-face on site collaboration with the customer is recommended it very often is impractical to implement (Inayat et al. 2015). Top management must ensure that customers are active participants in meetings and reviews through awareness of ASD principles and values. Customer involvement is important because misunderstandings can be detected early and the necessary changes can be made before the cost of changes become too high.

Project managers must outline to customers, at the onset of the project, their roles especially in Agile RE activities that demand stakeholder participation such as security requirements elicitation and requirements prioritisation. Ramesh et al. (2010) advised that informal and frequent communication with the customer between meetings and reviews is a core Agile RE activity. ASD methodologies recommend at least a single person who is an expert in the domain to represent all the customers (Sillitti & Succi 2005). The benefit of collaborative development can be enjoyed by both the customer and the developer. The benefit for the developer is that it is a way to ensure customer satisfaction. The benefit for the customer is that the system is now value-driven.

7.3.4 Theme 4: Security Requirements Elicitation

Step 1: Problem situation expressed

Table 5.21 of Chapter 5 showed that the majority of respondents (66.7%) in the survey analysis indicated that the development (subject) team was responsible for identification of security requirements (rule) from all stakeholders (community). There was tension created in the relationship (*subject-rule-object*) between the software development team (subject) and the convention of making it the responsibility of the software developer (rule) to ensure security (object) during coding. The tension presented in this reactive approach to security demanded deeper probing. Therefore, the researcher sought more explanations in phase 2 of the data collection namely, the qualitative study.

The qualitative data analysis confirmed that in small/medium companies, developers that felt strongly about security during implementation (coding) identified security requirements with the consent of the project manager. This was an ad hoc approach based on the developers ethical and moral principles. Deeper analysis reflected that at these small/medium ASD companies (community) identification of security requirements during requirements engineering phase (rule) was solely at the discretion of the business analyst (subject) who often omitted these requirements. This created tension in the relationship (*subject-rule-community)* between the business analyst (subject) and the stakeholders of the project (community). This tension resulted because not enough consideration was given to security requirements in the business requirements document as security requirements evident in requirements documentation were restricted to a few standard requirements such as authentication and encryption. Furthermore, the customer was not involved. Customer involvement is an important aspect of ASD (De Lucia & Qusef 2010). The common practice noted by the researcher was that security requirements were elicited as a system enhancement when there was a violation or when the customer specifically requested it.

Step 2: Root definition

Agile RE must include a security requirements elicitation activity, by means of a requirements security risk analysis process involving all stakeholders, to ensure that critical security requirements are included in the system.

Step 3: Conceptual model

Recommended activities to include security requirements elicitation in Agile RE

In Agile RE it is recommended that customers must have meetings with project managers to discuss security (De Lucia & Qusef 2010). The best time to involve all stakeholders in security requirements elicitation is in the security risk analysis activity where the risk factors of requirements are assessed. This is prescribed in the Agile Security Manifesto (Chapter 2). This process must be done prior to prioritisation of requirements. In ASD, RE is an iterative process that takes place in the sprint planning phase of each iteration, when new requirements are elicited, making requirements security risk analysis an ongoing process (Ernst & Murphy 2012). In short to solve the problem, every time a requirement is elicited a security risk analysis must be completed.

7.3.5 Theme 5: Security training and awareness

Step 1: Problem situation expressed

The survey analysis Table 5.14 of Chapter 5 illustrated that only a small percentage of respondents (28.2%) indicated that they received code related training in application security. Figure 5.4 of Chapter 5 showed that the majority (57.7%) from the group who attended training were unsure of the value of the application security training that they received. Table 5.14 of Chapter 5 illustrated that 59% of respondents reflected that they did not receive any security training in the last 12 months. Requirements engineers (subject) from the software development team (community) were not trained (rule) in application security. There was tension in the relationship (*subject-rule-community*) between the developers (subject) and management (community) as application security training was not scheduled for them.

The qualitative data analysis indicated that in large companies *(community)* security training *(rule)* was provided for the affected employees *(subject).* In smaller companies security training (*tool*) was not prioritised for developers (*subject)* to improve their skills *(object).* There was tension in the relationship (*subject-tool-object*) between developers (subject) and application security training (tool). Further, it was apparent that team members (*subject)* were unaware that the latest trends of application security (*object)* could be sourced from security knowledge sources such as OWASP and CWE (*tool).* There was tension between developers (subject) and security knowledge

sources (tool). Developers did not consult security knowledge sources that were meant to provide security information to developers. This duality hindered application security.

Step 2: Root definition

Requirements engineers must be kept abreast of the latest trends in application security, by means of attending annual training and regular consultation with information from security knowledge sources, to improve knowledge in application security.

Step 3: Conceptual model

Recommended activities to ensure training in application security

Security training for requirements engineers will ensure that security is given a thought in the development process. Since security knowledge always needs to be updated annual training is suggested for requirements engineers. Many researchers concur with this recommendation (Salini & Kanmani 2011; El-Hadary & El-Kassas 2014). Budget for security training of requirements engineers and other stakeholders must be made available by top management. Senior managers must ensure that developers are informed of the latest trends and guidelines in security from security knowledge sources such as OWASP, CWE, CERT and ISO Security Standard (Elahi et al. 2011). E-mail alerts from these knowledge sources can be sent to developers so that they are always kept informed of the latest security trends. Implementing these recommendations bring closer alignment to the security principles of the Agile Security Manifesto.

Section B: Recommendations provided from this study as well as unique contributions by researcher

This section comprises a discussion of Theme 6 to Theme 8, namely: Theme 6: Security Training and Awareness; Theme 7: Analysis and prioritisation of requirements and Theme 8: Agile RE satisfaction. Section B outlines the researcher's unique contributions in dealing with recommended activities for the problems. The researcher's contributions in each theme are as follows:

- Theme 6: Three step just-in-time security approach and the automated fuzzy tool for prioritisation of security requirements.

- Theme 7: Process model to guide practitioners on secure requirements engineering in Agile Software Development and the automated fuzzy tool for prioritisation of normal requirements.
- Theme 8: Product model for practitioners who have their own RE processes but require guidance on the pillars for secure requirements engineering.

7.3.6 Theme 6: Security Approach (methodology)

Step 1: Problem situation expressed

The survey analysis Table 5.15 of Chapter 5 illustrated that only 34.6% of respondents (subject) indicated that security experts (tool) were identified in their projects to advise on security approaches (object). By inference the majority of companies did not have dedicated staff to deal with security issues and tension is created in the relationship (*subject-tool-object*) between the software developers (subject) and application security approaches (object). Also further tensions surfaced when a large percentage of respondents (community) indicated that change in requirements (62.82%) and the limited time given to complete a project (61.67%) (division of labour), as shown in Table 5.23 of Chapter 5, were the greatest challenges to inclusion of security in the system (object). The tension created here in the relationship (*community-division of labour-object*) was between the developers (community) and their workload (division of labour).

The qualitative study was used to seek a deeper understanding into why no formal security approach was in place. This practice or lack of it created serious tensions in the activity system making the crowning achievement of secure systems development difficult to achieve. Respondents confirmed the following challenges:

- There was a high workload (division of labour) for development teams (community) to deliver features in short sprints that lasted approximately 4 weeks (object);
- Developers (subject) indicated that learning new technology (tool) took precedence over security concerns during development as features must be delivered (object);
- Developers (subject) indicated that there is a constant demand for new features not security (rule) from customers (community);
- Customers (community) are always changing requirements (division of labour) in terms of business needs (object);
- Developers (subject) indicated that there is not enough time given for development (rule) by

management (community);

- Developers (subject) indicated that the high costs to implement security prevent small to medium customers (community) from including it in the system.

The content analysis of interview transcripts and analysis of the business requirements documentation indicated that in most companies security was absent or unclear in Agile RE practices, with most ASD companies admitting that they did not have a security approach. The security approach was reduced to the following:

(i) Developers (community) based on their own ethical principles requested for security approaches during coding (division of labour*)* to protect the system (object). Tension existed in the relationship (*community-division of labour-object)* between the developer and the increase in his workload (division of labour).

(ii) Customers (subject) requested security during requirements engineering (rule*)* from the business analyst (community). Customers experienced tension requesting security as they only had a budget for functional requirements that they believed will add value to their business. The tension for the software development community was that security increases cost estimates and workloads.

(iii) Business analysts (community) could include security through consultation with the client and client security policies during requirements engineering (division of labour*)* to prevent vulnerabilities (object). Tension exists in the relationship (*community-division of labour-object)* between the business analyst and his novice security knowledge (division of labour). The researcher noted after viewing the requirements document that only a few generic security requirements such as authentication and access control were factored in. It was evident from interviews that the business analysts did not consider themselves experts in security.

Step 2: Root definition

Agile RE must implement a security approach based on best practices for security from the research study and ASD literature that is convenient enough for the average software development team to use in order to elicit critical security requirements.

Step 3: Conceptual model

Recommended activities from fieldwork to improve security in Agile RE

(i) Developers must ensure that customer involvement in security must be mandatory. Customers must be educated on their role in the development;

(ii) Developers must request a dedicated sprint involving all stakeholders to address security requirements at the start of the project and then must continue to place security on the agenda during sprint planning meetings;

(iii) When it comes to databases both SQL databases (relational databases) such as MySQL and NoSQL databases such as MongoDB are exposed to the same vulnerabilities and must be secured. Secure requirements must be written to regulate user access to the database (Oracle Database Security Guide 2003).

(iv) Project managers from small to medium companies can outsource security requirements to dedicated security companies to cater for security needs;

(v) Developers must use security patterns and security frameworks (tool) to combat security vulnerabilities. Security frameworks are available for access control, authentication and authorization to databases (ComputerWeekly.com 2017). This can free up the developer's time to concentrate on functional requirements.

(vi) Project managers must dedicate time for developers during the work day to spend on updating themselves on trends in application security (object). They can utilise this time to get up to speed on the latest security trends from knowledge sources such as Open Web Application Security Project (OWASP), Computer Emergency Response Teams (CERT) and Common Weakness Enumeration (CWE).

(vii) Proposed lightweight security approach for Agile RE:

In ASD, RE is rapid, with activities taking place much faster than mainstream RE processes. RE is also iterative. Therefore, the researcher recommends a lightweight approach over more complex formal methods discussed in security literature. After careful analysis of a number of approaches in security literature, the researcher encountered the following 4 step approach, proposed by Tondel et al. (2008).

- Security objectives
- Assets identification
- Threat Analysis

- Documentation of security requirements

The researcher used the above security approach as a starting point to model a new approach that could be implemented into a light weight process model such as ASD. In proposing a new approach, Step 1 above is omitted because RE in ASD is ongoing and it makes little sense to repeat "setting security objectives" at each iteration. Further, security policies, legislation and standards as related to the product must be conveyed at the feasibility study for the project and not in the iterations. Step 4 is also omitted because security requirements can be documented with normal requirements in ASD. The researcher proposes an efficient 3 step, "just-in-time" security approach suitable for Agile RE, namely:

1. The valuation of assets and resources of the software being developed: Inspect functional requirements to identify assets. Examine requirements from an attacker's point of view one at a time.
2. Security Risk Analysis: Consider the threats and vulnerabilities for each asset identified. Determine the impact of the threat. There are a number of approaches suggested in literature for example DREAD (Salini, P & Kanmani 2012), STRIDE (Tondel et al. 2008) and so forth. The researcher proposes use of the newly constructed automated fuzzy tool to categorise and prioritise security threats based on multiple stakeholders and multiple decision making criteria such as cost, time and cost of damage.
3. Elicitation of Security Requirements: Identify security requirements based on threats. The team can select through consensus a limited number of the most highly ranked security requirements to include as candidate requirements. These chosen security requirements must now be packaged with the list of refined candidate requirements in the product backlog for possible implementation in the next iteration. The decision to choose the security requirements for implementation will be determined by the RE prioritisation process.

This approach is sufficient to ensure that critical vulnerabilities are addressed in software releases. This approach is also aligned to the four principles of the Agile Security Manifesto described in Chapter 2. The approach does not guarantee resolution of all application security issues but is light weight enough to ensure that regular ASD developers are able to implement 'just-in-time' security

at a reasonable level in their projects. It is suited to the regular developer in a small to medium company and at the same time can satisfy large organizations in terms of their security needs.

7.3.7 Theme 7: Analysis and prioritisation of requirements

Step 1: Problem situation

The survey analysis Table 5.19 of Chapter 5 illustrated that the majority of the respondents (96.16%) agreed that the best time to negotiate on requirements with the customer is during requirements prioritisation. This stands to reason, after all, it is this process in Agile RE that ensures that the most valuable features get implemented for the customer (Sillitti & Succi 2005).

In the qualitative study several tensions were confirmed. The qualitative data analysis indicated that prioritisation took place in requirements analysis stage and confirmed that not all stakeholders were involved in the prioritisation process. This created tension in the activity system. The customer (community) did not physically play a role (division of labour) in the prioritisation of requirements (object). There existed tension in the relationship (*community-division of labour-object*) between the customer (community) and the customer's role (division of labour) in the prioritisation of requirements. Further tensions indicated that there was no specific prioritisation technique utilised in Agile RE practice and requirements were more often than not prioritised by the project manager alone and only in some instances the prioritisation took place through consensus in conjunction with the business analyst.

Step 2: Root definitions

1. The customer together with key members of the development team must rank requirements using a multi-criteria, multi-decision making tool to ensure that the highest business value is achieved by more important features being delivered first.
2. A process model detailing the requirements prioritisation process within secure Agile RE activities, synthesized from literature and the findings of the study, to support regular software development companies in their day to day practice.

Step 3: Conceptual model

<u>Recommended activities to improve analysis and prioritisation of requirements</u>

Conceptual Model for Root Definition 2

Requirements Analysis and Prioritisation take place after elicitation at the beginning of every iteration in ASD. The current Agile RE practice in companies is that the business analyst and project manager, informally, analysed requirements, individually or collectively. A more inclusive, team approach to requirements analysis and prioritisation is recommended. De Lucia and Qusef (2010) suggested an approach that can be used. Firstly, the analysis of requirements must take place collectively. Before the start of every iteration, after new requirements are elicited, they must analysed. All candidate requirements, new and old must, be analysed, formally or informally. Thereafter, they must be validated by the software development team at a formal meeting. Formal analysis can take place by using UML notation or informal analysis can occur by inspecting user stories written in a natural language. If the team decides that after proper examination, the requirement is invalid then this requirement must be placed on hold until it becomes valid. An invalid requirement must be resolved at a stakeholder review meeting such as a JAD workshop. All requirements that are valid must now proceed to the product backlog. The new list of candidate requirements must be prioritised to ensure that the most valuable features are implemented in the iteration. Owing to the nature of ASD, this process is ongoing at the beginning of each iteration.

The prioritisation process will assign requirements for the next iteration to be implemented. The current practice of autocratic decision making as revealed in the study results must be changed to a more inclusive approach to ensure best practice. In this regard, the newly created automated fuzzy tool is recommended to ensure that multiple stakeholders make decisions on requirements using multiple decision making criteria. Requirements ranked at the top of the prioritised backlog as output by the fuzzy automated tool are selected for implementation.

The automated fuzzy tool provides the following benefits to requirements engineers:

- Decision making is more accurate and precise (ratio data) ensuring that requirements that are truly valuable gets implemented first.
- Promotes collective decision making by ensuring that a number of stakeholders must make their input before a final decision is reached.

- Prevents biases and autocratic leaders from influencing decision making. This was the common complaint in the field work.
- An additional application of the tool is its use in security. It can be used in security risk analysis to prioritise security requirements.
- Besides the prioritisation of the product backlog the automated fuzzy tool could be used to prioritise the sprint backlog, if another round of decision making is necessary.

The prioritisation of requirements must occur immediately after the analysis process. A new model (explained under root definition 2) created by the researcher as an output of the study shows the ordering of RE processes in Agile RE.

Conceptual Model for Root Definition 2

The researcher in drawing from the findings of this research study produced a process model depicted in Figure 7.1 that describes activities and processes for secure requirements engineering in constrained ASD environment. The model provides evidence based guidelines on how ordinary software development companies, implementing ASD can ensure an adequate level of security in RE.

Figure 7.1, depicts a process model for secure requirements engineering in a constrained ASD environment. The figure depicts processes for secure Agile RE. Three very important RE processes are outlined, namely, initial Agile RE activities boxes (1-4, 6, 11); Security requirements engineering boxes (7-9) and finally requirements prioritisation boxes (12-14). In a rapid ASD methodology, RE takes place iteratively. Figure 7.1 shows the RE activities that take place at the start of every iteration. Although distinct boxes are used, the Agile RE activities are not clearly separated and takes place much faster than traditional RE processes.

The model is aligned to the principles of the Agile Security Manifesto in the following way: it relies on developers taking accountability for security by bringing the ‘just-in-time’ security approach into the RE process. The approach ensures that security is not an afterthought. The security approach also makes use of risk mitigation ensuring that security is not ad hoc.

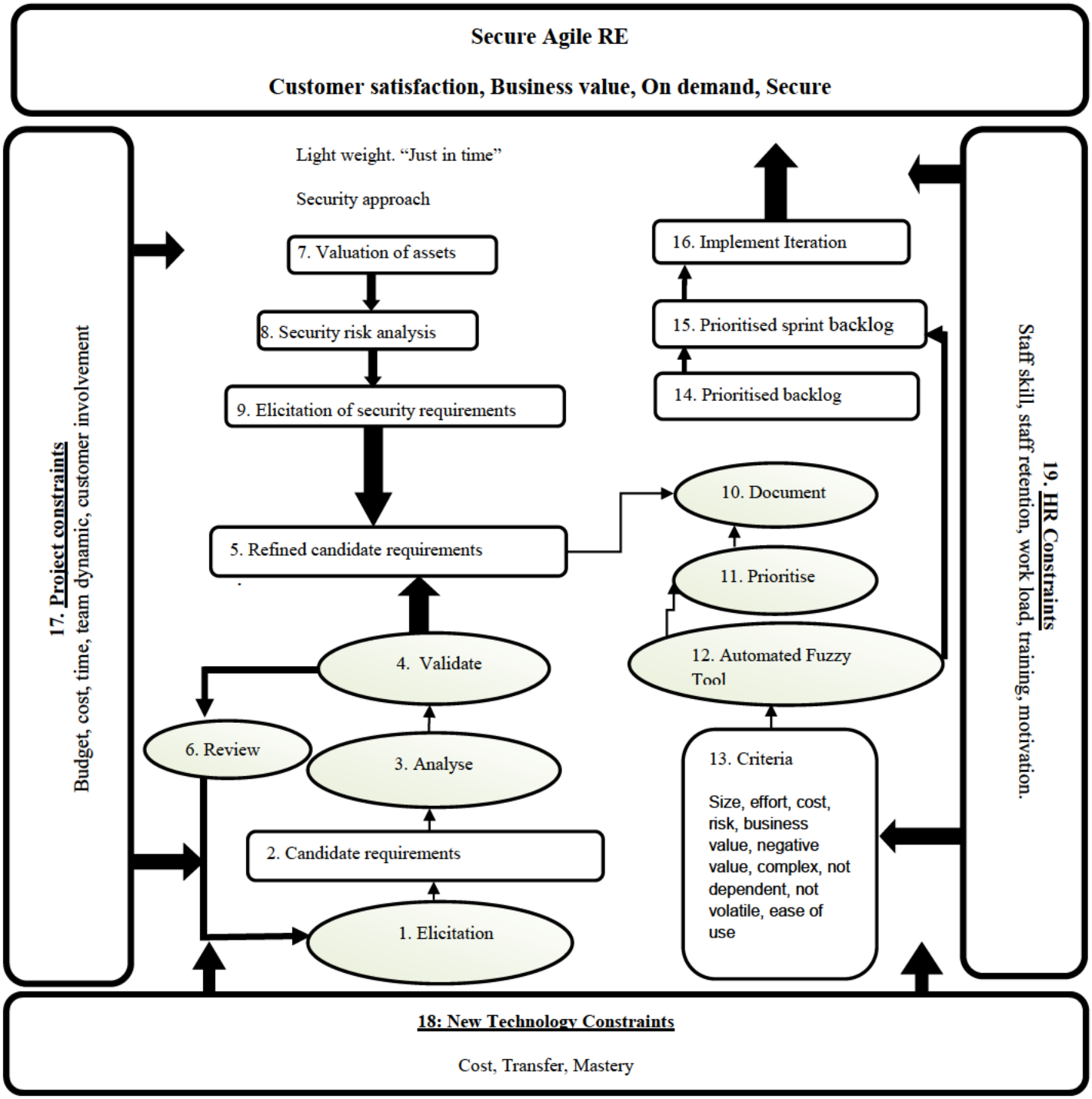


Figure 7.1: Process Model: Secure requirements engineering in a constrained ASD environment
Source: Researchers own construction

The association of key components of the process model namely normal Agile RE activities, Security Approach and Prioritisation of Security Requirements predicting secure requirements

engineering have been tested using structural equation modelling as shown in Section 5.11. A brief discussion of Figure 7.1 is given below.

Agile RE (1-4, 6, 11): Elicitation and Analysis are discussed in earlier themes. Box 6 is the Review box. This is the review process and it follows the validate process. During validation, requirements are inspected for consistency and completeness. When there is a problem with the requirement, it is sent for review. This is a manual process and must involve stakeholders with formal recordings of problems and corrections. De Lucia and Qusef (2010) recommends JAD sessions to sort out problems of consistency with requirements.

Security Approach (7-9): Prior to 7-9, it is important to identify the underlying security goals of the company during the feasibility study, prior to the iterations. This is important as requirements must be met to comply with legislation and policy (Tondel et al. 2008). During the iteration once normal requirements are identified, valuation of assets, security risk assessment and elicitation of security requirements must be completed. The practices within these processes are explained in theme 6 of Chapter 7 under security approach.

Requirements prioritisation (12-14): Once the list of refined candidate requirements consisting of critical security requirements and normal requirements are identified, the requirements prioritisation process can proceed. A novel prioritisation technique whose foundations are from fuzzy logics and the TOPSIS approach called fuzzy TOPSIS is suggested. The technique is introduced through the newly constructed automated fuzzy tool. The results of the ranking are presented as ratio data (decimals) showing not only the positioning of the requirement in comparison to other requirements but also give the magnitude of the ranking. Once the list of requirements is prioritised, the requirements at the top of the prioritised backlog are selected for implementation in the iteration. Box 13 consists of a choice of benefit criteria. The criteria given were suggested from this research study. This means that the list provided is not an exhaustive one.

Constraints (17, 18, 19): Constraints emerging from the study are categorised as project constraints, HR constraints and New Technology constraints. The requirements prioritisation and

reprioritisation in Agile RE must be utilised to minimise or combat the effects of constraining factors in Agile RE.

7.3.8 Theme 8: Agile RE Satisfaction

Step 1: Problem situation expressed

The qualitative data analysis indicated that informants expressed satisfaction of RE processes applied to their ASD projects however requested that the following tensions be resolved within the activity system:

(i) Developers must ensure greater customer involvement (community) in non-functional requirements (division of labour) such as security to ensure customer satisfaction in the ASD product (object);

(ii) Special security knowledge combined with all round development knowledge is lacking in Requirements Engineering. Requirements engineers (subject) must be more thorough about requirements documents and seek clarifications from the customers and technical staff when required (rule) to prevent confusion and frustration from developers (community) during implementation;

(iii) Developers (subject) need a high level view (tool) of the system (object);

(iv) Social interactions and personality traits of each team member (subject) play an important role in requirements engineering (rule) and more often than not the data gathered shows that it must be improved within the team (community);

(v) Staff retention (rule) of requirements engineers (subject) poses a threat to the success of the development process within an organization (community).

Step 2: Root definitions

1. Companies must improve RE practices, by observing evidence based guidelines for secure RE in ASD, towards secure software development.
2. A product model for requirements engineers, synthesized from the research study and ASD literature, to guide secure requirements engineering.

Step 3: Conceptual model

Root Definition 1:

Schön et al. (2015) recommends that business analysts must have a specialist knowledge of the product. Selection of suitable candidates in this pivotal role is vital to a methodology that bases its successes on collaborative development. A business analyst must have specialist domain knowledge as well as be equipped with skills in software development.

Staff retention poses a huge risk in software development and must be managed. When key staff leaves, costs are incurred hiring and training new staff. A business analyst often collaborates with customers and when this staff member leaves it can seriously compromise relationships with customers. Companies can even experience losses due to losing customers. Companies that are most profitable retain their staff. Small and medium companies that do not have the budget to provide attractive pay packages must find other rewards to retain staff. Senior management must find creative ways to keep staff loyal and motivated. One example is to provide annual vacation trips for top performers. This once off incentive will motivate staff to perform well and does not come at great cost to the company (Volper 2012).

To ensure that the ASD approach reaps the intended benefits the team dynamics must be positive. The results show that at many ASD companies this requires attention. Poor team dynamics waste time, money and resources. Team Technology (2017) recommends a diagnosis of the problem as this will differ from team to team. Once the problem is located, an appropriate strategy can be put in place to improve team dynamics.

Root Definition 2:

The researcher developed a product model for secure requirements engineering in a constrained ASD environment. The product model was synthesised, from the analysis of the findings of the study. Figure 7.2 shows that in order to implement secure requirements engineering, foundational elements must be in place supported by four pillars.

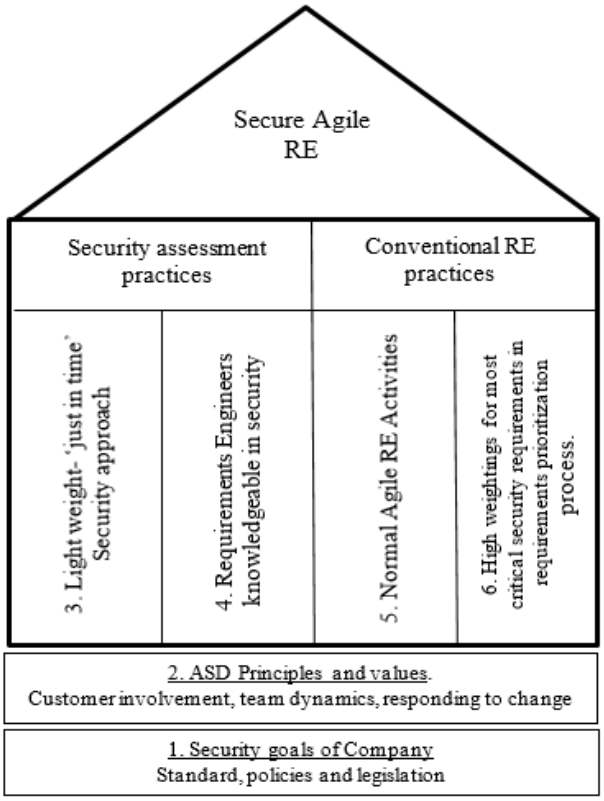


Figure 7.2: Product Model for Secure Agile RE

Figure 7.2 depicts the product model for secure Agile RE. The model is constructed from Figure 5.7 depicting the path analysis for the structural equation model. The factor analysis results (Table 5.9) on secure RE practice showed two factor groupings for secure requirements engineering namely, 'security assessment practices' and 'conventional requirements engineering practices. These are depicted in Figure 7.2. In order to ensure secure Agile RE, strong foundational elements are required. Standing on this foundation, the process is propped up by four pillars. According to Figure 7.2 the product model comprises the following:

Foundational Elements

1. *Security goals of the company*: standards, policies and legislation
2. *ASD principles*: customer involvement, team dynamics, responding to change

Pillars

1. Lightweight *security approach*: Discussed under Theme 5.

2. *High competence level of business analyst*: Subject matter expert to elicit requirements, knowledge and training in security and team player.
3. *Routine Agile RE Activities*: Elicitation, Analysis, Review, Ranking, Documentation, Management. These are depicted in the process model (Figure 7.1)
4. *Critical security requirements*: Stakeholders must be willing to elevate the priority of the most critical security requirements in the requirements prioritisation process. This will ensure that security gets implemented.

The product model has been validated using structural equation modelling in Section 5.11. The product model emphasises the intentions towards the crowning achievement of secure software development by ensuring that security is incorporated early in the software development process. The model provides a blue print and compass for secure Agile RE at a regular software development company. Finally it promotes some sense of 'standardisation' in a flexible approach and gives customers a sense of comfort around security.

7.4 Chapter summary

In this chapter the results of the study were interpreted using the Soft Activity Model under 8 themes. The steps followed were: expressing the problem situation, creating a root definition for the problem and finally construction of a conceptual model. In response to the findings several recommendations were suggested for improvement. The chapter also presented a new process and product model for secure requirements engineering in a constrained Agile RE environment that will serve as a guideline for practitioners. In the next chapter the summary, conclusions and implications of study are presented.

CHAPTER EIGHT: SUMMARY, CONCLUSIONS AND IMPLICATIONS OF STUDY

8.1 Introduction

The previous chapter focused on the interpretation of results and provided many recommendations for improvement. This is the final chapter which summarises the study, presents the conclusions and discusses implications of the study. The chapter exposes limitations of the research as well as suggests opportunities for further investigative studies at the post-doctoral level. The researcher's unique contribution to the body of knowledge is outlined as well. The chapter concludes with recommendations for future research in this area and a chapter summary.

8.2 Summary of the study

A quick reference thesis concept bank is provided at the beginning of the thesis to assist the reader with explanations to unfamiliar concepts when required. Chapter One provided an introduction and background to the study. The main research question was: ***How do Agile RE practices impact the security of Agile Software Development products?*** The three research sub questions were:
RSQ1: What is the Agile RE practices in the software development industry?
RSQ2: How do software engineers control requirements in Agile RE?
RSQ3: To what extent are secure RE processes implemented in Agile RE practices in industry?

The significance of the study is summarised as follows: to gain more insight in Agile RE activities that will provide guidelines for managers and decision makers on RE process improvement strategies; uncover challenges that teams face incorporating a security approach into Agile RE; develop an automated fuzzy tool to help requirements engineers control clients requirements; develop evidence based guidelines for secure RE within a constrained Agile environment and finally show how the use of security metrics tools can improve application security.

In Chapter Two the literature survey as related to the main research questions and research objectives were presented. Based on ASD literature the following were established: requirements engineering requires more research focus (Inayat et al. 2015), RE is a critical success factor for ASD projects (Sheffield & Lemetayer 2013), there are various approaches to requirements engineering in ASD (Inayat et al. 2015), there are various methods/frameworks in literature for security requirements engineering

(Elahi 2009) and finally in Agile RE the requirements prioritisation process is responsible to control requirements going into and exiting an iteration. The use of fuzzy TOPSIS as an automated tool for ranking requirements addressed a research gap. A worked example to illustrate the use of the fuzzy TOPSIS algorithm in ranking requirements was presented. The requirements prioritisation process is not only significant for normal requirements but also has a direct impact on the security of an application. The chapter concluded by discussing vulnerability testing such as static and dynamic analysis tests.

In Chapter Three the theoretical frameworks used in the study were presented. They were Activity Theory, Design Science Research Methodology, Soft Systems Methodology and Technology Acceptance Model. A conceptual model to best understand concepts and constructs of this study was created using parts of Soft Systems Methodology and Activity Theory. The conceptual model called Soft Activity Model was validated by a mini Delphi process. This chapter also presented the research design. This research was an explanatory sequential mixed methods study. The chapter described data collection methods, tools and analysis. The chapter concluded with a discussion of ethical considerations for the study.

Chapter Four presented the automated fuzzy tool. This tool was constructed using Design Science Research Methodology. The chapter gives a step by step account on how the tool was constructed. The tool is demonstrated using requirements from a live project at a study site.

Chapter Five presented the results of the quantitative study. The results were presented in the form of frequency distribution tables and graphs. Hypothesis tests and correlation analysis were completed to discover trends and significant relations in secure Agile RE. The regression model on secure RE practices was presented with eleven impacting factors. A new more focused regression model with fewer impacting factors was constructed. Structural equation modelling was used to validate the relationship between independent variables and secure requirements engineering. Finally, dynamic Analysis Security Tests (DAST) was conducted randomly to determine the security of the ASD product. The security of an ASD product was then compared to the scoring for secure RE practices of the product whilst in development. The results showed a correlation between the level of security measures taken during development of the project and the

actual security of the product. More vulnerabilities were found in projects that were given a low score for secure requirements engineering.

Chapter Six presented the results of the qualitative study. Qualitative data was collected in phase two of data collection. The qualitative study extended the quantitative study by seeking further explanations of study results as well as triangulating and strengthening study results. Critical review of ASD practices in requirements engineering paved the way for best practices as well as highlighted potential problems that needed to be addressed. The gathering of security requirements and how security concerns are addressed in ASD was also the focus of the study. The qualitative data was thematically analysed.

Chapter Seven interpreted and discussed the results using the Soft Activity Model under 8 emergent themes. Several recommendations were suggested for improvement based on the findings of the study. These recommendations were obtained from best practices in this study as well as other research studies. A new process and product model for secure requirements engineering in a constrained Agile RE environment that will serve as a guideline for practitioners was presented.

8.3 Conclusions of the study

In terms of the aim of the study and key research questions distinct conclusions were arrived at. This research study is underpinned by the four Agile Software Development principles from the ASD Manifesto presented in section 2.3 and the four security principles from the Agile Security Manifesto presented in section 2.3.2.

8.3.1 Research Question 1 (RQ1)

The first question was on what are the Agile RE practices in the software development industry. The findings are summarised below.

The survey reflected favourable results for RE processes in Agile RE practices with majority of respondents indicating the following for core RE practices:

- Objectives of the web application are identified (76.92%);
- All stakeholders are identified (76.49%);
- All viewpoints are established (73.08%);
- Assets of the system are identified (73.08%);
- Non-security goals identified (52.56%);
- Normal requirements are identified (89.74%) and
- Non-functional requirements are identified (71.79%).

Although standard RE practices were observed for requirements elicitation activity, an in depth investigation in the qualitative study revealed the following: the business analyst elicited all requirements and the team was not involved; the business requirements document was generated by the business analyst; some customers had a business analyst on their side which made the task of eliciting requirements much simpler; the customer was less active in the development process and finally the business analyst worked independently of the ASD team. The findings for the establish viewpoint activity were that the business analyst established all viewpoints. Viewpoints were captured in the business requirements document. The business analyst clarified users' viewpoints when necessary. In regard to customer involvement the findings were that poor customer involvement was consistent.

8.3.2 Conclusions drawn (RQ1)

Based on the findings from the data gathered and presented, the conclusion drawn is that RE processes are observed as core activities in Agile RE. Brainstorming and user stories are the two most popular approaches used for elicitation of requirements. Requirements were elaborated by generating use case diagrams. Requirements are analysed using structured requirements definition. The RE process is rapid and there is no clear separation between activities, for example analysis of requirements takes place very soon after the elicitation of requirements. The business analyst is considered to be the main requirements engineer and is responsible for the business requirements document. Customers were not based on site and did not play a significant role in requirements engineering. Overall it can be concluded that although there was close alignment with RE processes, Agile Software Development principles and values as outlined in the Agile Manifesto have been violated in practice.

8.3.3 Research Question 2 (RQ2)

The second question was on how software engineers control clients' requirements in Agile RE. Client requirements are managed by the requirements prioritisation process in Agile RE. The findings for the qualitative study were as follows: no team involvement in requirements prioritisation; no proper prioritisation techniques used; lack of direct involvement by customer in the requirements prioritisation process; there is no specific prioritisation tool; prioritisation generally takes place through consensus and important decision making criteria from the research were, namely, opportunity cost; business value and requirement dependence. The proposed fuzzy RE software tool for stakeholders will contribute towards RE process improvement ensuring that the desired quality output is produced with the available resources.

8.3.4 Conclusions drawn (RQ2)

Clients requirements are controlled by the requirements prioritisation process. This process had no team involvement. Companies did not use any specific requirements prioritisation technique. The prioritisation of the requirements was based on the project manager's decision and there was no consultation with the customer. The process was unilateral and not in keeping with Agile Software Development principles. Greater improvement must be shown in this area of Agile RE, to give customers better value. The new fuzzy automated tool was evaluated. The results were fairly

positive for use in Agile RE. The artifact will be subject to further trials and the feedback obtained will be factored into the tool for improvement.

8.3.5 Research Question 3 (RQ3)

The third question was on the extent that secure RE processes are implemented in Agile RE. The majority of respondents (66.7%) indicated that the software development team was responsible for identifying security requirements in projects. The findings from the qualitative study on security requirements elicitation were, namely, insufficient security requirements identified; security requirements only identified at coding when there is a glaring security issue; Security requirements identified are generic (not customised); customer requirements are only specified when the company experienced a security violation in another project and they want to mitigate upfront in the new project and finally security requirements are identified after a system violation.

With regards to what security approaches are used in Agile RE, the results included: the absence of formal security methodologies in Agile RE; approach to security was ad hoc; developers requested security approach during coding; business analyst was responsible for ensuring that critical security requirements were included; the majority of respondents (79.5%) indicated that the developers validated requirements and changing requirements was confirmed as the biggest challenge to secure requirements engineering.

With regards to security training, the majority of the respondents (59%) did not receive security training in the last 12 months. The qualitative study results confirmed that application security training was not prioritised. The qualitative study results showed poor awareness of security knowledge sources among the software development team.

The regression model showed that “requirements inspection to identify potential threats” had the most impact on secure requirements engineering. Finally several dynamic analysis tests were conducted on the live projects and showed correlation in terms of RE practices and the security of the product. It was concluded that more secure RE practices resulted in fewer vulnerabilities of the end product.

8.3.6 Conclusions drawn (RQ3)

No formal security approach was utilised in Agile RE. Secure requirements were added only after a violation. The process was reactive rather than proactive. The business analyst was tasked to elicit security requirements while developers were excluded from requirements engineering. The business analysts were not trained or experienced in security and were very reluctant to include security.

The software development teams did not attend security training and were very unaware of security information sources and standards. Dynamic analysis testing reported a host of vulnerabilities including vulnerabilities from the OWASP top 10. These findings allude to a high number of vulnerabilities in terms of application security. The systems developed are easy targets for attacks by hackers unless some action is taken. Apart from the lack of use of security frameworks there was poor alignment with the Agile Security Manifesto. The situation is at best unfavourable for secure requirements engineering. Recommendations mentioned in Chapter Seven must be considered to improve the current situation.

8.4 Implications of the study

This research study has several implications for researchers and industry practitioners. Firstly, ASD practices in industry show evidence that a hybrid approach is used in terms of Agile Software Development. Practitioners are incorporating ASD principles and values from the Agile Manifesto that are easy for them to implement and ones that bring them most value. To use an example of two principles in the manifesto to illustrate this, short iterations are included by most companies but the on-site customer is not. This means that development is conducted by using a mix of traditional and ASD, not pure Agile Software Development. Hence companies cannot derive maximum benefit intended from using the methodology.

The findings of this study showed positive results for the use of the automated fuzzy tool for controlling or managing, through a ranking process, client's requirements into and out of implementation iterations. The tool can be used by Agile RE practitioners to ensure that the requirements prioritisation process is inclusive of all stakeholders and decisions are based on

logical principles. Hence the automated tool fits well with the Agile Software Development philosophy.

Security has sparked research interest but is yet to progress to a point to be incorporated as a normal Agile RE practice. The research has shown that low emphasis on security during requirements engineering will impact the security of the Agile Software Development application. Dynamic analysis testing of live products showed vulnerabilities proving low emphasis on security during development. Further several challenges for incorporating security were obtained from the study. The researcher has suggested several recommendations in Chapter Seven stemming from best practice to improve security in Agile RE. Practitioners must adopt as many recommendations as they can. The researcher has also provided a simple three step, 'just in time' security approach for Agile RE that can be implemented by a regular software development company. Further, the researcher has constructed a process model and a product model using evidence based guidelines from the research for secure requirements engineering. These interventions are intended to improve security within Agile RE and is a positive step for secure systems development.

Adhering to these security practices could have prevented for example the Ransomware attack in Europe in 2017. A Ransomware named WannaCry infected millions of systems by encrypting files and requesting payment in bitcoins. A typical way this malware entered the system is through an unvalidated redirect or forward. This is vulnerability ten in the OWASP top ten vulnerabilities. This weakness could have been prevented in the application layer by the application developer ensuring that client input is approved and furthermore confirming the URL being referred to, is really an endorsed target URL (Gurubaran 2017).

8.5 Summary of researcher's unique contributions to the body of knowledge

This first unique contribution is the emergent conceptual model called Soft Activity Model (SAM). SAM was designed and concept validated through a mini Delphi process. This conceptual framework was used as a lens to express the problem situation; provide a systematic way to present a solution to the problem and allow for the suggestion of steps to be followed for achieving the

specific solution. This conceptual model can be used as lens or a paradigm to inform research methods and data collection.

The literature shows eleven factors that impact secure requirements engineering. A new regression model with a reduction in impacting factors from eleven impacting factors to six impacting factors for secure requirements engineering is a contribution to the body of knowledge. Practitioners now have a more focused model with fewer variables to control when conducting secure requirements engineering.

The third contribution emerging from the research is the light weight security approach that comprises three steps for the average Agile Software Development company. The steps are as follows: the valuation of assets and resources of the software being developed; security risk analysis and elicitation of security requirements. It is simple enough to be used by the typical software developer. This approach directly impacts secure requirements engineering as shown in Figure 5.7 (structural equation model) and the proposed regression model in Section 5.10.2.

An output of the research is an online web application called the automated fuzzy tool. This tool was constructed using a high level programming language to support stakeholders in the software development industry. Two applications of the tool in application development are firstly to rank requirements in the product backlog and secondly to prioritise security requirements during the security risk analysis process. The construction of the automated fuzzy tool is outlined in Chapter Two.

The final contribution is a process model for practitioners that provide guidelines on how to conduct secure requirements engineering in a constrained Agile Software Development environment. The model was developed from the findings of the research. The model includes the three step security approach as well as the use of the automated fuzzy tool for ranking requirements. The researcher also proposed a product model to guide Agile Software Development practitioners in their own processes towards secure requirements engineering. Relationships in this model was tested and validated by structural equation modeling as shown in Section 5.11.

8.6 Limitations of the study

In understanding the major conclusions and findings of the study the limitations of the study must also be considered. The literature review focused on research papers written in English. Other relevant studies written in other languages have been omitted as this was not included in the scope of the study. The challenges of incorporating security in Agile RE have been reported in the findings. These challenges were reported as: insufficient application security training; no access to security knowledge sources; high workload, other critical learning in technology innovation were prioritised; fixation on functional requirements, constantly changing requirements, high paced ASD environment and small companies did not have budget for security. The reasons underlying each of these challenges require further in-depth analysis. This could not be carried out in this study and hence can be seen as a limitation of the study.

8.7 Future research

A 'just-in-time' automated fuzzy tool was developed as an output of the research, for ranking requirements. This study showed that the evaluation of the online web application by potential users was positive. Improvements based on the findings of more assessments on the web application are planned as future research. Also, further research directions for the web application will be pursued, namely: research on use of the app in other software development methodologies as well as research on more application areas for the web application.

Researchers have shown that there is a need for empirical studies in Agile RE. This study focused on Agile RE with no mention of a specific ASD methodology. It will be of interest to the research community and practitioner's alike if more research on Agile RE is conducted using a particular ASD method, such as Scrum or XP. Further studies could be conducted by assessing for example Scrum RE activities. A security approach and software requirement prioritisation approach in RE will be of interest to the Scrum community (Achimugu et al. 2014).

Future research could consider distributed and global software development teams. There is a need for more research in Agile RE on distributed projects (Inayat et al. 2015). The challenges and best practice from research with distributed ASD teams will allow the Agile Software Development community to gain better insight. Researchers steer away from this kind of research owing to

difficulties in acquiring data. The outsourcing of features for the systems development may prove to be a convenient approach for incorporating security requirements and security features. However this needs to be assessed empirically.

Finally another much needed research area in Agile Software Development is customer involvement. This study has shown poor customer involvement. In an ASD environment, owing to the very nature of the methodology, it is not possible to prescribe a structured framework for customer involvement except to say that it is a very significant RE process to ensure that any assumptions during the software development process are validated. Researchers recommend further research in this under researched area in the domain of Agile Software Development (Schön et al. 2017).

8.8 Chapter summary

This chapter concludes the study. The chapter is intended to give a quick synopsis of the study. The chapter begins by providing the reader with a summary of the study. Thereafter conclusions are drawn based on the research questions of the study. Implications of the study are discussed. The chapter concludes with the recommendations for future studies.

REFERENCES

Achimugu, P., Selamat, A., Ibrahim, R. and Mahrin, M. N. 2014. A systematic literature review of software requirements prioritization research. *Information and Software Technology*, 56 (1): 568-585.

AL-Ta'ani, R. H. and Razali, R. 2013. Prioritizing Requirements in Agile Development: A Conceptual Framework. In: Proceedings of *4th Interntional Conference on Electrical Engineering and Informatics.* Selangor, Elsevier Ltd, 733-739.

AlBreiki, H. H. and Mahmoud, Q. H. 2014. Evaluation of static analysis tools for software security. In: Proceedings of *Innovations in Information Technology (INNOVATIONS), 2014 10th International Conference on.* IEEE, 93-98

Allison, B., O'Sullivan, T., Owen, A., Rice, J., Rothwell, A. and Saunders, C. 2001. *Research Skills for Students*. London: Kogan Page.

Antunes, C. H., Dias, L., Dantes, G., Mathias, J. and Zamboni, L. 2016. An application of Soft Systems Methodology in the evaluation of policies and incentive actions to promote technological innovations in the electricity sector. *Energy Procedia*, 106 (1): 258-278.

Barab, S. A., Barnett, M., Yamagata-Lynch, L., Squire, K. and Keating, T. 2002. Using activity theory to understand the systemic tensions characterizing a technology-rich introductory astronomy course. *Mind, Culture, and Activity*, 9 (2): 76-107.

Basharina, O. K. 2007. An activity theory perspective on student-reported contradictions in international telecollaboration. *Language learning & technology*, 11 (2): 82-103.

Bergman, M. M. 2008. *Advances in Mixed Methods Research*. London: SAGE Publications Ltd.

Biggam, J. and Hogarth, A. 2001. Using Soft Systems Methodology to Facilitate the Development of a Computer Security Teaching Module. In: *Advances in Information Security Management & Small Systems Security.* Springer, 113-125.

Blin, F. and Munro, M. 2008. Why hasn't technology disrupted academics' teaching practices? Understanding resistance to change through the lens of activity theory. *Computers & Education*, 50 (2): 475-490.

Booch, G., Rumbaugh, J. and Jacobson, I. 1997. The Unified Modeling Language For Object-Oriented Development, Documentation Set Version 1.0.

Boström, G., Wäyrynen, J., Bodén, M., Beznosov, K. and Kruchten, P. 2006. Extending XP practices to support security requirements engineering. In: Proceedings of *Proceedings of the 2006 international workshop on Software engineering for secure systems.* ACM, 11-18

Brinkman, W. P. 2009. *Handbook of Mobile Technology Research Methods*. Nova Publisher.

Cao, L. and Ramesh, B. 2010. Agile Requirements Engineering Practices: An Empirical Study. *IEEE Computer Society*, 1 (1): 60-67.

Changing Minds. 2016. *Decision Criteria*. Available: http://changingminds.org/explanations/decision/decision_criteria.htm (Accessed 2 August 2017).

Chern, T. 2017. *Women in Software Development? YES PLEASE!* Available: https://pspdfkit.com/blog/2017/women-in-software-development/ (Accessed 24 July 2017).

Cigital. 2016. *Agile Security Manifesto*. Available: https://www.cigital.com/press-release/cigital-releases-agile-security-manifesto/ (Accessed 12 June 2017).

Computer World UK. 2016. *Top software failures 2015/2016*. Available: http://www.computerworlduk.com/galleries/infrastructure/top-10-software-failures-of-2014-3599618/ (Accessed 1 May 2016).

ComputerWeekly.com. 2017. *Securing NoSQL applications: Best practises for big data security*. Available: http://www.computerweekly.com/tip/Securing-NoSQL-applications-Best-practises-for-big-data-security (Accessed 14 July 2017).

Converse, J. M. and Presser, S. 1986. *Survey Questions*. Thousand Oaks: SAGE Publications, Inc.

Cramer, D. and Howit, D. 2004. *The SAGE Dictionary of Statistics*. London: SAGE Publications, Ltd.

Creswell, J. W. 2009. *Research Design:Qualitative & Quantitative Approaches*. London: Sage Publications.

Crow, I. 2006. *The SAGE Dictionary of Social Research Methods*. SAGE Publications, Ltd: London.

CWE. 2017. *Common Weakness Enumeration*. Available: https://cwe.mitre.org/data/definitions/928.html (Accessed 25 July 2017).

De Lucia, A. and Qusef, A. 2010. Requirements engineering in agile software development. *Journal of Emerging Technologies in Web Intelligence*, 2 (3): 212-220.

Dukes, L., Yuan, X. and Akowuah, F. 2013. A Case Study on Web Application Security Testing with Tools and Manual Testing. Paper presented at the *IEEE SOUTHEASTCON 2013*. Jacksonville

Dwibedy, D., Sahoo, L. and Dutta, S. 2013. A Generalized Definition Language for Implementing the Object Based Fuzzy Class Model. *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, 2 (4): pp: 1363-1367.

El-Hadary, H. and El-Kassas, S. 2014. Capturing security requirements for software systems. *Journal of advanced research*, 5 (4): 463-472.

Elahi, G. 2009. *Security requirements engineering: state of the art and practice and challenges.* Available: http://www.cs.utoronto.ca/~gelahi/ (Accessed 25 July 2017).

Elahi, G., Yu, E., Li, T. and Liu, L. 2011. Security requirements engineering in the wild: A survey of common practices. In: Proceedings of *Computer Software and Applications Conference (COMPSAC), 2011 IEEE 35th Annual*. IEEE, Munich, 314-319.

Engeström, Y. (1987) *Learning by Expanding: an activity-theoretical approach to developmental research* (Helsinki, Orienta-Konsultit).

Ernst, N. A. and Murphy, G. C. 2012. Case studies in just-in-time requirements analysis. In: Proceedings of *Empirical Requirements Engineering (EmpiRE), 2012 IEEE Second International Workshop on.* IEEE, 25-32.

Floyd, F. J. and Widaman, K. F. 1995. Factor analysis in the development and refinement of clinical assessment instruments. *Psychological assessment*, 7 (3): 286.

Fontana, R. M., Fontana, I. M., da Rosa Garbuio, P. A., Reinehr, S. and Malucelli, A. 2014. Processes versus people: How should agile sofware development maturity be defined ? *The Journal of Systems and Software*, 97 (1): 140-155.

Franzosi , R. P. 2004. *Handbook of Data Analysis*. London: SAGE Publications, Ltd.

Gartner. 2016. *Magic Quadrant for Application Security Testing* Available: https://www.linkedin.com/pulse/magic-quadrant-application-security-testing-neil-deepak-hota (Accessed 25 July 2017).

Ge, X., Paige, R. F., Polack, F. and Brooke, P. 2007. Extreme programming security practices. In: Proceedings of *International Conference on Extreme Programming and Agile Processes in Software Engineering.* Springer, 226-230

Gibbs, G. 2013. *Analyzing Qualitative Data*. London: Sage.

Graziano, A. M. and Raulin, M. 1997. *Research Methods: A Process of Inquiry*. New York: Longman.

Gubrium, J. F. and Holstein, J. A. 2002. *Handbook of Interview Research: Context and Method*. SAGE Publications.

Gurubaran, S. 2017. *OWASP A10-Unvalidated Redirects and Forwards*. Available: https://gbhackers.com/owasp-a10-unvalidated-redirects-forwards/ (Accessed 12 November).

Hagras, H. and Wagner, C. 2009. *Introduction to Interval Type-2 Fuzzy Logic Controllers -Towards Better Uncertainty Handling in Real World Applications* Available: http://ieeesmc.org/newsletters/back/2009_06/SMC-Hagras.html (Accessed 17 April 2016).

Haley, C. B., Moffett, J. D., Laney, R. and Nuseibeh, B. 2006. A framework for security requirements engineering. In: Proceedings of *Proceedings of the 2006 international workshop on Software engineering for secure systems.* ACM, 35-42

Harding, J. 2013. *Qualitative Data Analysis from start to finish*. London: Sage.

Harvard Business School. 2017. *A Refresher on Regression Analysis*. Available: https://hbr.org/2015/11/a-refresher-on-regression-analysis (Accessed 5 July 2017).

Hasan, M. S., Mahmood, A., Alam, M. J., Hasan, S. and Rahman, F. 2010. An evaluation of software requirement prioritization techniques. *International Journal of Computer Science and Information Security (IJCSIS)*, 8 (9).

Hatton, S. 2008. Choosing the right prioritisation method. In: Proceedings of *Software Engineering, 2008. ASWEC 2008. 19th Australian Conference on.* IEEE, 517-526.

Heath, N. 2016. *No place for the old? Is software development a young person's game?* Available: http://www.techrepublic.com/article/no-place-for-the-old-is-software-development-a-young-persons-game/ (Accessed 1 October 2017).

Henver, A. R., March, S. T. and Park, J. 2004. Design Research in Information Systems Research. *MIS Quarterly,*, 28 (1): 75-105.

Houška, M. and Dömeová, L. 2003. Cost and Benefit Criteria in Methods Based on Distances from Ideal and Negative Ideal Variants. *Proceedings of Mathematical and Computer Modelling in Science and Engineering*: 150-154.

Hussein, Z. 2017. Leading to Intention: The Role of Attitude in Relation to Technology Acceptance Model in E-Learning. *Procedia Computer Science*, 105: 159-164.

Inayat, I. and Salim, S. S. 2015. A framework to study requirements-driven collaboration among agile teams: Findings from two case studies. *Computers in Human Behavior*, 51 (1): 1367-1379.

Inayat, I., Salim, S. S., Marczak, S., Daneva, M. and Shamshirband, S. 2015. A systematic literature review on Agile requirements engineering practices and challenges. *Computers in Human Behavior*, 1 (1): 915-929.

Investopedia. 2017. *Delphi Method*. Available: http://www.investopedia.com/terms/d/delphi-method.asp (Accessed 28 June 2017).

Kagdi, H., Maletic, J. I. and Sutton, A. 2005. Context-free slicing of UML class models. In: Proceedings of *Software Maintenance, 2005. ICSM'05. Proceedings of the 21st IEEE International Conference on.* IEEE, 635-638.

Kaptelinin, V. 1995. *Activity theory: implications for human-computer interaction.*

Karlsson, J., Wohlin, C. and Regnell, B. 1998. An evaluation of methods for prioritizing software requirements. *Information and Software Technology,* 39 (14-15): 939-947.

Kassab, M. 2014. An Empirical Study on the Requirements Engineering Practices for Agile Software Developement Paper presented at the *40th Euromicro Conference on Software Engineering and Advanced Applications.* Verona, 254-261.

Kavitha, C. R. and Thomas, S. M. 2011. Requirement Gathering for small projects using Agile Methods. *Computational Science-New Dimensions & Perspectives*, 1 (1): 122-128.

Kellett, M. 2005. *How to Develop Children as Researchers: A Step-by-Step Guide to Teaching the Research Process*. London: SAGE Publications Ltd.

Kumar, M., Shukla, M. and Agarwal, S. 2013. A Hybrid Approach of Requirement Engineering in Agile Software Development. In: Proceedings of *International Conference on Machine Intelligence and Research Advancement.* Katra, India: IEEE Computer Society, 515-519.

Kuuti, K. 1995. Activity Theory as a potential framework for human computer interaction research. *MIT Press*, 1 (1): 17-44.

Kvale, S. 2007. *Doing Interviews*. London: Sage.

Lavrakas, P. 2008. Encyclopedia of Survey Research Methods.

LeCompte, M. D. 2000. Analyzing qualitative data. *Theory into practice,* 39 (3): 146-154.

Legris, P., Ingham, J. and Collerette, P. 2003. Why do people use information technology? A critical review of the technology acceptance model. *Information & Management,* 40 (3): 191-204.

Licorish, S. A. and MacDonell, S. G. 2014. Understanding the attitudes, knowledge sharing behaviors and task performance of core developers: A longitudinal study. *Information and Software Technology,* 56 (12): 1578-1596.

Lim, C. P. and Hang, D. 2003. An activity theory approach to research of ICT integration in Singapore schools. *Computers & Education*, 41 (1): 49-63.

Lima F R Jr, Osiro L and R, C. L. C. 2014. A comparison between Fuzzy AHP and Fuzzy TOPSIS methods to supplier selection. *Applied Soft Computing,* 1 (1): 194-209.

Lin, C. T., Chiu, H. and Y.H., T. 2006. Agility evaluation using fuzzy logic. *International Journal of Production Economics*, 101 (1): 353-368.

Lonescu, P. 2015. *The 10 Most Common Application Attacks in Action*. Available: https://securityintelligence.com/the-10-most-common-application-attacks-in-action/ (Accessed 18 April 2016).

Ma, Z. M., Yan, L. and Zhang, F. 2012. Modeling fuzzy information in UML class diagrams and object-oriented database models. *Fuzzy Sets and Fuzzy Systems*, 186 (1): 26-16.

Madan, R. C. 2014. *An Introduction to MATLAB for Behavioral Researchers*. Thousand Oaks: SAGE Publications, Inc.

Maheshwari, S. and Sharma, C. 2014. Ten Security Practices to a formidable ERP System. In: Proceedings of *International Conference on Smart Structures & Systems.* Chennai, 41-50

Marashdih, A. W. and Zaaba, Z. F. 2016. Cross Site Scripting: Detection Approaches in Web Application. *International Journal of Advanced Computer Science and Applications*, 7 (10): 155-160.

Martin, H. A., Zarchi, M. K., Azizollahi, S. 2011. The Application of Fuzzy Topsis Approach to Personnel Selection for PAdir Company, Iran. *Journal of Management Research* 3(2): 1-13.

Matin, H. Z., Fathi, R. M., Zarchi, M. K. and Azizollahi, S. 2011. The Application of Fuzzy TOPSIS approach to Personnel Selection for PAdir Company, Iran. *Journal of Management Research*, 3 (2): 1-14.

Mouton, J. 2004. How to succeed in your Master's and Doctoral Studies. In: Pretoria: Van Schaik publishers.

Murphy, E. and Rodriguez-Manzanares, M. A. 2008. Using activity theory and its principle of contradictions to guide research in educational technology. *Australasian Journal of Educational Technology*, 24 (4).

Nardi, B. A. 1995. Activity Theory and Human-Computer Interaction. *MIT Press*, 1 (1): 4-7.

Nasir, M. H. and Sahibuddin, S. 2011. Critical success factors for software projects: A comparative study *Scientific Research and Essays*, 6 (10): 2174-2186.

Nortje, Y. 2013. *The great debate – permanent employee or contractor?* Available: http://www.itweb.co.za/index.php?option=com_content&view=article&id=64403 (Accessed 1 October 2017).

O'Leary, Z. 2007. *The Social Science Jargon Buster*. London: SAGE Publications Ltd.

Oppenheim, A. N. 2003. *Questionnaire Design, Interviewing and Attitude Measurement*. London: Continuum.

Oracle Database Security Guide. 2003. *Introducing Database Security for Application Developers*. Available: http://www.computerweekly.com/tip/Securing-NoSQL-applications-Best-practises-for-big-data-security (Accessed 28 August 2017).

Ornstein, M. 2013. *A companion to Survey Research*. London: Sage.

Payne, G. and Payne, J. 2004. *Key concepts in social research*. London: SAGE Publications, Ltd.

Pearson, A. 2016. *What's The Difference Between Static and Dynamic Software Testing?* Available: http://www.securityinnovationeurope.com/blog/whats-the-difference-between-static-and-dynamic-software-testing (Accessed 16 April 2016).

Peffers, K., Tuunanen, T., Gengler, C. E., Rossi, M., Hui, W., Virtanen, V. and Bragge, J. 2006. The design science research process: a model for producing and presenting information systems research. In: Proceedings of *Proceedings of the first international conference on design science research in information systems and technology (DESRIST 2006).* sn, 83-106

Peffers, K., Tuunanen, T., Rothenberger, M. A. and Chatterjee, S. 2007. A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems*, 24 (3).

Peng, C. 2009. Data Analysis Using SAS&#174. Thousand Oaks, California.

Pett, M., Lackey, N. and Sullivan, J. 2003. Making Sense of Factor Analysis. In: Thousand Oaks, California: SAGE Publications, Inc. Available: http://methods.sagepub.com/book/making-sense-of-factor-analysis (Accessed 17 July 2017)

Positive Technologies. 2015. *Web application vulnerability statistics*

Pressman, R. S. and Maxim, B. R. 2015. *Software Engineering: A Practitioner's Approach*. Singapore: McGraw Hill Education.

Racheva, Z., Daneva, M., Sikkel, K. and Wieringa, R. 2010. Do we know enough about requirements prioritization in Agile Projects: Insights from a Case Study. In: Proceedings of *18th IEEE International Requirements Engineering Conference.* Sydney, IEEE Computer Society,

Ramesh, B., Cao, L. and Baskerville, R. 2010. Agile requirements engineering practices and challenges: an empirical study. *Information Systems Journal*, 20 (5): 449-480.

Rapley, T. 2004. *Qualitative Research Practice*. London: SAGE Publications Ltd.

Regoniel, P. 2010. *What is the Difference Between the Theoretical and the Conceptual Framework?* Available: https://college-college-life.knoji.com/what-is-the-difference-between-the-theoretical-framework-and-the-conceptual-framework/ (Accessed 17 July 2017).

Rubin, H. J. and Rubin, I. S. 2012. *Qualitative Interviewing*. London: Sage.

Safari, H., Faghih, A. and Fathi, M. R. 2012. Fuzzy multi-criteria decision making method for facility location selection. *African Journal of Business Management*, 6 (1): 206.

Salini, P. and Kanmani, S. 2011. A Model based Security Requirements Engineering Framework applied for Online Trading System. In: Proceedings of *IEEE-International Conference on Recent Trends in Information Technology,*. Chennai, IEEE,1195-1202.

Salini, P. and Kanmani, S. 2012. Application of model oriented security requirements engineering framework for secure E-voting. In: Proceedings of *Software Engineering (CONSEG), 2012 CSI Sixth International Conference on.* IEEE, 1-6.

SANS Institute. 2015. *2015 State of Application Security: Closing the Gap.*

Schön, E.-M., Escalona, M. J. and Thomaschewski, J. 2015. Agile Values and Their Implementation in Practice. *IJIMAI*, 3 (5): 61-66.

Schön, E.-M., Thomaschewski, J. and Escalona, M. J. 2017. Agile requirements engineering: a systematic literature review. *Computer Standards & Interfaces*, 49: 79-91.

Serrador, P. and Pinto, J. K. 2015. Does Agile work? - A quantitative analysis of agile project success. *International Journal of Project Management*, 33 (1): 1040-1051.

Sheffield, J. and Lemetayer, J. 2013. Factors associated with the software development agility of successful projects. *International Journal of Project Management*, 31 (1): 459-472.

Sillitti, A. and Succi, G. 2005. 14 Requirements Engineering for Agile Methods. *Engineering and Managing Software Requirements*: 309-326.

Silverman, D. 1998. *Qualitative Research: Theory Method and Practice*. London: Sage Publications Ltd.

Sodhi, B., Prabhakar T.V. . 2012. *A Simplified Description of Fuzzy Topsis.* Kanpur: Dept. of Computer Science and Engineering.

Sommerville, I. 2016. *Software Engineering* tenth ed. London: Pearson.

Souag, A. 2012. Towards a new generation of security requirements definition methodology using ontologies. In: Proceedings of *24th International Conference on Advanced Information Systems Engineering (CAiSE'12) Gdańsk, Poland, 25-29 June 2012.* 1-8

Souag, A., Mazo, R., Salinesi, C. and Comyn-Wattiau, I. 2015. Reusable knowledge in security requirements engineering: a systematic mapping study. *Requirements Engineering*: 1-33.

Souag, A., Salinesi, C. and Comyn-Wattiau, I. 2012. Ontologies for security requirements: A literature survey and classification. In: Proceedings of *Advanced Information Systems Engineering Workshops.* Springer, 61-69

Stankovic, D., Nikolic, V., Djordjevic, M., Cao, D.B. 2013. A survey study of critical success factors in agile software projects in former Yugoslavia IT companies. *The Journal of Systems and Software*, 86 (1): 1663-1678.

Sutcliffe, A. and Sawyer, P. 2013. Requirements elicitation: Towards the unknown unknowns. In: Proceedings of *Requirements Engineering Conference (RE), 2013 21st IEEE International.* IEEE, 92-104

Team Technology. 2017. *Strategies to improve team dynamics*. Available: http://www.teamtechnology.co.uk/team/dynamics/strategies/ (Accessed 29 July 2017).

Teddlie, C. and Tashakkori, A. 2010. SAGE Handbook of Mixed Methods in Social & Behavioral Research.

Tondel, I. A., Jaatun, M. G. and Meland, P. H. 2008. Security requirements for the rest of us: A survey. *IEEE software*, 25 (1).

Tuunanen, T. and Kuo, I.-T. 2015. The effect of culture on requirements: a value-based view of prioritization. *European Journal of Information Systems*, 24 (3): 295-313.

UCLA Institute for Digital Research and Education. 2017. *SPSS Annotated Output Regression Analysis*. Available: https://stats.idre.ucla.edu/spss/output/regression-analysis/ (Accessed 5 July 2017).

Uden, L. 2006. Activity theory for designing mobile learning. *International Journal of Mobile Learning and Organisation*, 1 (1): 81-102.

Venable, J., Pries-Heje, J. and Baskerville, R. 2012. A comprehensive framework for evaluation in design science research. In: Proceedings of *International Conference on Design Science Research in Information Systems.* Springer, 423-438

Vinodh, S., Devadasan, S. R., Reddy, B. V. and Ravichand, K. 2010. Agility index measurement using multi-grade fuzzy approach integrated in a 20 criteria agile model. *International Journal of Production Research*, 48 (23): 7159–7176.

Vogt, W. P. 2008. *Dictionary of Statistics & Methodology, Third Edition* Thousand Oaks: SAGE Publications, Inc.

Volper, R. 2012. *7 Compensation Tactics To Help Retain Employees*. Available: https://www.cnbc.com/id/46045960 (Accessed 29 July 2017).

Walker, L. O. and Avant, K. C. 2011. *Strategies for Theory Construction in Nursing*. 5th ed. Pearson.

Wall, J. D., Lowry, P. B. and Barlow, J. 2015. Organisational violations of externally governed privacy and security rules: Explaining violations of externally governed privacy and security rules: Explaining and predicting selective violations under conditions of strain and excess. *Journal of Association for Information Systems*, 17 (1): 39-76.

Walliman, N. 2004. *Your Research Project*. London: Sage Publications Ltd.

Wellman, C., Kruger, F. and Mitchell, B. 2005. *Research Methodology*. Cape Town: Oxford University Press.

Williams, P., Ashill, N. J., Nauman, E. and Jackson, E. 2015. Relationship quality and satisfaction: Customer-perceived success factors for on-time projects. *International Journal of Project Management*, 33 (1): 1836-1850.

Woolford, S. 2015. *(Factor) Analyze This:PCA or EFA*. Available: https://www.genome.gov/pages/careers/.../samwoolford_factoranalysispcaorefa.pdf (Accessed 17 July 2017).

World Economic Forum. 2016. *The Global Information Technology Report 2016.*

Wurfel, D., Lutz, R. and Diehl, S. 2016. Grounded requirements engineering: An approach to use case driven requirements engineering. *The Journal of Systems and Software*, 1 (1): 1-13.

Yang, S. L. and Li, T. F. 2002. Agility evaluation of mass customization product manufacturing *Journal of Materials Processing Technology*, 129 (1): 640-644.

Yin, R. K. 1999. Enhancing the quality of case studies in health services research. *Health Services Research*, 34 (5 Pt 2): 1209.

Zahari, N. I. N. and Abdullah, M. L. 2012. Evaluation of Sustainable Development Indicators With Fuzzy TOPSIS Based on Subjective and Objective Weights. *IIUM Engineering Journal*, 13 (1).

Zadeh, L. A. (1965), Fuzzy sets, *Information and Control* 8, 338-353.

ANNEXURE A
PROPOSED CONCEPTUAL FRAMEWORK

Delphi Questionnaire

Kindly provide your expert opinion on the emergent Soft Activity Model (SAM) and its intended use in this study by answering the questions below:

1. Is the proposed Soft Activity Model (SAM) in your assessment reflective of the study concepts and does it provide realistic opportunities for:
   - Expressing the problem situation
   - Providing constructs for creating a solution to the problem
   - Providing the researcher with conceptual tools to suggest activities to achieve the solutions mentioned above.

2. What additional concepts can be suggested to align better with interpreting Agile RE practices and security approaches in requirements engineering?

3. Comment on concepts and constructs that have been omitted from Soft Systems Methodology and Activity Theory. Do you think that the researcher was justified in leaving those items out in terms of this research study?

ANNEXURE B
RESEARCH PROJECT PLAN: SCHEDULE OF ACTIVITIES

	Jan 2015	Mar 2016	Apr 2016	Aug 2016	Sept 2016	Oct 2016	Nov 2016	Dec 2016	Jan 2017	Feb 2017	Apr 2017	May 2017	Jul 2017	Oct 2017	Nov 2017	Dec 2017
Propose and Select Topic																
Select and meet supervisor																
Preliminary literature review																
Write and submit research proposal																
In-depth literature study																
Identify data sources and gain access																
Develop instrument for survey																
Conduct survey																
Develop and conduct interviews																
Data capturing and editing																
Data analysis and synthesis																
Writing thesis																
Proof reading by 3 academics																
Advise Registrar of intention to submit thesis for examination																
Final editing and Approval by Sup.																
Submit Thesis for Examination																

ANNEXURE C
RESEARCH PLAN- SUMMARY

TITLE				
SECURE REQUIREMENTS ENGINEERING IN A CONSTRAINED AGILE SOFTWARE DEVELOPMENT ENVIRONMENT				
AIM				
To determine how Agile RE practices affect the security of Agile Software Development products.				
Research questions	**Objectives**	**Data Collection Methods**	**Data Sources**	**Data Analysis**
	1. Assess RE processes, security RE approached and Agile RE practices from the existing body of knowledge.	Literature Review		
1. What are the Agile RE practices in the software development industry?	2. Evaluate the extent to which secure RE processes are implemented in Agile RE practices in Industry.	Structured Interview Document Review Survey Questionnaire (breadth of study)	From Individual Projects From Individual Projects From the Agile Software Development Stakeholders	Content Analysis Statistical Analysis
2. How do software engineers control requirements in Agile RE?	3. Apply Fuzzy TOPSIS as an alternate method to rank client requirements.	Desktop Review	Literature	
	4. Evaluate the automated fuzzy tool to support secure Agile RE practices.	Structured Interviews (depth of study)	Software Development Team	Content Analysis
3. To what extent are secure RE processes implemented in Agile RE practices in Industry?	5. Evaluate application security using a dynamic analysis tools (DAST).	Acunetix Web Vulnerability Scanner	Individual Projects	Statistical Analysis
	6. Develop evidence based guidelines for implementing security in Agile RE that will be convenient for the regular software developer to adopt.	All above		Content Analysis Statistical Analysis

ANNEXURE D

SURVEY QUESTIONNAIRE

UNIVERSITY OF KWAZULU-NATAL
INYUVESI YAKWAZULU-NATALI

Project No. [ ]

Dear Respondent

Thank you for volunteering to participate in this research. It will take approximately 10 mins to answer the questions.

The purpose of this questionnaire is to:

- **Evaluate the extent that secure requirements engineering (RE) approaches are implemented in Agile RE practices in Industry;**
- **Establish how software engineers manage client security requirements.**

It is important that you answer questions as honestly as possible. Your responses are voluntary and will be treated confidentially. If you have any questions kindly contact N.K. Naicker via. e-mail nalindrenn@dut.ac.za

Thanking You

N.K. Naicker

SECTION A: DEMOGRAPHIC INFORMATION

1. What is your current role in the Agile Software Development team? *Tick ☑ one of 5 options.*

Project Manager		**1**
Team Leader		**2**
Team member		**3**
Business Analyst		**4**
Product owner		**5**

2. What type of qualification do you have? *Tick ☑ <u>one</u> of 5 options.*

Certificate		**1**
Diploma		**2**
Degree		**3**
Postgraduate Studies		**4**
Other		**5**

3. Employment type *Tick ☑ <u>one</u> option*: Permanent [ ] 1 Contract [ ] 2

4. Indicate your age group? (In completed years) *Tick ☑<u>one</u> of 5 options.*

18-20		**1**
21-25		**2**
26-30		**3**
31-45		**4**
46^{+}		**5**

5. Gender *Tick ☑<u>one</u> option*: Male ☐ 1 Female ☐ 2

6. Number of years' experience in Software Engineering? (In completed years)

Tick ☑ one of 4 options.

0-2		**1**
3-5		**2**
6-10		**3**
More than 10		**4**

7. What type of application security training did you receive in the last 12 months?

Tick ☑ the appropriate option/s.

Security requirements/modeling related		**1**
Code related		**2**
General application threats and vulnerabilities		**3**
Security metrics tools		**4**
No, I did not receive security training in the last 12 months		**5**

8. The value of application security training received by developers *Tick ☑ one option only.*

Is adequate for gathering security requirements		**1**
Is inadequate for gathering security requirements		**2**
Not sure		**3**

SECTION B: REQUIREMENTS ENGINEERING

Answer the following questions as honestly as possible in terms of the requirements engineering process in Agile Software Development at your company.

Tick ☑<u>one</u> option in each row.

9. We used the following elicitation techniques for the functional requirements:

	1	2	3	4	5
Statement	**Strongly Disagree**	**Disagree**	**Not Sure**	**Agree**	**Strongly Agree**
1. Objectives of the web application are identified					
2. All stakeholders are identified					
3. All viewpoints are established					
4. Assets of the system are identified					
5. Security experts are identified					
6. Non-security goals identified					
7. Normal requirements are identified					
8. Non-functional requirements are identified					

Tick ☑ the appropriate option/s.

Brainstorming		**1**
Focus Groups		**2**
Interviews		**3**
Joint Application Development (JAD)		**4**
Prototyping		**5**
Usage Scenarios		**6**
Quality Function Deployment		**7**
Surveys/questionnaire		**8**
Viewpoints		**9**
User stories		**10**
Ethnographic study		**11**
Innovative workshops		**12**
Other		**13**
Not Sure		**14**

10. We used the following elicitation techniques for the non-functional requirements:

Tick ☑ the appropriate option/s.

Brainstorming		**1**
Focus Groups		**2**
Interviews		**3**
Joint Application Development (JAD)		**4**
Prototyping		**5**
Usage Scenarios		**6**
Quality Function Deployment		**7**
Surveys/questionnaire		**8**
Viewpoints		**9**
User stories		**10**
Ethnographic study		**11**

Innovative workshops		**12**
Other		**13**
Not Sure		**14**

On a scale from [0-5], rate the degree of difficulty to elicit requirements for each case below?

Circle a number [0-5] in each row. *Not difficult*🡻 *Most difficult*🡻

11. When project goals are unclear	0	1	2	3	4	5
12. When stakeholders priorities differ	0	1	2	3	4	5
13. When people have unspoken assumptions	0	1	2	3	4	5
14. When stakeholders interpret meanings differently	0	1	2	3	4	5
15. When requirements are stated in a way that makes it difficult to verify	0	1	2	3	4	5
16. When the customer is unavailable	0	1	2	3	4	5

Requirements are elaborated as follows:

Tick ☑ one option in each row.

	1	**2**	**3**	**4**	**5**
Statement	**Strongly Disagree**	**Disagree**	**Not Sure**	**Agree**	**Strongly Agree**
17. Generating Use Cases					
18. UML activity diagrams					
19. Class diagrams					
20. State diagrams					

Requirements are Analysed as follows:

Tick ☑ one option from each row.

	1	**2**	**3**	**4**	**5**
Statement	**Strongly Disagree**	**Disagree**	**Not Sure**	**Agree**	**Strongly Agree**
21. Structured Requirements Definition					
22. Object Oriented Analysis					
23. Structured Analysis and Design					
24. No methodology					

On a scale from [0-5], rate the extent to which each case below is a factor for requirements negotiation with the client:

Circle a number [0-5] in each row.

Not a factor ↓ ... *Very strong factor* ↓

Statement						
25. Time-to-market	0	1	2	3	4	5
26. Tradeoff between functional and non-functional requirements.	0	1	2	3	4	5
27. Cost	0	1	2	3	4	5
28. Conflicting requirements	0	1	2	3	4	5
29. Security requirements	0	1	2	3	4	5
30. Prioritisation of requirements	0	1	2	3	4	5

31. Security Requirements in projects are generated as follows:

Circle <u>one</u> option only.

	Option
• Identifying security goals • Identifying threats/vulnerabilities • Perform formal Risk Assessment • Prioritizing threats/vulnerabilities • Generate misuse case • Identify security requirements	A
• System modeling • Asset identification • Threats and vulnerabilities identification • Security requirements elicitation • Security requirements evaluation	B
Another Method	C
No specific Method	D

32. Please indicate how security requirements are specified.

Tick ☑ <u>one</u> option only.

Security Specifications Language (eg. CLASP, Secure TROPOS, etc.)		1
Semi-Formal Notations (UML, class, sequence diagram)		2
Informal language (User stories/scenarios)		3

33. In the requirements engineering phase of projects security requirements are identified by:

Development team		**1**
Dedicated security team		**2**
Not sure		**3**

34. Is the dedicated security expert/s brought into the team to validate of requirements?

Yes		**1**
No-developers validate all requirements		**2**

On a scale from [0-5] rate the degree to which each factor is a constraint to secure requirements engineering in Agile?
Circle a number [0-5] in each row.

	Not a factor ↓				*Very strong factor* ↓	
35. Large scope	0	1	2	3	4	**5**
36. Limited budget for project	0	1	2	3	4	**5**
37. Limited time to complete	0	1	2	3	4	**5**
38. Change in requirements	0	1	2	3	4	**5**
39. Non-security risks in the project	0	1	2	3	4	**5**
40. Limited human resources	0	1	2	3	4	**5**
41. Limited security knowledge of team	0	1	2	3	4	**5**
42. Poor management support for security	0	1	2	3	4	**5**
43. Lack of interest in security by the customer	0	1	2	3	4	**5**

SECTION C: RATE YOUR RE PRACTICES IN PROJECTS

On a scale from [1-5] rate the following RE practices in Agile Software Development projects at your company: *1: very weak* *2: weak* *3: satisfactory* *4: Good* *5: Excellent*

Circle a number [1-5] in each row.

Processes					
1. Requirements gathering	**1**	**2**	**3**	**4**	**5**
2. Identification of security goals	**1**	**2**	**3**	**4**	**5**
3. Requirements analysis and modelling	**1**	**2**	**3**	**4**	**5**
4. Requirements estimation efforts	**1**	**2**	**3**	**4**	**5**
5. System for requirements traceability to work products	**1**	**2**	**3**	**4**	**5**
6. The tradeoff between functional and non-functional requirements	**1**	**2**	**3**	**4**	**5**
7. The valuation of assets and resources of the software being developed	**1**	**2**	**3**	**4**	**5**
8. Requirements inspection to identify potential threats	**1**	**2**	**3**	**4**	**5**
9. Security Risk Analysis	**1**	**2**	**3**	**4**	**5**
10. Security requirements identification	**1**	**2**	**3**	**4**	**5**
11. Requirements validation methods	**1**	**2**	**3**	**4**	**5**
12. System for requirements management (changes, tracking and control of requirements)	**1**	**2**	**3**	**4**	**5**

Thank you for your time and co-operation in completing this survey.

ANNEXURE E

INTERVIEW QUESTIONNAIRE FOR SECURE REQUIREMENTS ENGINEERING IN AGILE

Title: Secure Requirements Engineering in a Constrained Agile Environment

Preamble

Thank you for volunteering to participate in this research. Software vulnerabilities are a common problem and result in huge losses for the customer. It has become critical for security concerns to be addressed early in the software development lifecycle and carried through to other phases to ensure that secure systems are built. The researcher requires your assistance to delineate the Agile RE practices to explicate the relationship between RE practices and the security of an application.

The purpose of this questionnaire is to therefore:

- **Evaluate the extent to which secure RE approaches are implemented in Agile RE practices in Industry;**
- **Establish how software engineers manage client security requirements.**

INTERVIEW QUESTIONS: Guided questions for stakeholders of the Agile Project

1. *Researcher:* What is your role in the Agile Software Development project?
   Respondent: ……

2. *Researcher:* Provide a detailed explanation of how the requirements were elicited for this project?
   Respondent: ……

3. *Researcher:* How did you ensure that all viewpoints were catered for during requirements engineering of this project?
   Respondent: ……

4. *Researcher:* At what stage in the RE process do you elicit security requirements.
   Respondent: ……

5. *Researcher:* What are the other non-functional requirements identified and when were they identified within requirements engineering processes?

Respondent: ……

6. *Researcher:* What security knowledge sources were referenced for this project?
   Respondent: ……

7. *Researcher:* There are several known Security Requirements Engineering (SRE) approaches in security literature such as *Misuse Cases, CLASP, Secure TROPOS, Anti-Models, Abuser stories, etc*. Have you implemented any structured SRE approach in this project.
   Respondent: ……

8. *Researcher:* Who was responsible for identifying security requirements in this project?
   Respondent: ……

9. *Researcher:* How knowledgeable are your Software Engineers experts with regard to secure software development? What level of training do Software Engineers receive when working on this project?
   Respondent: ……

10. *Researcher:* If the Software Engineers are non-security experts, explain some challenges experienced by them when factoring security into the system.
    Respondent: ……

11. *Researcher:* Discuss your SRE approach.
    Respondent: ……

12. *Researcher:* Comment on customer involvement in SRE and at what stages in RE did they get involved.
    Respondent: ……

13. *Researcher:* What was the main focus of your security requirements engineering approach for this project and can you remember some of the security requirements identified?
    Respondent: ……

14. *Researcher:* List the detailed steps involved to elicit and analyse the security requirements for this project. In your answer discuss identification of assets and risk assessment.
    Respondent: ……

15. *Researcher:* How were threats/vulnerabilities identified, rated and prioritised in this project.
    Respondent: ……

16. *Researcher:* What role did the client's security policy play in your SRE approach.
    Respondent: ……

17. *Researcher:* What would you say are the benefits of your SRE approach?
    Respondent: ……

18. *Researcher:* Who ranked the priority of requirements in this project?
    Respondent: ……

19. *Researcher:* Describe the process involved in ranking requirements and the role of security requirements in this process of the project?
    Respondent: ……

20. *Researcher:* Are you satisfied with Agile RE practices employed for this project in general? Motivate.

    Respondent: ……

21. *Researcher:* Do you have any suggestions for improvement of security requirements in Agile RE practices on this project?

    Respondent: ……

THE END

ANNEXURE F
INTERVIEW QUESTIONNAIRE:TOOL EVALUATION

Title: Secure Requirements Engineering in a Constrained Agile Environment

Preamble

Thank you for volunteering to participate in this research. A new software tool has been developed to address the prioritisation of clients requirements using a computerized Fuzzy based system as a more effective and efficient way to rank normal requirements and non-functional requirements.

The purpose of this questionnaire is to therefore:

- **Elicit the views of requirements engineers on the extent to which the new software tool is able to meet their requirements.**

INTERVIEW QUESTIONS: Guided questions for stakeholders of the Agile Project

SECTION A: GENERAL

1. *Researcher:* What is your role in the Agile Software Development project?
   Respondent: ……
2. *Researcher:* What techniques are presently used to rank client requirements?
   Respondent: ……
3. *Researcher:* What are the limitations of current techniques?
   Respondent: ……
4. *Researcher:* Who are the stakeholders involved in the ranking of client requirements presently?
   Respondent: ……

SECTION B: EASE OF USE

1. *Researcher:* Describe your experience with the ease of use of the software tool?
   Respondent: ……
2. *Researcher:* What changes would you make to improve the ease of use?
   Respondent: ……

3. *Researcher:* Comment on your satisfaction with the user interface. Does it promote easy Use?

   Respondent: ……

SECTION C: FUNCTIONALITY AND CAPABILITY OF THE SOFTWARE TOOL

1. *Researcher:* Discuss your satisfaction with the time taken by the tool to effectively rank requirements?

   Respondent: ……

2. *Researcher:* Comment on the correctness of the automated fuzzy tool?

   Respondent: ……

3. *Researcher:* Comment on the scale of measurement used?

   Respondent: ……

4. *Researcher:* Comment on the scalability of the tool. How is the performance of the tool as the number of requirements increases?

   Respondent: ……

5. *Researcher:* Is the tool suitable for high level requirements as well as detailed requirements?

   Respondent: ……

SECTION D: OVERALL

Researcher: Final comments on the automated fuzzy software tool in comparison to current Methods?

Respondent: ……

ANNEXURE G
INFORMED CONSENT FORM TO PARTICIPANT

UNIVERSITY OF KWAZULU-NATAL

SCHOOL OF MANAGEMENT, IT and GOVERNANCE

Dear Respondent,

PhD Research Project
Researcher: N. K. Naicker (Tel: 031 3735588)
Supervisor: Professor M. S. Maharaj (Office Telephone Number: 031 2608003)

I, **N.K. Naicker**, am a PhD student, at the School of IT, Management and Governance, of the University of KwaZulu-Natal. You are invited to participate in a research project titled "***SECURE REQUIREMENTS ENGINEERING IN A CONSTRAINED AGILE ENVIRONMENT".***

The aim of this study is to investigate:

- ***How Agile Requirements Engineering practices affect the security of Agile Software Development applications.***

Through your participation I hope to:

- Understand the nature of Requirements Engineering (RE) in Agile Software Development;
- Evaluate the extent that secure RE processes are implemented in Agile RE practices;
- Elicit evidence based guidelines of implementing RE in Agile that will ensure secure applications development;
- Elicit your opinions regarding the development of a new fuzzy-based software tool; and
- Ascertain what contributions you can provide in the development of the new tool.

The results of the survey will assist in determining how Agile Software Development requirements engineering practices affect the security of Agile Software Development applications as well as how effective and efficient the new fuzzy-based software tool is to requirements engineers.

Your participation in this project is voluntary. You may refuse to participate or withdraw from the project at any time with no negative consequence. There will be no monetary gain from participating in this survey. Confidentiality and anonymity of records identifying you as a participant will be maintained by the School of Information Systems and Technology, UKZN.

If you have any questions or concerns about completing the questionnaire or about participating in this study, you may contact me or my supervisor at the numbers listed above.

The survey should take you about **25** minutes to complete. I hope you will take the time to complete this survey.

Researcher : N. K. Naicker Date : 11 August 2016

ANNEXURE H
ETHICAL CLEARANCE

UNIVERSITY OF KWAZULU-NATAL
INYUVESI YAKWAZULU-NATALI

08 December 2016

Mr Nalindren Kistasamy Naicker (214585809)
School of Management, IT & Governance
Westville Campus

Dear Mr Naicker,

Protocol reference number: HSS/1932/016D
Project title: Secure Requirements Engineering in a Constrained Agile Environment

Full Approval – Expedited Application

In response to your application received on 01 November 2016, the Humanities & Social Sciences Research Ethics Committee has considered the abovementioned application and the protocol have been granted **FULL APPROVAL.**

Any alteration/s to the approved research protocol i.e. Questionnaire/Interview Schedule, Informed Consent Form, Title of the Project, Location of the Study, Research Approach and Methods must be reviewed and approved through the amendment/modification prior to its implementation. In case you have further queries, please quote the above reference number.

PLEASE NOTE: Research data should be securely stored in the discipline/department for a period of 5 years.

The ethical clearance certificate is only valid for a period of 3 years from the date of issue. Thereafter Recertification must be applied for on an annual basis.

I take this opportunity of wishing you everything of the best with your study.

Yours faithfully

[redacted]

Dr Shenuka Singh (Chair)

/ms

Cc Professor MS Maharaj
Cc Academic Leader Research: Professor Brian McArthur
Cc School Administrator: Ms Angela Pearce

Humanities & Social Sciences Research Ethics Committee
Dr Shenuka Singh (Chair)
Westville Campus, Govan Mbeki Building
Postal Address: Private Bag X54001, Durban 4000
Telephone: +27 (0) 31 260 3587/8350/4557 **Facsimile:** +27 (0) 31 260 4609 **Email:** ximbap@ukzn.ac.za / snymanm@ukzn.ac.za / mohunp@ukzn.ac.za
Website: www.ukzn.ac.za

ANNEXURE I
AUTMATED FUZZY TOOL: UTILITY CLASS WITH HELPER/UTILITY METHODS

```
namespace FuzzyTopsis.Utilities
{
   /// <summary>
   /// Contains helper/utility functions
   /// </summary>
      public class Utils
   {
      Step1 _step1 = new Step1();

      /// <summary>
      /// Convert an array of qualitative values into a fuzzy.FuzzyNumber </summary>
      /// <param name="decisions"> Array of decisions which are linguistic terms </param>
      /// <returns> fuzzy.FuzzyNumber </returns>

      public FuzzyNumber QualitativeToFuzzy(string[] decisions)
      {
         double[] minArray = new double[decisions.Length];
         double[] geoMeanArray = new double[decisions.Length];
         double[] maxArray = new double[decisions.Length];

         for (int i = 0; i < decisions.Length; i++)
         {
            // Converting the linguistic term to a Fuzzy number, then extract each component
            minArray[i] = _step1.LinguisticToFuzzy(decisions[i]).Min;
            geoMeanArray[i] = _step1.LinguisticToFuzzy(decisions[i]).Mean;
            maxArray[i] = _step1.LinguisticToFuzzy(decisions[i]).Max;
         }

         return new FuzzyNumber(GetMinValue(minArray), GetAverage(geoMeanArray),
        GetMaxValue(maxArray));
```

```
}

/// <summary>
/// Calculates the average value from the given array of values </summary>
/// <param name="values"> Array of values </param>
/// <returns> Average value in the values array </returns>

public double GetAverage(double[] values)
{
   return values.Average();
}

/// <summary>
/// Determines the maximum value from the given array of values </summary>
/// <param name="values"> Array of values </param>
/// <returns> Maximum value in the values array </returns>

public double GetMaxValue(double[] values)
{
   Array.Sort(values);
   return values[values.Length - 1];
}

/// <summary>
/// Determines the minimum value from the given array of values </summary>
/// <param name="values"> Array of values </param>
/// <returns> Minimum value in the values array </returns>

public double GetMinValue(double[] values)
{
   Array.Sort(values);
   return values[0];
}
```

```
/// <summary>
/// Calculates the geometric mean of a given array </summary>
/// <param name="data"> Array of values </param>
/// <returns> Geometric mean </returns>

public double GeometricMean(double[] data)
{
   if (data.Length == 0)
   {
      return 0;
   }
   // calculates the product
   double geoMean = 1.0;
   for (int i = 0; i < data.Length; i++)
   {
      geoMean *= data[i];
   }
   // raise the product to 1/(the number of elements in data)
   geoMean = Math.Pow(geoMean, 1.0 / (double)data.Length);

   // rounding off to one decimal place
   // geoMean = (double)Math.Round(geoMean * 10) / 10;
   return geoMean;
}

/// <summary>
/// Create a generic Criteria div
/// </summary>
/// <returns></returns>

public HtmlGenericControl CreateDiv()
{
   System.Web.UI.HtmlControls.HtmlGenericControl createDiv = new
   System.Web.UI.HtmlControls.HtmlGenericControl("DIV");
```

```
            createDiv.ID = "div";
            createDiv.Style.Add(HtmlTextWriterStyle.BackgroundColor, "White");
            createDiv.Style.Add(HtmlTextWriterStyle.Color, "Black");
            createDiv.Style.Add(HtmlTextWriterStyle.Height, "20px");
            createDiv.Style.Add(HtmlTextWriterStyle.Width, "400px");
            return createDiv;
        }
    }
}
```

ANNEXURE J
LETTER 1 CONFIRMING NO SECURITY TRAINING OFFERED

DUT AppFactory
3rd Floor ML Sultan Campus
Durban

4000

October 26, 2017

Dear Researcher

Durban University of Technology
Ritson Campus
Durban
4000

Re: Information on application development training supplied for research purposes

The following training was completed by software development teams at our Company in the last 12 months:

- WooCommerce Guided Tour Videos
- Introduction to ASP.NET MVC
- Programming in C# Jump Start

[redacted]

Cassim Vanker

Project Manager

DUT AppFactory

Agile Software Developer

LETTER 2 CONFIRMING NO SECURITY TRAINING OFFERED

E4
420 Island Office Park
35 Island Circle
Briardene Durban
4051

25 October 2017

Dear Researcher

Durban University of Technology
Ritson Campus
Durban
4000

Re: Information on application development training supplied for research purposes

The following training was completed by software development teams at our Company in the last 12 months:

No Training was provided

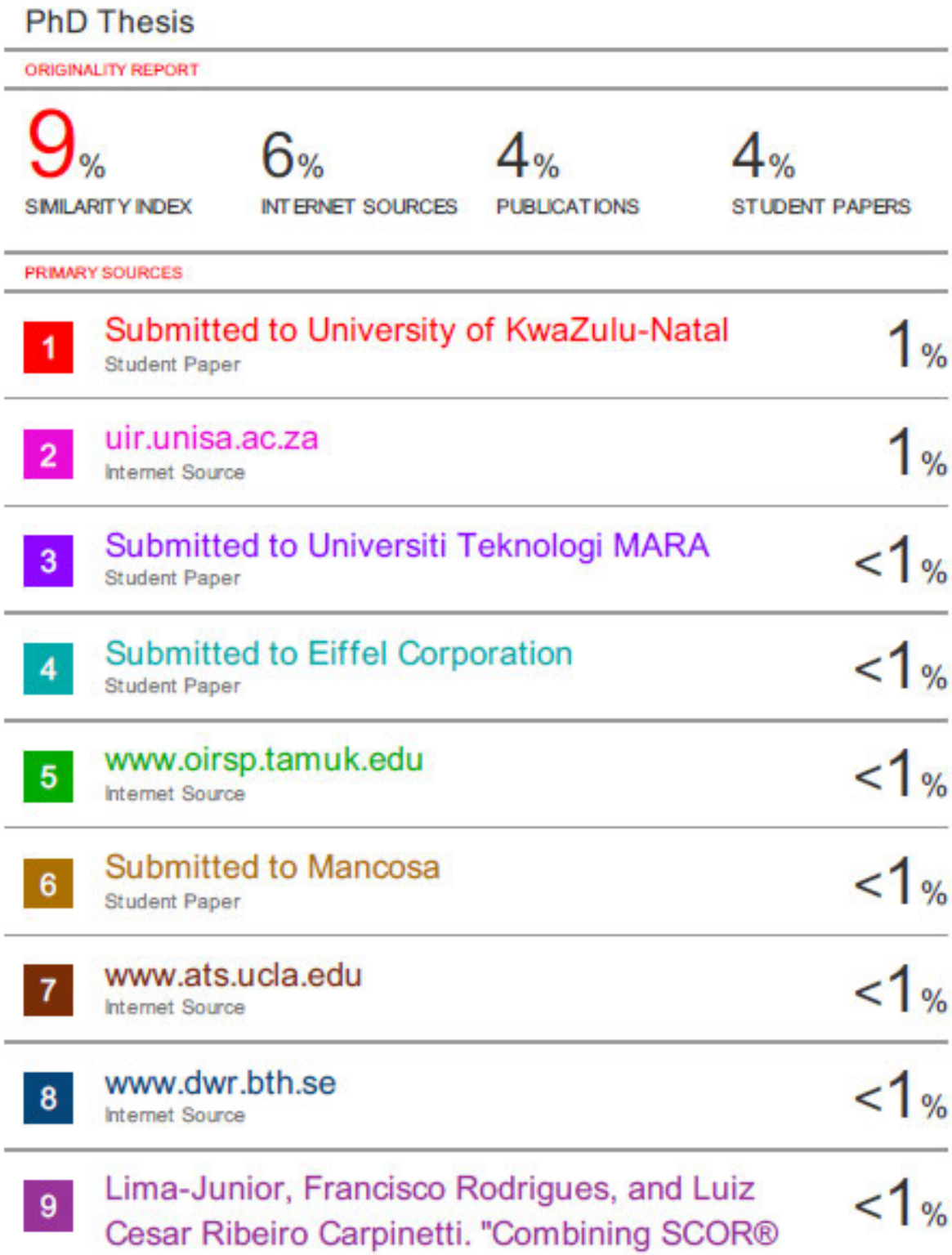
[redacted]

Development Team Lead

E4

ANNEXURE K

TURN IT IN REPORT-COVER PAGE



PhD Thesis

ORIGINALITY REPORT

9%	6%	4%	4%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to University of KwaZulu-Natal Student Paper	1%
2	uir.unisa.ac.za Internet Source	1%
3	Submitted to Universiti Teknologi MARA Student Paper	<1%
4	Submitted to Eiffel Corporation Student Paper	<1%
5	www.oirsp.tamuk.edu Internet Source	<1%
6	Submitted to Mancosa Student Paper	<1%
7	www.ats.ucla.edu Internet Source	<1%
8	www.dwr.bth.se Internet Source	<1%
9	Lima-Junior, Francisco Rodrigues, and Luiz Cesar Ribeiro Carpinetti. "Combining SCOR®	<1%

ANNEXURE L

LANGUAGE PROFICIENY CERTIFICATE

Dr R. Sucheran

Durban University of Technology

Management Sciences

Ritson Campus

15 November 2017

To whom it may concern

Re: N.K.NAICKER. Reg. No. 214585809.

This letter serves to confirm that the language in the PhD thesis entitled, Secure *Requirements Engineering in a Constrained Agile Software Development Environment* has been checked and is of a good standard.

Sincerely

R. Sucheran

PhD

reshma@dut.ac.za

Cell. 083 658 7426

ANNEXURE M

RESEARCH GAP: <u>No formal security approach for Agile RE</u>

Google Scholar

Agile RE requirements security approach

Articles

About 74,500 results (0.07 sec)

My profile My library

Any time
Since 2018
Since 2017
Since 2014
Custom range...

Sort by relevance
Sort by date

include patents
include citations

Create alert

Agile requirements engineering practices: An empirical study [PDF] jayurbain.com
L Cao, B Ramesh - IEEE software, 2008 - ieeexplore.ieee.org
... Some organizations treat these tests as part of **require**- ments specification. Benefits ... Challenges. The participants suggested that their **agile RE** practice focuses more on **requirements** validation than traditional **approaches** ...
Cited by 368 Related articles All 17 versions Import into EndNote

Eliciting **security requirements** with misuse cases [PDF] gannon.edu
G Sindre, AL Opdahl - **Requirements** engineering, 2005 - Springer
... analysis must be embedded in well-documented and tool-supported **RE** methods that ... software development organizations already employ, or are considering employing, **agile** methods [47 ... misuse case patterns and on using them to determine **security requirements** in practical ...
Cited by 1134 Related articles All 33 versions Import into EndNote

Get ready for **agile** methods, with care [PDF] pace.edu
B Boehm - Computer, 2002 - ieeexplore.ieee.org
... Some **agile** methods such as Crystal and DSDM do more architectural preplanning ... On the other hand, if the architecture anticipates and accommodates **re**- quirements changes, plan ... less-than-great developers, refactoring effort increases with the number of **requirements** or sto ...
Cited by 1150 Related articles All 29 versions Import into EndNote

Towards **agile security** assurance [PDF] nspw.org
K Beznosov, P Kruchten - ... of the 2004 workshop on New **security** ..., 2004 - dl.acm.org
... **Re**- quire- ments Review ... to be done to get to the point where tools integrated in development environments would incrementally and continuously check, test, and analyze the various artifacts—code, design, **requirements**—in regards ... 5] A. Cockburn, **Agile** Software Development ...
Cited by 120 Related articles All 20 versions Import into EndNote

Agile requirements engineering practices and challenges: an empirical study
B Ramesh, L Cao, R Baskerville - Information Systems Journal, 2010 - Wiley Online Library
... (AgileConsult). 3. Neglect of non-functional **requirements**. A major concern with iterative **RE** in **agile** development is the inadequate attention given to non-functional **requirements**. They are often ill defined and ignored during early development cycles ...
Cited by 187 Related articles All 4 versions Import into EndNote

Security requirements for the rest of us: A survey [PDF] bibsys.no
IA Tondel, MG Jaatun, PH Meland - IEEE software, 2008 - ieeexplore.ieee.org
... Likewise, **agile** development projects might find abuser stories best suited ... This also makes it natural to tie **security requirements** to the rele- vant functional **requirements**, as Haley and his col ... As we mentioned, we're aware that a lightweight **approach** won't be bullet-proof ...
Cited by 116 Related articles All 14 versions Import into EndNote

Management challenges to implementing **agile** processes in traditional development organizations [PDF] semanticscholar.org
B Boehm, R Turner - IEEE software, 2005 - ieeexplore.ieee.org
... **Agile** is more suited to emerging **requirements** and ca ... evolutionary or iterative process, particularly in long-term estimates, is most likely higher with **agile** ap- proaches ... We're still caught in the certainty/am- biguity conundrum; our business processes and infrastructure **require** near ...
Cited by 475 Related articles All 16 versions Import into EndNote

A systematic literature review on **agile requirements** engineering practices and challenges [HTML] utwente.nl
I Inayat, SS Salim, S Marczak, M Daneva... - Computers in human ..., 2015 - Elsevier
... However, the authors claimed that additional empirical evidence is **required** to demonstrate the validity of ... These include tools for modelling non-functional **requirements** (Farid & Mitropoulos, 2012a, 2012b), a mind-mapping technique to facilitate **agile RE** (Mahmud & ...
Cited by 116 Related articles All 9 versions Import into EndNote

Method and system for rapid accreditation/**re**-accreditation of **agile** IT environments, for example service oriented architecture (SOA)
U Lang, R Schreiner - US Patent 8,856,863, 2014 - Google Patents
... This is particularly relevant for dynamically changing ("**agile**") IT systems ... The mapper models 115 enable model **re**-use, flexibility, technology-independence, and abstraction layer ... The "**requirements** model reader" (520) then reads one or more **requirements** models 120 from the ...
Cited by 18 Related articles All 4 versions Import into EndNote

[PDF] **Requirements** engineering in **agile** software development [PDF] semanticscholar.org
A De Lucia, A Qusef - Journal of Emerging Technologies ..., 2010 - pdfs.semanticscholar.org
... a short report, which recommends whether or not it is worth carrying on with the **RE** and AD ... B. **Requirements** Elicitation In this activity, **agile** teams work with stakeholders to find out about the application ... The most important techniques used for **requirements** elicitation in AD are ...
Cited by 83 Related articles All 9 versions Import into EndNote

Related searches

agile requirements **engineering**
agile requirements **empirical study**
security requirements **analysis**
misuse cases security requirements
agile requirements **communication patterns**
agile security **web applications**
systematic review security requirements
agile **practices large scale** requirements **engineering**
requirements **engineering methodology** agile **environment**
agile requirements **system projects**
security **quality** requirements **engineering**
security requirements **engineering framework**
agile **methodologies** requirement **engineering practices**

Goooooooooogle

1 2 3 4 5 6 7 8 9 10 Next

Help Privacy Terms

RESEARCH GAP: <u>fuzzy TOPSIS not used as a method for Prioritisation</u>