Modelling Security Requirements Through Extending Scrum Agile Development Framework

PhD Thesis

Minahi Hamed Alotaibi

Software Technology Research Laboratory

Faculty of Technology

De Montfort University

This thesis is submitted in partial fulfilment of the requirements for the Doctor of Philosophy

February 2016

To the spirit of my father and my brother

ABSTRACT

Security is today considered as a basic foundation in software development and therefore, the modelling and implementation of security requirements is an essential part of the production of secure software systems. Information technology organisations are moving towards agile development methods in order to satisfy customers' changing requirements in light of accelerated evolution and time restrictions with their competitors in software production. Security engineering is considered difficult in these incremental and iterative methods due to the frequency of change, integration and refactoring. The objective of this work is to identify and implement practices to extend and improve agile methods to better address challenges presented by security requirements consideration and management.

A major practices is security requirements capture mechanisms such as UMLsec for agile development processes. This thesis proposes an extension to the popular Scrum framework by adopting UMLsec security requirements modelling techniques with the introduction of a Security Owner role in the Scrum framework to facilitate such modelling and security requirements considerations generally. The methodology involved experimentation of the inclusion of UMLsec and the Security Owner role to determine their impact on security considerations in the software development process. The results showed that overall security requirements consideration improved and that there was a need for an additional role that has the skills and knowledge to facilitate and realise the benefits of the addition of UMLsec.

KEYWORDS

Security requirements, UML, UMLsec, agile, Scrum, security requirements, Security Owner.

DEDICATION

The work of this thesis dedicated to:

My mother

A strong and gentle who taught me to trust in Allah, to believe in hard work and that so much could be done with less

My father

For being my first teacher, earning an honest living for us and for supporting and encouraging me to believe in myself

My wife

For giving me her love, supporting me in all critical and difficult times

My children

For adding a sweet with love in most of my studies and research days

My brothers and sisters

For supporting me during my educational career

DECLARATION

I declare that the work of this thesis was submitted to De Montfort University for the degree of Doctor of Philosophy in computer security and software engineering in the Software Technology Research Laboratory (STRL), Faculty of Technology, De Montfort University, Leicester, United Kingdom. The work contained herein is my original work with an exemption to the citations and that this work has not been submitted at any other university, either in part or its entirety, for the award of any degree.

ACKNOWLEDGEMENTS

Initially and foremost, praise be to Allah, who blessed me completing this thesis. Thanks to him that honoured me with faith, patience and confidence in order to accomplish this research. Without him I could not complete the work or part of it.

I am grateful with special thanks to the main supervisor Dr. Helge Janicke who was the leader at all stages of my research, who guided me and helped to get me started on my PhD degree, his encouragement and belief in me throughout have been extremely helpful. He was all the time available for any questions, open mind and listen to my queries. He gave generously from his time, even was busy, and vast knowledge. He always knew the answers to problems facing me while leading me to the correct source. Thanks go to the second supervisor Dr. Martin Ward for following me through my research.

I also like to extend my sincere thanks to Professor Hussein Zedan, Professor Hongji, Dr. Platt and Dr. Cau, the previous Software Technology Research Laboratory staff, at De Montfort University, for their guidance, persistent support, and encouragement throughout this research. Moreover, they gave me motivation for starting my new topic and freedom in my research interests. Welcome and good impression when meet me as a new researcher.

Thanks continue to all of the current staff members in the department, Dr. Siewe, Dr. AlBayatti, Dr. Chen, and all others, for their support in helping me and all the researchers.

I would like to thank all researchers and my colleagues of the STRL, for their valuable comments and knowledge sharing especially in my research area. Last but not least, my thanks and appreciation to my family, who are always partners in the achievement and my successes.

PUBLICATIONS

Alotaibi, M. (2014), 'Security Requirements Modelling in Agile Development Methods', paper presented in 7th Scientific Saudi Conference SSC 14 in Edinburgh, UK, 1st-2nd Feb. 2014.

Alotaibi, M. (2014), 'UMLsec in Agile', poster presented in 7th Scientific Saudi Conference SSC 14 in Edinburgh, UK, 1st-2nd Feb. 2014.

Alotaibi, M. and Janicke, H. (2015)**, '** Extending Scrum Framework to Emphasize Security *How a "Security Owner" is integrated in the Scrum Team.*', paper presented in 8th Scientific Saudi Conference SSC 15 in London, UK, 31st Jan -1st Feb. 2015.

Alotaibi, M. and Janicke, H. (2015), 'Integrating Security Owner within the Agile Teams', poster presented in 6th PhD Experience Conference (PhDEC) in the University of Hull, UK, 7th - 8th April. 2015.

CONTENTS

LIST OF APPREVIATIONS

AC	Access Control
ASD	Agile System Development
DoD	Definition of Done
DoS	Denial of Service
E-ESS	E-Exam secure system
GQM	Goal Question Metric
OO	Object Oriented
OWASP	The Open Web Application Security Project
PHRS	Patient Health Record System
PO	Product Owner
RBAC	Role Based Access Control
ROI	Return On Investment
ScrumBan	Scrum development method with Kanban technique
SDL	Security Development Lifecycle
SO	Security Owner
UML	Unified Modelling Language
UMLsec	Unified Modelling Language with security features
WIP-Limits	Maximum number of work-in-progress cards in Kanban
XMI	XML Metadata Interchange
XP	Extreme Programming

LIST OF FIGURES

LIST OF TABLES

CHAPTER ONE

INTRODUCTION

Objectives

- Introduce and motivate the research.
- Describe the aims, objectives and research questions.
- Clarify the contribution and research methodology to be used.
- Present thesis structure.

1. Introduction

1.1 Introduction

Based on the understanding of the potential problems that could arise during the development of any software system and the importance of requirements gathering, security is also considered as a basic foundation in software development. Due to the failure of some traditional software development methods, which includes time restrictions and the need to introduce more flexible or agile development methods, the modelling of security requirements has become an essential part in the rapid development of secure systems. Although advances have been made in software development methods that provide more rapid and agile approaches (Hartson, 2012), (Sheetal, 2012) in order to meet the customer's changing requirements, consideration of security requirements in such methods is lacking. Security engineering in such incremental and iterative methods is difficult, because although there have been some attempts (Kongsli, 2006), (Mougouei, et. al, 2013), (Keramati, et. al, 2008) to consider the analysis of security requirements within iterations, there is a lack of clarity of how this is achieved. Thus, there is a need to extend or improve agile methods of software development in order to cope with the consideration and management of security requirements.

The specific focus of the study is the Scrum software development framework (Schwaber, 2004), one of the agile development methods used for managing software development projects. The Scrum framework is a popular agile method (Cohn, 2010) used by organisations when they transition to agile development because of its ability to respond rapidly to changes in requirements (Eliza et al., 2010, Jakobsen and Sutherland, 2009). The Scrum framework was chosen for this study because the processes that take place within the framework involve communication, interaction, discussion and planning within a team and part of the study aims to assess the impact of a Security Owner role on teamwork. Within this framework there are a number of actors or roles who are responsible for product development, they include the Product Owner who represents the client and understands their needs, the Scrum Master responsible for removing distracting influences from the development team and finally, the development team itself who are responsible for incremental product development. This study proposes a fourth role, namely; the Security Owner who will be responsible for security considerations in all parts of the development process. UMLsec is included because it is a tool that emphasises the consideration and modelling of security requirements.

1.2 Research Motivation

In agile development the quality and success of software development depends on the quality of gathered requirements and refinement of those requirements in each agile iteration. The highest priority is to satisfy the customer through early and continuous delivery of valuable software (an agile principle) and the quality of the requirements will determine the success and the failure of the software. These points are also relevant to security requirements and there is a need for a suitable person responsible for overseeing security requirements gathering as well as coordination of such requirements throughout the development process. It is also proposed in this research that the role be responsible for security testing and liasing with the product's customer. Moreover, this role will be responsible for ensuring rigour in the development process and will also help to create a culture of rigour among the team in relation to security requirements in order to achieve a more effective development process in terms of security requirements resulting in a more secure product.

This study is concerned specifically with security requirements which are often neglected in most software systems (Ramesh et al., 2010). Security requirements have been considered at the end of the software development process and there is less emphasis placed during the inception stages of the development process. Security challenges that are present in today's world require that security requirements gathering should be an integral part of any software development process (Rehman et al, 2013, Security Innovation white paper, Calder and Watkins, 2012). The motivation of this study is to include a new security function to the development process, specifically, a Security Owner within the Scrum team. This motivates to investigate the suitability of a modelling tool, namely UMLsec, and the proposed techniques in the current study for security requirements modelling in the Scrum process, and if the use of UMLsec can assist the Scrum team to improve software security and that the Scrum framework remains agile while considering security features through the use of UMLsec.

Scrum teams need to know which security requirements to consider at each iteration or sprint and how to formulate the chosen requirements. At present the responsibility for security requirements lies with all of the roles in the Scrum team, and while it is necessary that they are aware of such requirements and that they are responsible for implementation, security requirements are often in conflict with other requirements as their business value is ill understood. To overcome this conflict, this work proposes a centralised role responsible for overall coordination of security requirements between various team members. It is granted that Scrum, as an agile approach to development, allows for communication between members on security issues, however, there is no role responsible for coordinating security requirements. Moreover, the introduction of such a role, proposed in this study as a Security Owner (SO) will bring more understanding of security requirements from a project-wide perspective and more expertise about security. Specifically, the

SO will be the point of contact in the Scrum team who is aware of concepts related to threat modelling, security vulnerabilities and security code review checklists.

1.3 Research Aims, Objectives and Hypotheses

Aims

1. To develop methods by which Scrum can be improved to address security requirements early and throughout the development process.

2. To identify best practices and techniques that facilitate the security focus of an agile team.

Objectives

1. To establish if the inclusion of a Security Owner role will improve security requirements consideration through identifying and managing security requirements, negotiation with Product Owner (PO) and supporting the Scrum team. This objective addresses aim 1.
2. To see if the Security Owner, is able to improve security requirements consideration in agile methods.
3. To investigate the suitability of UMLsec for integrating techniques for security modelling in the Scrum process (Chapter 4). The success criteria here are that the use of UMLsec assists the Scrum team to improve software security and that the Scrum framework remains agile while considering security features through the use of UMLsec. This objective addresses aim 2.

4. To conduct experimentation of extended Scrum framework to determine the benefits of Security Owner role and UMLsec (Chapter 5). The success criteria of the experimentation includes that it shows validity in revealing the opinions of the participants in relation to how the Security Owner role and UMLsec have an impact on the process.

Hypotheses

To validate if the inclusion of a Security Owner role improves the security of applications developed using Scrum and UMLsec. This can be summerised in the following testable statements:

H1. The introduction of a Security Owner role helps the Scrum team in terms of effectiveness when considering and managing security requirements (when using UMLsec)

H2. The Security Owner role helps to identify security requirements

H3. The Security Owner role will help team members with UMLsec in terms of team working

H4. The Security Owner role will help team members to prioritise security requirements using UMLsec

H5. UMLsec is an appropriate tool for modelling security requirements (when facilitated by a Security Owner)

H6. Scrum as an agile process remains agile when UMLsec is used when / if facilitated by a Security Owner

1.4 Research Questions

Based on the discussion above, the proposed research seeks to find answers to the following questions:

1. The introduction of UMLsec and a Security Owner role will extend and improve existing agile methods (Scrum framework) in modelling security requirements.
2. The introduction of the Security Owner role will assist other Scrum team members in relation to security requirements.
3. UMLsec can assist the Scrum team to prioritise and apply security requirements.
4. Agility of the Scrum framework remains the same with the integration of UMLsec.

The answers to these questions are addressed in the following chapters. The contributions expected in this research are listed in the forthcoming section.

1.5 Research Contribution

The main two contributions of the research are that it has shown that the Scrum framework can be extended through the addition of a new role, namely the Security Owner role, to the standard Scrum framework and the introduction of UMLsec as a tool for modelling security requirements, and that, as the second contribution, the integration of the Security Owner role and UMLsec have improved the process of agile software development leading to a more secure product. Further contribution include the introduction of a Security Owner role which has been beneficial in an agile development process taking into consideration management and coordination, and implementation of security requirements as well as assisting other Scrum team members in relation to security requirements. Moreover, the introduction of UMLsec to the Scrum framework allowed team members to be more aware of security requirements.

1.6 Research Methodology

Depending on the empirical research guidelines in software development (see Chapter three), the research methodology for this study is mixed (quantitative/qualitative). Specifically, the quantitative/qualitative methods employed are a review of the literature, extended Scrum framework design, experimentation and analysis of the collected data.

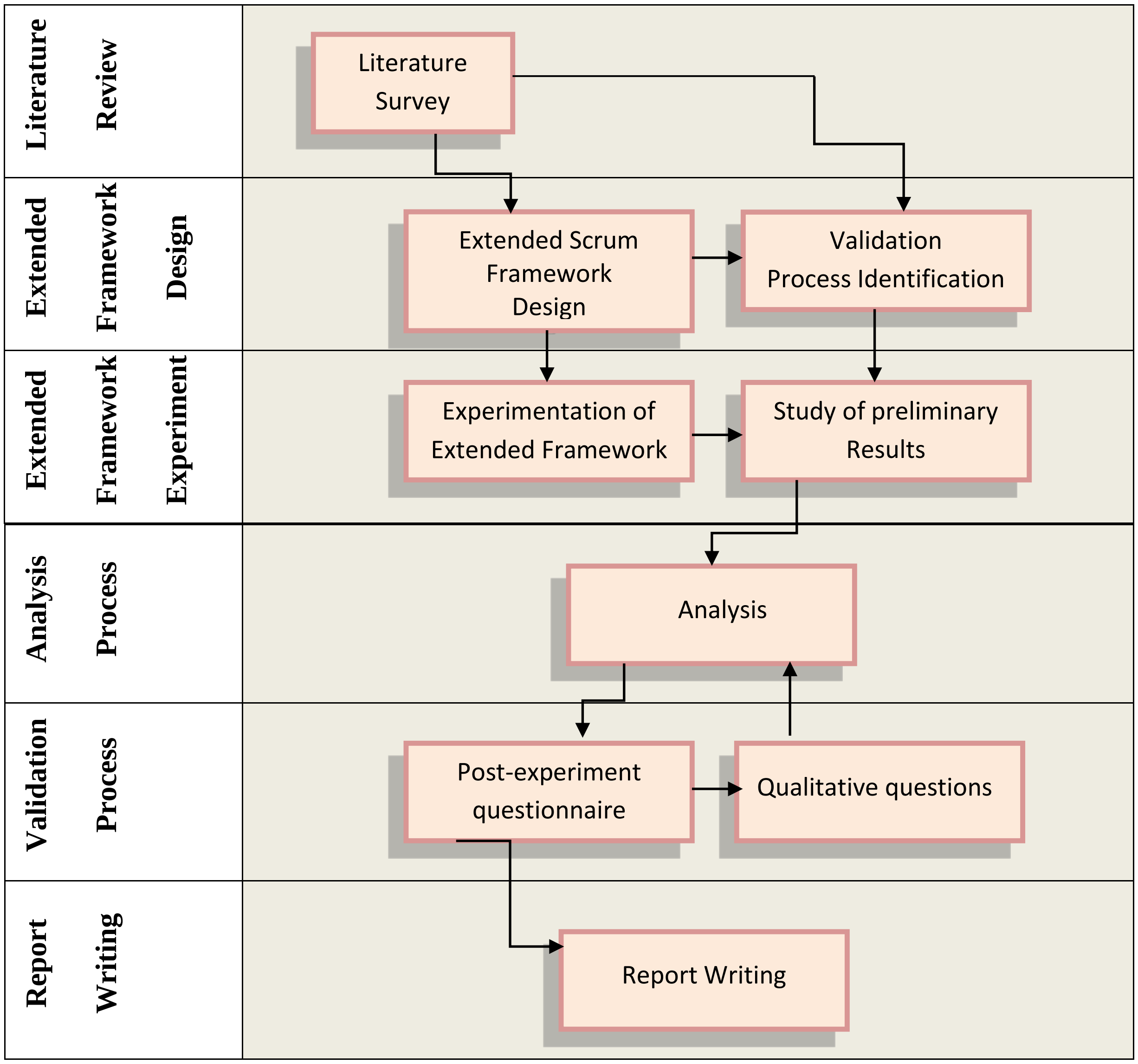

Literature Review
Extended Framework Design
Extended Framework Experiment
Analysis Process
Validation Process
Report Writing
Literature Survey
Extended Scrum Framework Design
Validation Process Identification
Experimentation of Extended Framework
Study of preliminary Results
Analysis
Post-experiment questionnaire
Qualitative questions
Report Writing

Figure 1.1: Framework of the research

Based on the above research framework, the research will be conducted through development of research methods (chapter three) that will facilitate the evaluation of the proposed integration of the Security Owner role and UMLsec in an extended Scrum development framework.

1.7 Thesis Outline

The work of this research is structured in the following chapters:

- **Chapter One**

Chapter one introduces the research and motivates the current research. It describes the key aims and objectives behind this effort (section 1.3). In addition, it presents the key research questions (section 1.4) and contribution (section 1.5) and the research methodology (section 1.6). This chapter concludes with a presentation of the thesis structure (section 1.7).

- **Chapter Two**

Chapter two is a review the literature related to this study of requirements gathering, Unified Modelling Language (UML), misuse cases and UMLsec for modelling security features. The chapter also reviews security requirements consideration, agile software development, its success and failure factors, teamwork in agile, and the Scrum framework.

- **Chapter Three**

This chapter describes the philosophical perspectives of quantitative and qualitative research and research methods, developing the conceptual research framework and empirical research in software development (section 3.3). It presents the research design (section 3.5) and describes these research methods for applicability to the current research in section 3.6. Section 3.7 describes the experiments and justification for the real experimentation. Furthermore, it introduces the data analysis methods and concludes with a summary.

❖ **Chapter Four**

This chapter presents the proposed framework for the inclusion of a Security Owner and UMLsec. As part of the framework, maintaining a Security Backlog as a new proposed artefact used by the Scrum team as well as misuse cases in the Scrum framework are demonstrated in section 4.7. Section 4.8 explains security requirements in planning poker followed by adoption of UMLsec and security requirements in agile methods in section 4.9.

❖ **Chapter Five**

This chapter presents the evaluation experiments of adopting UMLsec and Security Owner role in agile methods. Section 5.2 describes the training course followed by experiment development and piloting in section 5.3 and then experiments procedure in section 5.4. It concludes with a summary in section 5.5.

❖ **Chapter Six**

Chapter six presents the collected data analysis and the associated discussion. This discussion is based on the participants' perceptions and their experience when they involved in the expermintation. Section 6.2 is to analyse and discuss the results of identifying security requirements, followed by managing and modelling of security requirements in sections 6.3 and 6.4. Then the teamwork is discussed in section 6.5 to see the effect of the inclusion of a Security Owner role to the Scrum team. Section 6.6 discusses Prioritising of security requirements followed by the agility discussion in section 6.7 and then conclude with the chapter summary in section 6.8.

- **Chapter Seven**

  Chapter seven summarises the thesis and presents the research conclusions. The chapter also highlights the limitations of the research and presents the thesis recommendations and future work.

The following figure (Figure 1.2) illustrates the thesis outline.

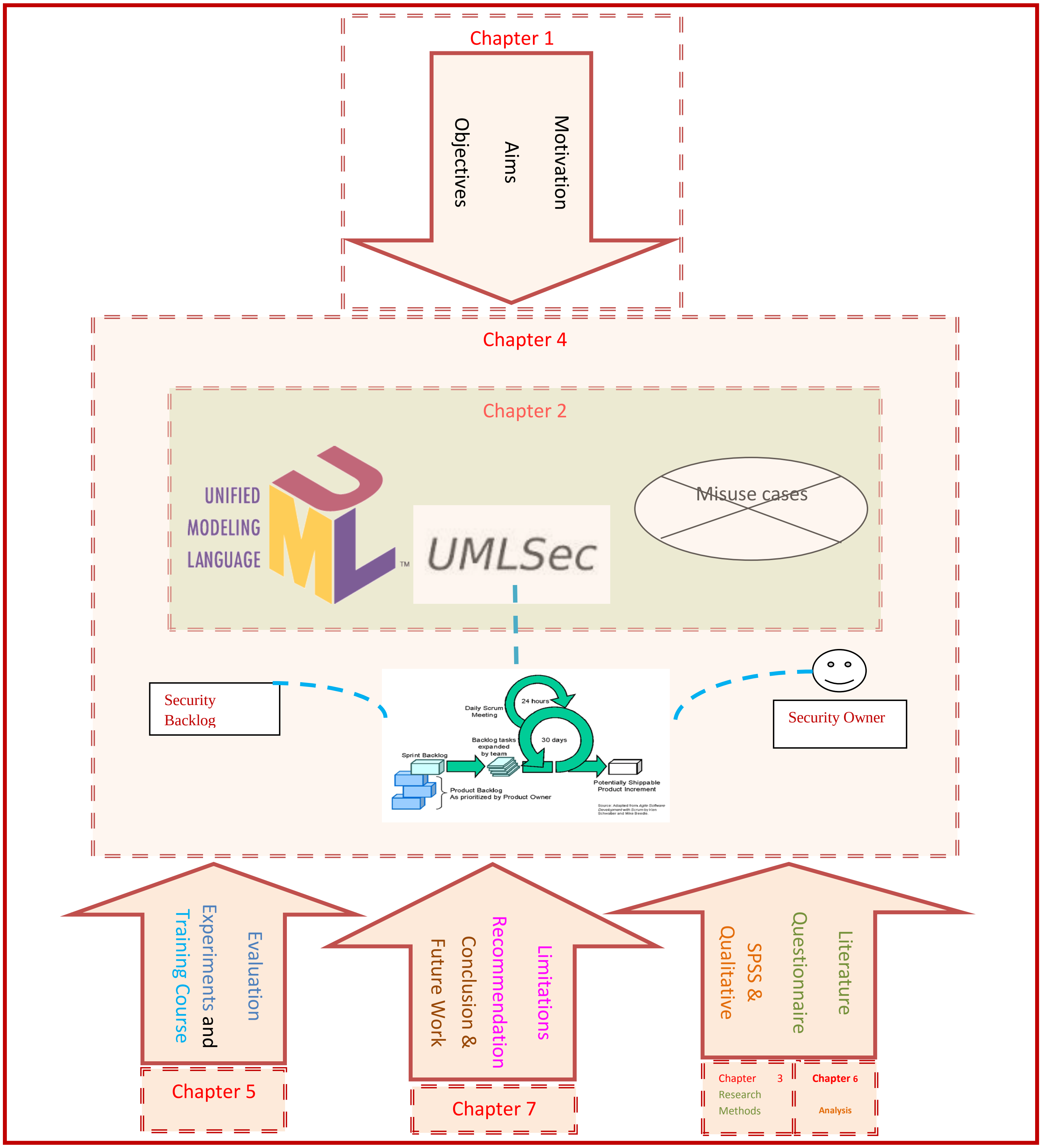


Figure 1.2: Thesis Structure

CHAPTER TWO

LITERATURE REVIEW

Objectives

- Critical review of the efforts considering and integrating security into agile methods to establish the context of this work.
- Reviewing agile success factors, teamworking and Scrum development framework.
- Present UML and UMLsec when modelling security requirements towards motivating this research.

2. Literature Review

2.1 Introduction

The aim of this chapter is to provide a critical review of the literature related to the relevant areas to this research. This review is concerned with security requirements in agile methods and the use of use and misuse cases in requirements, UML modelling and UMLsec. These as relevant areas for the extension of agile methods security requirement modelling through UMLsec and a Security Owner role. Furthermore, there is a review of the different approaches to considering and integrating security into agile methods. Specifically, the areas that are covered in the review of the literature include the areas that are relevant to agile development and security in agile. The review includes literature related to requirements gathering in software development (section 2.3), UML as a tool for modelling security requirements (section 2.4), the use of use and misuse cases for eleciting requirements (sections 2.4.1 and 2.5.1) and UMLsec as an extended tool for modelling security requirements (section 2.5). Moreover, because the study aims to extend and improve agile methods in relation to security, there is a review of agile methods and the efforts that have been made to integrate security into such methods (sections 2.6 and 2.7). Finally, the study proposes to extend and improve the Scrum agile development method and there is a review of literature related to Scrum including functional roles and processes (section 2.8).

2.2 Literature Review Method

The aim of the literature review is to review the available literature on the topic of security requirements, agile development methods, UML, UMLsec and use and misuse cases. The papers that were reviewed were selected based on the relevance to these areas and the overall subject of security requirements in agile development methods. More specifically, the papers were also considered in light of the hypotheses of this study which included the following:

H1. The introduction of a Security Owner role helps the Scrum team in terms of effectiveness when considering and managing security requirements (when using UMLsec)

H2. The Security Owner role helps to identify security requirements

H3. The Security Owner role will help team members with UMLsec in terms of team working

H4. The Security Owner role will help team members to prioritise security requirements using UMLsec

H5. UMLsec is an appropriate tool for modelling security requirements (when facilitated by a Security Owner)

H6. Scrum as an agile process remains agile when UMLsec is used when / if facilitated by a Security Owner

The search for the papers included a manual search of a number of different databases which include ACM, IEEE, ScienceDirect, EBSCO, the University library database and Google Scholar. The goal was to find articles which are related to the current and most recent developments in the established areas for review. Additionally, there are numerous online qualitative research methods. The online resources, e.g. blogs and online surveys, are useful in extending the outreach and collecting data from large geographical areas.

It was important to use keywords that would derive the results required. These included single search terms such as ‘agile’ and word combinations such as ‘agile + UML’, ‘agile + security’ and ‘security requirements modelling + UMLsec’. Because of the nature of agile development methods and the ever changing security requirements of software, it was important to consider papers post 2000. However, earlier papers were considered where they were relavant and useful to the study, for example Basili (1996) that informed about empirical research in software development. All of the papers had to be published, however, websites with articles written by industry professionals were also used to contribute to the review.

As part of the inclusion criteria, it was important to include articles where researchers have looked at the issue of security requirements in agile methods, articles were included that proposed models, theories and ideas and papers that particularly identified the problems associated with modelling security requirements in agile methods. In order to determine the quality of the articles, it was necessary that they had research questions and aims that were clearly defined. Also, they gave a clear explanation of the problems that were associated with security requirements consideration in agile methods. In addition to their approach including the methodology was clear and could the ideas be applied to this study.

2.3 Requirements Gathering

It has been noted in Ambler and Lines (2009) that the requirement gathering stage is very essential and a major phase in developing software. If it is done in the right way putting all the standards at the inception stage, the entire development cycle will be easier and understandable to all developers. Motivation for the need to consider requirements is based on different experiments that have been done in this field, for instance, it has been noted that poor requirements gathering will translate to the failure of most software. Moreover, according to Pandey and Mustafa (2010) half of all failures related to software development are attributed to poor quality of requirements.

This was confirmed by the fact that 'over seventy percent of all software development failed in 2000 companies due to poor requirements gathering, analysis and planning' (Pandey and Mustafa, 2010 p. 1079). Moreover, over sixty percent of failures in software development projects in the United States was mainly caused by poor requirements and in addition to requirements gathering, it has also been revealed that creeping requirements in most cases account for more than eighty percent of project schedule overruns (Pandey and Mustafa, 2010).

One reason why it is important to fix software related problems at the earlier stages of development is because it is more cost effective than fixing the problem at the end. This is the case with the traditional waterfall method where actual development is seen later in the process so if mistakes are discovered, although the waterfall method does allow returning to earlier stages, fixing any problems would require costly rework (Munassar and Govardhan, 2010). Unlike agile development methods the waterfall approach does not allow for the iterative nature of exploratory development (Munassar and Govardhan, 2010). Agile can respond to changes in requirements and thus consider security requirements at all iterations. To this end, many technological affiliated designers and architects have begun to consider security requirements at an earlier stage of the software inception process (Ambler and Lines, 2009).

The first stage of ensuring the security of a software project is to gather security requirements which are non-functional requirements which basically say what a system should not do as opposed to functional requirements which establish what the software should do (Jain and Ingle, 2011). The gathering of security requirements allows the consideration of the malicious aspects of the environment in which the software functions that can be nullified and includes consideration of integrity, confidentiality, authentication, authorization and availability to be included during requirements gathering (Jain and Ingle, 2011). Spears and Parrish (2013) say that requirements gathering has also been focused on functional requirements and has neglected security requirements but say that reducing information security vulnerability in software is a difficult task that is faced by organisations. Mellado et al. (2010) bring attention to the fact that due to the increasing complexity of information systems (IS) there is an increasing risk of security breaches and therefore, changing security requirements. However, security is often not a priority for the developers in a software project and more attention is given to functional requirements than security requirements, and importantly security requirements are often not properly understood (Mellado et al., 2010). Once requirements have been gathered, as part of the development process they need to be modelled. A commonly used tool for modelling requirements is UML, presented in the following section.

2.4 UML Modelling Framework

Unified Modelling Language (UML) is a standard notation language used to specify, construct, represent and document models of complex software and other non-software systems, that help to structure, analysis and design their requirements (Pooley and Wilcox, 2004). According to Bennet et al. (2010) developing software has become a very complicated process because of various aspects such as advancement in technology, different requirements of end users and the size of projects. Because of an increase in the level of knowledge among users, the requirements of software have become more and more sophisticated (Bennet et al., 2010). In addition to that the

UML used as a solution to the software and business problems, it also enhance and improve the communication between the people who engaged to solve and analyse these problems (Pooley and Wilcox, 2004). This is required in the present study as techniques used by the agile teams (will be seen in chapter 4) when considering security requirements and negotiate to prioritise their requirements. Agile teams should know how to use UML and apply its diagrams in order to be able to understand and show the potential links between different diagrams. This can be achieved through choosing a suitable process of modelling, for example, which diagrams are helpful to address their problem (Pooley and Wilcox, 2004).

As a response object oriented (OO) analysis and design are usually aligned or based on different tools such as class templates/diagram, object diagrams and object state diagrams. Most of the class diagrams/templates usually offer vital information that is related to major concepts in the problem domain and their connectivity. Object diagrams aid in creating interfaces between the various class diagrams. According to Daniel (1995) an object state diagram is used in creating dynamic behaviour. It also helps in describing any possible state of an object. An integrative approach is usually used in summarising the use of Object-oriented analysis and design. Based on this approach, the output of one stage is usually taken as the input of the next stage; this is a refinement process that continues in a progressive manner. The initial step in an object state diagram is to evaluate and formalise each object`s life cycle. The next step involves definition of class relationships. In other words, each service offered by a class is supposed by definition. The final step involves completion of class and object diagrams along with the class templates. Agile teams can use object state diagrams iteriatively because the product dynamically changes due to changing customer requirements.

According to Bennet et al. (2010) in order to have a complete picture of the system, the models are usually put in position in order to visualize a specific element of the entire system. Below is a breakdown of various diagram types. Some of them are detailed as integration parts for agile teams when using UML for modelling (chapter 4, section 4.9.1.1).

Behaviour diagrams which are essential in evaluating various behavioural aspects of a system. They include state machine, activity, the four interaction diagrams, and the case diagrams.

Interaction diagrams which are mainly used in the modelling process of object interaction. They include a subgroup of diagrams aligned to the behaviour of the system such as interaction overview, communication, timing, and sequence diagrams.

Structure diagrams which are mainly used in the modelling process of the system, especially on the aspect of its structure. They include class, object, composite structure, deployment, component, and package diagrams.

Many concepts and views have been given related to what artefacts would constitute a model, especially one that is based on both object oriented analysis and design processes. According to the assumption of Bennet et al. (2010), a model is usually an idea of a system or a sub-system from a specific viewpoint.

2.4.1 Use Cases

Use-case analysis is a useful method that is used to document different requirements and has helped to enhance shared understanding between the developer and the stakeholder about requirements, and helps non-technical stakeholders to understand the resulting documentation unlike other approaches such as the educator approach (Kanyaru and Phalp, 2009).

Use-cases are essential, especially in documentation of various system functional requirements which can be described in terms of actors and use cases (Gomma and Olimpiew, 2005).

Nevertheless, there are certain issues related to use-case approaches in the engineering of requirements. These include that it does not allow for a clear definition of system boundaries because it does not establish the scope of the system, for example, a system could be a computer system, and application, a component or an entire business enterprise, thus the actors and use cases for one system boundary may not be applicable to another system boundary (Lilly, 1999). Moreover, although use cases are useful for eliciting functional requirements, an issue that is usually of concern is that they do not offer sufficient support for non-functional attributes of a system, such as security requirements (Jonstone, 2011).

According to Bennet et al. (2010) there are different notations used to show the link between different use-cases and they all represent high level user goals that every system must accomplish. Based on a use-case diagram, one can identify different information such as what the system must do based on the perspective of all users, how the system must or should behave from the standpoint of different users, and finally the functional requirements among other information.

Use-case diagrams are not only used in representing specific analysis of output in an object-oriented system, but is also used in a variety of other elements attributed to system analysis. For instance, it is used in the process of business modelling among other areas as described by Barn (2007). Barn’s (2007) discussion is based on a meta - model that is aligned to a business process model. He has also offered various aspects of mapping between use-case modelling and business process modelling. Dijkman and Joosten (2002) have also offered a good introduction to business use-case and vital information on use-case modelling used in business process modelling. Both Barn (2007) and Dijkman and Joosten (2002) have used the idea of activity diagrams in representing business processes. This helps in modelling all decisions related to information in different steps. Use cases are used for eliciting requirements and as show in section 2.4 UML is a tool for modelling requirements. In the following section an extension of UML, UMLsec is presented which focusses on modelling security requirements.

2.5 UMLsec

UML is considered the de facto standard for object-oriented modelling and a large number of developers are trained to use UML, however, in order to support using UML for secure systems development it is necessary to use an extension of UML, namely; UMLsec (Jurjens, 2005A). This allows knowledge about security engineering to be encapsulated to be made available to developers who may not have specialist knowledge of security (Jurjens, 2005A).

Numerous proposals (Keramati et al., 2008, Matulevičius and Dumas (2011) have been offered in order to address security in software development. One of the most common offerings according to Jürjens (2002) is UMLsec. There is still a major concern on the need to look at security requirements mainly on gathering mechanisms such as UMLsec and abuser stories for agile development processes (Peeters, 2005) in addition to extending agile methods in order to model security requirements using UMLsec as proposed in this study.

When UML is used correctly, it will allow analysts to express security requirements within the diagrams of a UML system. One aspect of UMLsec is the extension of the UML profile. This is done by the use of tagged values, stereotypes, and constraints (Matulevičius and Dumas, 2011). Constraints are used to specify the security requirement of the diagrams. Most specifications of threats usually correspond to various actions by adversaries. Based on the strength of adversary actions, threats in different situations can be specified.

Houmb et al. (2010) say that there is a lack of security expertise in system development teams regarding UMLsec because it is phrased in security domain terminology and therefore, difficult to understand. Moreover, those responsible for coding and those responsible for requirements engineering activities do not have much contact between them, this is especially the case with security requirements (Houmb et al., 2010). In light of these problems Houmb et al. (2010) offer a solution that develops the ability of development team members through providing them with tools that allows them to use UMLsec and solve security requirements themselves. Although team members are provided with a tool that to some extent alleviates the need for expertise, it places the responsibility of security requirements on team members using a tool that ususally requires experts. In the present study a Security Owner is introduced to help teams to use UMLsec and identify security requirements, taking the responsibility of coordinating this out of their hands.

Stereotypes are one subset of UMLsec and are relevant to this study, for example, access control (AC), as a subset, usually has <<RBAC>> (Role Based Access Control) as its constraints and tag values in a system. The work of this stereotype in most businesses is to enforce RBAC in processes, and this is usually specified in the activity diagram of a system. This subset has other three key tags related to its functions and they include {role}, {protected}, and {right} (Jürjens, 2002). The {protected} tag is used to show the states of activities in each diagram. Additionally, it also shows whose access requires some protection. For the {role} tag, there are other pairs usually associated with it such as (actor, and role). The actor is a performer in the activity diagram, whereas the role is just a role in the diagram. Finally, the {right} tag also has a list of pairs associated to it. These include (role, and right). The work of role a pair it to represent the role function in the diagram, whereas the right pair is used to represent accessibility rights to any resource that is protected in the diagram. According to Matulevičius and Dumas (2011) for constraint requirements it is important for the actor in the activity diagram to execute those actions that allow appropriate right of access.

All the tags and the stereotypes discussed above are important because they help in formulating security requirements and other assumptions that are important to the system environment. Most of the constraints discussed above offer criteria to determine whether the system diagrams are able to meet all the requirements. UMLsec features include the following:

- Evaluating specifications of UML in order to determine design`s vulnerabilities
- Summarizing key rules based on practical security engineering

- Offering security starting from the inception stage of the design based on the system setting
- Making verification easier
- Availing all the necessary requirements for developers, especially those not specified in security
- Offer UML Extension mechanisms such as tagged value, stereotype, profile, and the constraints

(Jürjens, 2002, Alhir, 2002, Leiwo and Virtanen, 2005).

In addition to these features there are other concepts provided by UMLsec which include the following:

- Activity diagram; which is responsible for securing the control flow and coordination of the activities
- Class diagram; which is responsible for data exchange and preserving the security of this data
- Sequence diagram; responsible for critical security interaction in the system
- State chart diagram; responsible for any preserved security within the entire object
- Deployment diagram; responsible for all requirements for physical security
- Package responsible; for all holistic view related to the security issues

These concepts and techniques can be used by agile teams and are discussed in more detail in chapter four.

Apart from these features and aspects discussed, there is also the verification framework related to UML. This verification is done in order to support the creation of programmed requirements' analysis tool for the diagrams. This helps in connecting the framework to the industrial CASE tools by the use of XMI, the importance of this connection is that it makes data easily accessible (Jurjens, 2005A).

Overall, UMLsec has been shown to be an effective tool for modelling security requirements and this study proposes to use UMLsec to extend the Scrum development framework as an agile method. However, as shown the literature above UMLsec is phrased in security terminology and therefore requires a role that helps in the use of UMLsec when modelling security requirements. Chapter Four will provide a framework for the inclusion of UMLsec in Scrum and the inclusion of the required role, namely the Security Owner, to facilitate UMLsec as part of the role. Chapter Five presents the experimentation required to support the hypotheses of the study that these inclusions will extend and improve the Scrum framework in terms of security.

2.5.1 Misuse Cases

Misuse cases are a technique used to capture security requirements (Johnstone, 2011) and can be considered as a use case from the viewpoint of an attacker of a system (Alexander, 2003). Misuse cases are about describing the system behaviours that the users would think unacceptable. An attacker is defined as an actor and in the misuse case approach each actor as a threat is defined as well as its goals in misusing the system (Peterson and Steven, 2006). One of the advantages of misuse cases is that it allows security considerations to be injected into a project in the early stages of development instead of fixing security problems using security mechanisms at the end (El Attar, 2012 A). However, it is important that high quality misuse case models are developed to avoid vulnerability in the developed software product (El Attar, 2012 B)

Misuse cases should include detail enough to drive the design activities, this is achieved through considering potential attacks, such as theft, privacy violations and denial of service (DOS) attacks, moreover, misuse cases stop the modelling analysts from becoming mired in technical details that are not important. Much like use case models, misuse cases are iteratively refined throughout the development process (Peterson and Steven, 2006). Similarly, Firesmith (2003) draws a distinction between the purpose of misuse cases and security use cases and says that although misuse cases are suitable for analysing threats, they are not suitable for specifying security requirements which protect against these threats.

It is important to note that security is different from other specifications because it is about somebody trying to attack the system and the employment of misuse cases to analyse potential system scenarios can assist in mitigating these threats and thus enhance security (Alexander, 2003). Hope et al. (2004) start by introducing the idea that software products are developed and sold on the premise of what they can do, i.e. their features, however, more customers are thinking about security and reliability and therefore, in order to build for security and reliability there is a need to anticipate abnormal behavior, something that is not normally described in software development where the focus is on what the product can do. Hope et al. (2004) say there is a need to consider non-normative behavior through the use of misuse cases. In reference to security and reliability, both as non-functional requirements, Alexander (2003) says that misuse cases can be applied to reliability much in the same way they are applied to security in that reliability requirements can be elicited through analysing threats to reliability. Therefore, misuse case can be employed by those invovlved in the development of software, such as agile team members, in order to identify security threats and the associated security requirements that are needed to inform security requirements modelling.

It is therefore important to offer enough features and concepts in order to ensure security from the initial stage of analysis. This is vital in stopping security being a ad-hoc or add-on feature rather than a quality feature that requires equal treatment throughout the entire process of development. In relation to this idea, according to Sindre and Opdahl (2000) many security approaches aligned with industries usually come from the solution world rather than from the problem world. Based on good requirements engineering, it is important to have a thorough understanding of the potential problems at hand before initiating a solution to that problem which the misuse case approach helps to achieve. The diagram below (Figure 2.1) shows misuse cases against use cases, to provide an example from the figure; a use case would be ‘order goods’ and the associated misuse case would be ‘steal card info’.

.

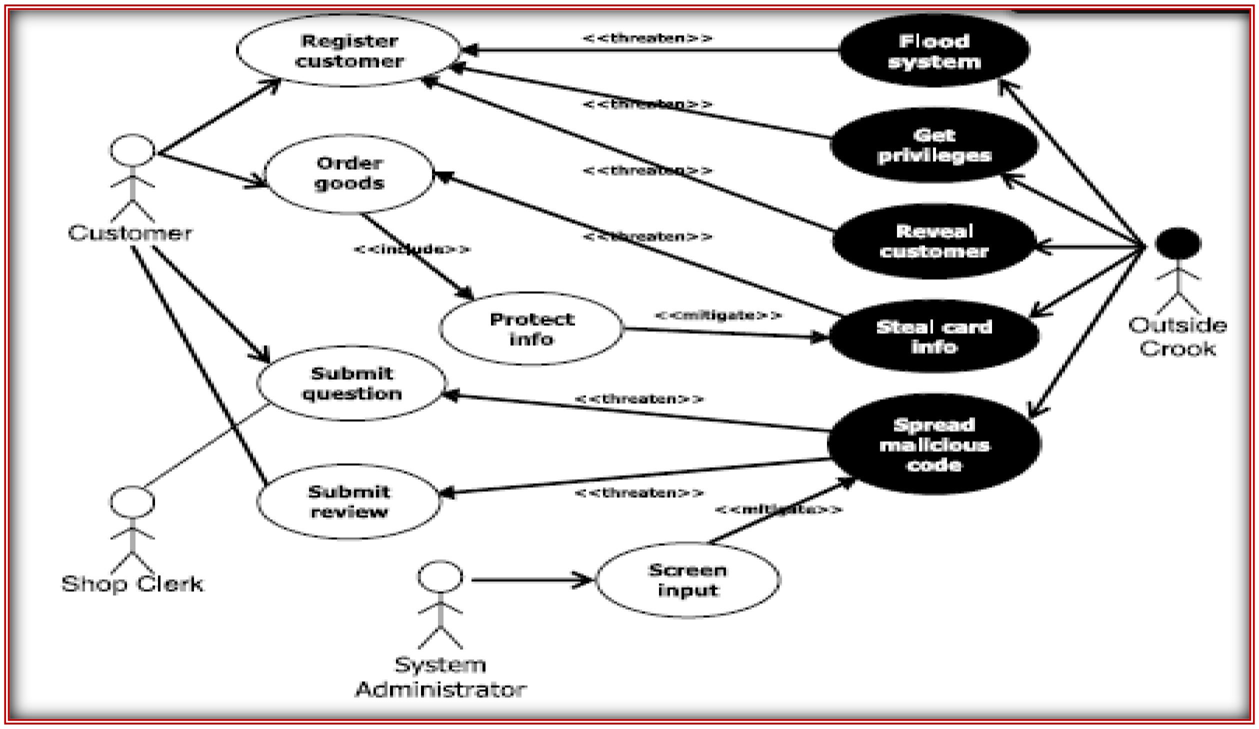


Figure 2.1: (Mis) use diagram taken from (Sindre and Opdahl, 2000)

According Sindre and Opdahl (2000) when modifications are done on use-cases as a response to misuse cases, they can help in the integration part of both the functional and non-functional requirements. This is also an important aspect, especially when it comes to the security requirements of the software development process. Spears and Parrish (2013) show how a UML use case diagram can be transformed in to a misuse case through the identification of threats to use cases and how this can be used to analyse and document threats.

It is also worth to mention the “attack trees” method equally employed as a method of integrating security in agile (Sohaib and Khan, 2010). Opdahl and Sindre (2009) conducted experimentation for the effectiveness in terms of the number of threats found, the types of threat found and user perceptions of attack trees and found that attack trees were effective in terms of identifying threats.

This idea is supported by Ouchani and Dubbabi (2015) who present attack trees as a way of specifying, verifying and quantifying security where nodes represent attacks

Jurjens (2005B) refers to goal-oriented development techniques that integrates goal trees in development that is UMLsec-based in order to consider security at the different stages of requirements engineering.This goal-oriented technique is used to establish security threats which are analysed in threat trees, thereafter, security requirements can be analysed using formal analysis tools to see if they answer the identified threats (Jurjens, 2005B)

However, there are criticisms of attack trees which include that they involve computational complexity and thus building a complete attack tree takes time and they also need considerable resources (Kotenko and Chechulin, 2013). Furthermore, there is inflexibility where changes need to be made to the components and the links between them and the attack trees require reconstruction (Kotenko and Chechulin, 2013).

The attack tree is essentially risk-based modelling which places an emphasis on the modelling of threat scenarios (Schieferdecker et al., 2012). Another problem with the risk-based approach is that information security risk analysis cannot be applied to the complete life cycle of a product because the context of usage could change or new vulnerabilities could arise (Schieferdecker et al., 2012).

Overall, misuse cases are considerd in this study because they can be used by agile team. Their benefits include that they are a quick way to capture and elicit security requirements and can be used as stories in agile. Moreover, they can be understood by team members when facilitated by a Security Owner.

2.6 Agile Software Development

Many researchers have tried to define agile and agile software development but no common agreement has been reached. A definition that conforms to the principles and values of agile manifesto stipulates agile software development as an iterative and incremental (evolutionary) approach to software development which is performed in a highly collaborative manner (Moniruzzaman and Hossain, 2013). This is achieved by self-organising teams within an effective governance framework with 'just enough' ceremony that produces high quality solutions in a cost effective and timely manner which meets the changing needs of its stakeholders (Moniruzzaman and Hossain, 2013). According to Petersen and Wohlin (2009), agile software development is normally a section of software creation methods founded on incremental and iterative development, where requisites and solutions develop through alliances between self-categorised and cross-functional groups. The present study aims to improve these iterative processes carried out by teams by improving collaboration and specifically, the identification and management of security requirements by these teams, such improvement in security requirements will contribute to more timely and cost effective solutions.

In the field of software engineering, agile software development has remained a key paradigm, being highly and intensively used by industry to come up with reliable and secure software (Dyba and Dingsoyr, 2009). Traditionally, software development engaged a general paradigm of necessities which include design, build, and maintenance. Such methods are equally referred to as waterfall-based, plan-driven, documentation driven, big design upfront, and heavyweight methodologies among others (Boehm and Turner, 2004). Unlike these traditional methods, the agile approach to software creation is entirely based on the perception that software prerequisites are entirely dynamic and driven by market forces (Harleen et al., 2014). The emergence of ASD (agile system development) came about as a reaction to the incapacity of previous plan-driven methods to address the swiftly changing technological environment (Zaigham, 2012). In relation to this idea, Wingo and Tanik (2015) say that software development for complex problem domains suffer from volatility and unpredictability which places high demands on the software development, these forces can have an effect on the software product itself which was the focus of their study. They found the agile methods, in particular Extreme Programming (XP) can embrace change and can affect the design in such a way that it meets the needs of software solutions for complex problems (Wingo and Tanik, 2015).

Additionally, agility is the capacity to have intellectual response to business forecast with the intent of remaining aggressive and inventive for any unstable and swiftly shifting business environments (Zaigham, 2012). According to Liza, et al. (2014), agile teams comprise of multi-skilled personalities, with on-site customers possessing credible domain knowledge to aid them in proper understanding of the requirements. Therefore, we can derive a proper description of agile methodologies as a set of software development procedures that take several cycles to complete (iterative), incremental (not deliver the entire product at once), and emergent (principles, processes, and structures of work are realized during the project period rather than predetermined).

Most agile development methods endorse collaboration, teamwork, development, and process adaptability during the entire lifecycle of the project. According to Pikkarainen et al. (2008), the best known agile methods are Extreme Programming (XP) and Scrum. Both XP and Scrum have their own unique way of being applied to projects, however, in reference to their differences; XP is concerned with engineering practice and coding and Scrum focusses on software project management and the team who are responsible for the Product Backlog (Almseidin et al. 2015).

Kanban (a Japanese word which means "Signboard" or "Visual Card") is a technique developed in Japan by Toyota Motor Corporation (Wagener et al., 2012). A Kanban schedule system utilises visual use and informs what to produce, how much, and when to produce it. Ikonen et al. (2010) examine Kanban-driven software development and found that Kanban is effective for organising and visualising work, however, it does not prevent waste from creeping in.

The limitation of XP is that its more focused on coding and is not compatible with additional roles, this is not suitable for the present study as it is concerned with the introduction and effect of a new role to an agile team. Scrum, on the other hand, is a more suitable framework for the inclusion of a new role because it is already a role based framework, including existing roles such as Product Owner and Scrum Master. Other disadvantages of XP include that there is a lot of customer involvement, it does not specify the artefacts and does not support many tools (Al Saleem and Ullah, 2015).

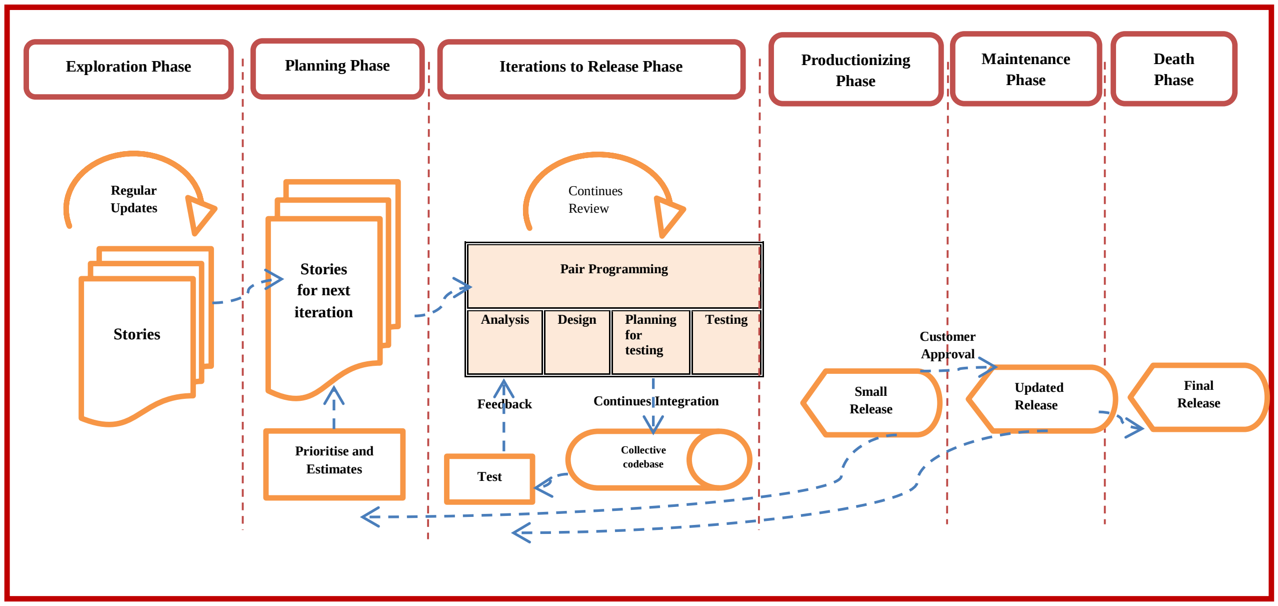


Figure 2.2: The lifecycle of XP process (Al Saleem and Ullah, 2015)

Scrum development framework as chosen method in the present study is discussed in section 2.8.

2.6.1 Success Factors in Agile Development

There are four main success factors in Agile development, namely; the team, the customer, the organisation and communication, as related factors it is important to remember that team autonomy and management support, and importantly to the present study - team characteristics are also important (Hummel and Epp, 2015). However, the limitation in Hummel and Epp's (2015) study is that the team component as a success factor provides little detail other than to say that it is the characteristics of a team that are required to complete a task and also the emphasis on team autonomy. The idea of these four success factors is supported by Rao et al. (2011) who say the reasons for adopting Agile methods include: a) Agile can adapt because it acknowledges that customer requirements will change. The current research aims to improve Scrum agile method which addresses changing requirements and associated security considerations. b) Small iterations can respond to the short time frames demanded by the market. For the present study, each sprint lasts between two to four weeks as a small iteration. c) Short time frames between releases allows quick feedback from customers and d) because there is continuous testing and integration this delivers bug-free software which is required by organisational processes (Rao et al., 2011). This motivates the work of including a new role (SO) who will help the agile team in continuous testing and integration regarding security requirements. Rao et al. (2011) study found that feedback, learning and teamwork (see section 2.6.2) as well as employees being empowered and improved communication and social interactions are important advantages of Agile methods.

Success factors for Scrum in reference to the team include that if the team is large it should have less autonomy, the need for discipline in order to make meetings short and focused within a short time frame and to ensure rules are followed, and finally; that the team is motivated (Hummel and Epp, 2015). In relation to this, the Security Owner will have a number of functions which include improving communication, coordination and teamworking in addition to improving management of security requirements in Scrum.

Turk et al. (2002) addressed the limitations of Agile software processes in organisations' development environments and say that management is finding difficulty in assessing the appropriateness of such processes. Turk et al. (2002) specifically identify the limitations of Agile processes in terms of the types of project they are suitable for. One stated limitation is the lack of support for software develop using large teams and the reason for this is that there is a need for communication, control and coordination. If the teams are too large then there will be too many lines of communication which can reduce effectiveness, especially in terms of face-to-face communication which is a critical part of the effectiveness of agile methods. However, there are situations where large Scrum teams are required and a proposed solution has been the Scrum of Scrums solution whereby the Scrum team is represented in hierarchical order in order to optimise communication (Al Qurashi and Qureshi, 2014), a solution to the aforementioned problems associated with communication.

2.6.2 Teamwork in Agile

In spite of the fact that teamwork and communication are important advantages of Agile methods, as mentioned above, Misra et al. (2011) look at the evolution, principles and criticisms of Agile software development and provide an overview of the criticisms of Agile software development from the literature. Many of the criticisms can be linked to issues that are related to the team, these include difficulty in getting the most suitable people involved (Irons, 2006), limited support for development using large teams (Mahanti, 2006), the need for cultural change for adoption in projects (Stephens and Rosenburg, 2003), the need for the involvement of senior development team members (Stephens and Rosenburg, 2003) and the need for explicitly stated performance requirements (Lindvall et al., 2002).

There are issues related to the agile team itself that have a bearing on the integration of security in agile methods. Firstly, the issue of larger teams, according to Williams and Cockburn (2003), “Agile development is more difficult for larger teams…as size grows coordinating interfaces become a dominant issue,” (pp.39). Secondly, according to Silva da Silva (2011), agile characterised with face-to-face interactions breaks down and brings forth complexity and becomes difficult where more than 20 developers are involved.

Romano and da Silva (2015) in a study that examines project management using Scrum found that team members felt that communication was significantly improved and that members worked together to achieve a goal, moreover, the team members felt that they became stronger in terms of understanding each other's tasks and how they could help. Moreover, during the use of the Planning Poker it exposed the difficulties of those with less experience which allowed those with more experience to help; overall there was increased team motivation and confidence (Romano and da Silva, 2015). However, there were negative aspects recorded which included the need to improve the levels of internal communication (Romano and da Silva, 2015). Even where a development project is geographically distributed the reduced informal contact and the associated risk of being unaware of tasks is mitigated by Scrum through the use of online meetings (Bannerman et al., 2012).

The authors that are reviewed here offer criticisms related to teamwork in agile (Irons, 2006, Mahanti, 2006, Stephens and Rosenberg, 2003), how agility can be compromised (Williams and Cockburn, 2003, Silva da silva, 2011) and how communication can be affected (Romano and da Silva, 2015). However, they do not propose suggestions on how team related issues can be resolved in agile. The hypotheses of the present study include that the introduction of a Security Owner will improve team working and communication in an agile method. The improvement of team working is intended to improve security management in an agile process, the following section addresses security in agile development.

2.7 Security in Agile Development

Literature (Baca and Carlson, 2011, McGraw 2006) has tried to define security in agile development methods and concluded that a secure product is a product that has the capacity to operate as expected regardless of being under an attack, or even if it fails, it does so gracefully (McGraw 2006). There are diverse perspectives and approaches when it comes to security issues in development of software, more so in agile development methodologies which are addressed in this section. Researchers have argued the presence of disparity between the methodologies projected by agile supporters and the established security methodologies, when argued from an engineering perspective (Beznosov and Kruchten, 2005).

In this work we adopted the view of Baca and Carlson (2011) that there are numerous benefits to agile development which include that development takes place at a faster rate with more agility. However, there are concerns that this quicker development pace means that there is lack of documentation which can lead to less secure software (Baca and Carlson, 2011). Furthermore, research depicts that because agile methodologies primarily focus on less documentation than other processes they deter away from security considerations (Baskerville, 2004). Security specialists offer criticism of agile methods on the grounds of limited inherent security measures to establish secure software because they involve high-speed development that does not react to threats are continuously evolving and becoming increasingly sophisticated and dynamic (Baskerville, 2004). The current advocates of extending agile methodologies have, however, failed to establish why the current process cannot contain security practices as part of the recognised framework and the reasons behind necessity of an extension or amendment in order for this to occur, addressed in this study through the inclusion of a Security Owner role and UMLsec (see Chapter 4).

Kumar et al. (2011) emphasise the advantages of considering security at the modelling level in agile which will ensure improvement of security at the architecture which will in turn improve requirements engineering. Software remains susceptible to security threats, and there appears to be no agreement on how to provide an appropriate level of security through the addition of testing mechanisms that are intended to uncover faults. With reference to the research by Kongsli (2006), testing and authentication tools lacked security reassurances addressed during testing, this was because tests went beyond application level vulnerabilities and were concerned with the exploitation environment instead of the software being written. According to research conducted by Goertzel et al. (2007) into the founding agile principles, they established a negative impact on security. This confirms the urgent need for reconsidering the integration of security requirements in agile.

Moreover, contrary to traditional development methodologies that are characterised by goals such as repeatability, predictability, and optimization; most agile methodologies suffer a lack of reliable safety critical software creation. This means that agile methods may not have the ability to develop software of high integrity and security. Silva da Silva (2011) and Boehm and Turner (2004) equally point out that there are always chances of greater risks to architectural flaws that cannot be easily identified by external users because they lack documentation. However, the present study hypothesises that agile methods can be improved in terms of security consideration through the inclusion of a Security Owner role and UMLsec building on previous efforts that are reviewed in the following section.

2.7.1 Contemporary Efforts to Integrate Security

Beznosov (2003) brought in XSE (eXtreme Security Engineering) which was an appliance of XP practices for security purposes that brought forth the idea of "Good Enough Security" as a structure of security guarantee (Beznosov, 2003). However, there is no indication of who is responsible for the deployment or facilitation of XSE which means there is no clarity or recommendation of who would be sutiable for such a role, either team members or an additional role. This means the approach is deficient in terms of how it is facilitated.

Aydal et al. (2006) approached security within web applications by adjourning security verification until all functionality was accomplished and then only devoted a few iterations to mind some security aspects towards the end of the project. In order to overcome these limitations they outlined system-wide issues of security via use of Extreme Programming to classify and prioritise security techniques and later execute them using iteration of their own at the latter stages of functional development iterations. However, despite this approach providing reasonable benefits, it is not a popular idea to extend security techniques only at the end of the project since there may be a call for significant refactoring, and if this is attained earlier while functionality is being created, changes could easily be integrated into the design and development course which can be further assessed via experimental measurements.

As a solution to this problem, Kongsli (2006) introduces an automated testing device, named Selenium, employed to facilitate development and execution of automated security tests for web applications. This tool provided a basis for the final acceptance test suite and assisted in meeting the application's argument for security guarantee (Kongsli, 2006). However, there is no mention in Kongli's (2006) study about how this testing device is deployed. This criticism is similar to the above criticism of Beznosov (2003), where despite a solution being offered, it is deficient in terms of who would be suitable for its facilitation.

Mead et al. (2008) incorporate security requirements engineering into Dynamic Systems Development Method (DSDM), an agile development framework, because security requirements engineering is more effective when it is part of an organisation's existing development process. Often security requirements in development methods are an afterthought, however, the approach taken by Mead et al. (2008) emphasises the need to implement the security method at the beginning stages of the development life cycle in order to build security into the product from the outset, moreover, their approach incorporates the steps of the security requirements engineering so that changes in requirements or new risks can be responded to. An important point that is relevant to the present study is that it is difficult for organisations to adopt new processes when they are developing new security requirements. This study addresses this issue by demonstrating through the addition of a Security Owner role how the adoption of a new process, specifically UMLsec, can be facilitated.

Similarly, Sani et al. (2013) propose a method for mapping security activities into DSDM phases. The resulting Secure DSDM or SDSDM consists of security related activities which support a more secure DSDM, this was based on the problem that DSDM lacked the inclusion of elements of security in its phases. Specifically, the security activities that were integrated included the identification of security concerns and issues, agreement of a security plan and the creation and review of a secure prototype (Sani et al., 2013). It is proposed in the present study that there needs to be a role to identify and management security considerations for all of the other team members in Scrum, in support of this idea, Sani et al. (2013) say that there is a lack of security know-how from those who are involved in software development, namely; software developers, software engineers and programmers and therefore, there is a need to introduce security techniques to those professionals who are not experts. Although the idea is different to the present study, in that it involves the inclusion of security activities as additional phases as opposed to the introduction of a Security Owner, the principle is the same, in that there is a need to make agile team members more aware of security considerations.

Recognition of the importance of considering security throughout the agile iterations is also emphasised by Raschke et al. (2014) who present a method for evaluating security throughout the development process, importantly there methods does not interfere with the principles of agile methods because it allows for changing customer requirements and time-to-market constraints and it provides early feedback about security issues during development. Similarly, Franqueira et al. (2011) criticise the idea that security is often addressed by isolated security practices which do not provide an overview of the state of the security of the system that is under development. Franqueira et al. (2011) feel that such security practices should be integrated into the short iterations of agile towards updating security requirements, in response to these issues they propose a light weight, agile approach to security management which is integrated as part of development process. However, Franqueira et al. (2011) acknowledge that although their idea may bring benefits it may not be suitable for the development of safety-critical systems. It is one of the objectives of this study that integrated security practices will contributed to the development of safety-critical systems.

Another method to incorporate security into agile methodologies is “misuse stories” (Siponen et al., 2005). Misuse stories describe any unwanted purposes of the system that needed concern as part of security methodology (Siponen et al., 2005). Misuse stories involve multiple incremental iterations and have a fast development pace, and can therefore be integrated into Agile methods. In addition, they are simple and do not impede development, and can be adapted to changing requirements in a business environment (Siponen et al., 2005).

Baca and Carlsson (2011) integrated security engineering processes into agile methods to produce security-enhanced Agile development process. A potential criticism of the integration of these engineering processes is that they are not primarily designed for agile methods, in fact this was recognised as one of the threats to validity by the authors, additionally, there was no mention of a facilitating role to facilitate the security functions. In the present study UMLsec is designed to be integrated into agile methods.

In summary, these ideas have shown efforts and are promising in terms of attempts to introduce security within agile methodologies as a response to the substantial deficiency of guaranteed security needed in development projects. The above critical review shows that many attempts to integrate security into Agile methods lack in their explanation of how such efforts will be implemented or who is responsible for doing so.

The above overview of efforts to integrate security requirements into agile methods has highlighted limitations, these include that despite these efforts security requirements are not often considerd throughout the agile development process. Moreover, where they are considered are integrated throughout the agile development process, there is a clear need for a Security Owner role to facilitate using an appropriate tool, in this case UMLsec.

2.8 Scrum Development Framework

The Scrum framework was chosen for the present study because it is used by organisations who wish to transition to agile development due to the fact that it can respond to requirements change (Eliza et al., 2010, Jakobsen and Sutherland, 2009). Moreover, all processes in Scrum involve communication, interaction, discussion and planning within a team working environment which a Security Owner role is hypothesied to improve in this study.

Scrum is defined as a structure used by cross-functional groups in the development of projects as well as products. Such developments are done in an incremental and iterative manner to ensure efficiency. In a typical Scrum development project a period of approximately four weeks is taken to complete each iteration cycle or "sprint" (Scrum-Software Development, Schwaber and Beedle, 2002). A cross-functional team uses a prioritised list (Product Backlog) to select items and various clients' needs. To ensure efficiency, a cross-functional team sets logical objectives at the beginning of the sprint. The set targets should be mutual, real and accurate. The existing sprint concentrates on clear, established as well as lesser objectives. No change is allowed for the period of the sprint. However, changes can be made in the following sprint. The team examines the sprint's development daily and corrections are made so that the remaining work can be done effectively. Both the stakeholders and the team do the evaluation (review) of the sprint at the end to determine exactly what they have achieved. The various responses are used as ideas and lessons in the following sprint. The focus of the Scrum is to improve the quality of the product at the end of the sprint. Figure 2.3 illustrates the Scrum development framework processes.

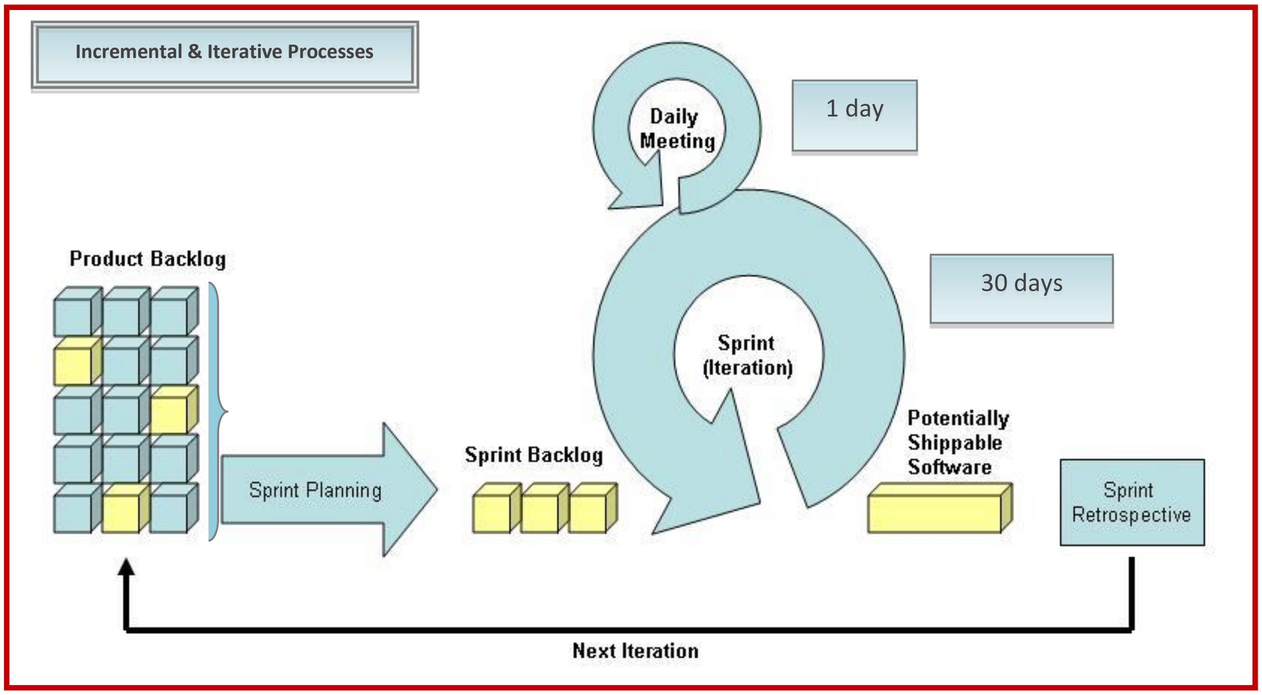


Figure 2.3: Scrum development framework.

2.8.1 Functional Roles in Scrum

Scrum includes three key roles collectively referred to as the "Scrum team". These include the Scrum Master, Product Owner and Development Team (Schwaber and Beedle, 2002, Schwaber, 2004, Matharu et al. 2015). The Scrum Master plays a crucial role in the management of the attained business standards which enables the members of the group to discover and apply the ideas of Scrum throughout the sprint. It is the responsibility of the Scrum Master to assist the Product Owner, the Team as well as the general organisation to achieve the set objectives.

The role of the Product Owner is to capitalise on the return on investment (ROI) through the identification of various product characteristics and creation of a Product Backlog by identifying the specific products that should be included in the next sprint, as well as restructuring this product list. The prioritisation of the Product Backlog by the Product Owner is done based on the risks involved such as the risk which causes vulnerabilities and effects of utilizing system available resources and denial of service. Risk identification and threat analysis is an example of a Scrum team process to prevent/reduce expected risks in order towards a more secure system.

The Development Team plays a crucial role in structuring the product as specified by the Product Owner. This team is independent because it is self-governing in both its administration and organisation but is also accountable. In every sprint, the team embraces required proficiency for the supply of the potentially shippable products. This is the reason why it is termed as cross-functional team. Normally, the Scrum team comprises of between five and seven members (Cohen, et al., 2004). However, the team may include individuals with knowledge in various fields such as architecture, database design, documentation, interface design, analysis, testing and development and security in the case of software products.

The success of a Scrum team is highly dependent on the team members and their performance and although it is accepted that teams should self-manage and be autonomous (Hummel and Epp, 2015, Moe et al., 2010) it has been found in a study to understand team working in an Agile team, specifically Scrum (Moe et al., 2010) that there are problems with team orientation, team leadership and coordination, moreover, there were issues with the division of work which were found to barriers to team effectiveness (Moe et al., 2010). In fact collaboration is a characteristic of Scrum which is based on cross-functional teams whereby individuals who have their own skills contribute to the development of the software product; this cross-functional team may be comprised of software architects, programmers and software analysts (Mathura et al. 2015). It should be noted at this point that this collaboration takes place when the team members come together in daily meetings, daily Scrum meetings, which are notably short, a characteristic that provides its efficiency (Mathura et al. 2015).

Daily Scrum meetings are reviewed in more detail in section 2.8.5. Thus there is a situation where individual roles and areas of expertise come together briefly to discuss product development. This meeting is targeted to generate insight into project process, however, in reference to security, it is difficult to determine who is responsible. If we look at the main roles in Scrum, the Product Owner would be responsible for defining security requirements and the cross-functional development team would be responsible for implementing them. However, existing Scrum roles are not necessarily suitable for identifying, considering and managing security requirements.

Although some of them members may be experts in security, they do not have the authority to make decisions and are focused on the development of functional requirements. Similarly, the

Product Owner may not be knowledgeable of the best security practices and are also focused on functional user requirements. Therefore, a Security Owner role, that understands the best security practices is a solution for the management of security requirements.

2.8.2 Product Backlog

A Product Backlog is a list of all prioritised requirements. Before the first Sprint commences, a Product Backlog is required. A product has only one Product Backlog and this requires the Product Owner to make prioritisation decisions in all the fields to ensure that the interests of the development team and the stakeholders are considered. There are some items within the Product Backlog that require consideration of security and a Security Owner role would assist the Product Owner in considering and prioritising the security requirements in the Product Backlog.

As new information becomes available, all products of the release should be estimated and as everyone learns it is essential that the products should be re-estimated. The Product Owner is then provided with the required estimate for each of the Product Backlog by the development team as well as all the estimated security requirements. It is therefore the work of the Product Owner and other stakeholders to give all the information pertaining product values. Such product value includes returns earned, risks associated with the business including security, reduced costs, stakeholder significance and more developing. The Product Backlog should be refined frequently for purposes of learning and consistency. They can as well add, remove, modify, split or change the product items in each sprint. The Product Owner then constantly updates the Product Backlog so as to meet the customer needs, reflect innovative ideas, competition and other technical difficulties.

2.8.3 Definition of Done (DoD)

The definition should include all the activities required for purposes of shipping the product. All these activities should be done during the sprint. The "Definition of Done" should also include the ultimate Development Team's improvement in the event of trying to deliver a product that is "Potentially Shippable" in every sprint.

The Definition of Done should be agreed between the Development Team and the Product Owner before the setting up of the first sprint. This Definition of Done is a subset of the events that lead to the creation of Potentially Shippable Product Increment. Therefore, the sprint work done by the Development Team will be planned according to the Definition of Done.

2.8.4 Sprint Planning

The team meets to plan for the sprint at the beginning of every sprint. This is referred as the Sprint Planning Meeting and is divided into two different sub-meetings. The first part of sprint planning entails a review of the most important items contained in the Product Backlog that the Product Owner is intending to implement in the following sprint. Both the team and the Product Owner do the review. Such items are highly evaluated in the Product Backlog Refinement in the preceding sprint. This makes the meeting efficient because only a few questions are discussed.

In the first part of the meeting, the team and the Product Owner discuss various objectives and the frameworks for the high-priority items contained in the Product Backlog. This gives the team an intuition of the thoughts of the Product Owner. In addition, this part helps the team to understand the various requirements of the Product Owner as well as reasons why such requirements are essential.

The second part of the Sprint Planning mainly concentrates of the implementation on the Sprint Goal. Here, the team estimates the volume of the items they can probably manage to complete by the close of the sprint. They start with high-priority items from the top of the Product Backlog for the Product Owner then work down the list in order. The team makes decisions regarding the volume of work they can complete rather than the Product Owner assigning the work. Also during this stage in consideration of the items the team has to factor in the security requirements.

2.8.5 Daily Scrum

Daily Scrum is a brief meeting of fifteen minutes taking place at the start of the sprint day at a chosen time. All team members attend this meeting. The team is asked to stand so as to make this meeting brief. Here, the team highlights their various challenges and harmonization of work is done. In the Daily Scrum, all the members of the team highlight three major things. These include; things achieved since the previous meeting; things to be done before the following meeting; and the challenges (impediments) faced in the work. These are the three main questions which each team member asked at the meeting.

The Daily Scrum assists the team in co-ordination because the team members share various things such as achievement and challenges. Notes relating to the impediments are done and the Scrum Master assists the team to solve them. Discussions may be held immediately after the Daily Scrum in case it is necessary. Security is considered by team members for their own particular aspect of the development, however, there is a need for all team members to be aware of all security considerations, this can be achieved through the introduction of a Security Owner role who would highlight security issues during the meeting and offer a progress update on security.

2.8.6 Sprint Review

Sprint Review is whereby the people review the sprint and it comes after the sprint ends. The Product Owner, the Scrum Master and the Team Members should all be present during sprint review. Also the stakeholders which include customers, users, executives, experts and other interested parties can be present. During this time, all the parties present can ask questions as well as give their input. The Review has been taken to be a demo which is not actually the case as the meeting does more than this. Inspection and adaptation are the main ideas in Scrum i.e. observation and learning that will then rely on the feedback in a recurring manner. The product goes through an inspect and adapt process that is represented by the Sprint Review. This is the time the Product Owner learns the progress of the product and of the team (Sprint Review).

During the Sprint Review there is a need to review security entirely. The Review is mainly a comprehensive discussion between the Product Owner, the Scrum team and stakeholders to review the sprint and for the development team to show what they have accomplished, which is assessed against the goal of the sprint. The software increment developed by the team during Sprint is used during the review.

2.8.7 Sprint Retrospective

This is the time for the Team to talk about what is effective and what is not then agree on the changes that should be made. During the Retrospective (Derby And Larsen, 2006), the Scrum Master can act as a facilitator. The Scrum Master should facilitate their own retrospectives for a good idea-exchange among the teams.

It is unfortunate that many teams conduct their retrospectives focusing on the problems and this may mislead people to think that retrospectives as a negative or miserable incident. Therefore, the Retrospective should also focus on the positives and strengths as well as make them more exciting by introducing a variety of techniques over time. At Retrospective meeting, depending on the feedback, Scrum Master and the team focus on how to improve the product increments next sprint.

2.8.8 Scrum Releases

One of the questions that can be asked is exactly how a long-term release plan is done in an iterative model. Here, two cases should be considered which includes; a new product in its initial release; and a current product waiting for future release. The Product Backlog refinement should be done.

Both the Team and the Product Owner should do this refinement before the start of the first sprint where the proper Product Backlog is shaped. Normally, this involves a workshop, which is sometimes referred to as Release Planning or Initial Product Backlog Creation. It takes some few days to complete this. Moreover, it entails estimation and comprehensive analysis of requirements for all the selected items in the initial release.

In Scrum, there is no need for some comprehensive or exceptional release planning for the following release in case of a recognized product with a proven Product Backlog. This is because the refinement of the Product Backlog should be done in every sprint by both the team and the Product Owner. During the release planning there is a need to consider security so that security benefits from the continuous improvement of the product that is afforded by the Scrum release.

The continuous improvement of the product means there is no need for the preparation and execution that are associated with traditional development cycles. The Product Owner and the Team do release planning, estimation, content and priority refinement. This is done in initial Product Backlog refinement workshop and in the regular backlog refinement each sprint.

Release items are items that are expected to be in the present release. In cases where fixed date, deliverable or price is necessary, some of the parameters should be incorporated buffer to give room for uncertainty as well as change. This happens in contract development and therefore Scrum is dissimilar from other approaches.

2.8.9 Scrum Challenges

A Scrum is both a transparent framework as well as substantial set of practices and an inspection and adaption mechanism. Scrum is important in the detection of impediments that affect the effectiveness of the team and Product Owner in order to rectify in time (Deemer et al., 2012). For instance, the Product Owner may not be well conversant with the market and its features and the evaluation of its business value. Moreover, the team may be less knowledgeable in effort estimation or development work and therefore these weaknesses will easily be exposed by the Scrum framework, if security requirements are not considered properly this could affect the effort estimation. Although Scrum resolves the problems of development and presents a framework, exploring ways to resolve these problems in short cycles it does not consider the security in the process of the Scrum development framework. This can occur due to poor skills in task analysis and estimation and an improvement in the consideration of security requirements can improve task estimation. This challenge is very essential to the team as it makes them more realistic and thoughtful about its predictions. The Scrum therefore helps the team identify any dysfunction and allowing them to do something about it. This is a mechanism that the team employing the Scrum experience gets the most important benefits.

Non adherence to Scrum rules is one of the most common mistakes that arise when given a challenging Scrum practice. For instance, some teams that do not deliver may decide to extend their Sprint period so that they get enough time and by doing so, they never get to learn how doing a good job with limited time provided estimating and managing it. In this case, to avoid Scrum being transformed into a reflection of their own dysfunction and weaknesses, there should be enough support and coaching from an experienced Scrum Master. The real benefits offered by Scrum will therefore not undermine and thus making visible the good and the bad and offering the organisation an opportunity to rise to a higher level.

Other challenges that one should be cautious about is the issue of Scrum being imposed to teams by respective managers. Managers should in turn give a team instruments and space to manage itself as this is what Scrum is all about. Successes will not be achieved if the above factors are dictated. The first Sprint is often very challenging to the team but it is very encouraging to note that Scrum benefits will be seen towards the end.

2.9 Chapter Summary

This chapter has provided a background and critical evaluation of the relevant areas to this study. It has not only addressed areas such as security in software and agile as one such approach to developing software but has also brought attention to the issues in integrating the two. Moreover, the chapter has presented an understanding of the internal mechanisms of agile development and its associated success and failure factors, there was particular reference to the team members in agile and how they interact and communicate towards successful use of agile methods. Furthermore, there was a focus on those who have made efforts to consider and integrate security into agile methods. These aforementioned issues were invaluable in a study that aims to introduce a new role in to an agile methodology, namely; Scrum and pursues a novel way of integrating security into agile through the use of UMLsec.

The chapter has served to show the gaps in the research and the state of the art challenges with agile development in terms of attempts at further integrating security and the functional roles in a Scrum team. Thus, the objectives of the review are to provide a critical review of the efforts considering and integrating security into agile methods and reviewing teamworking and the Scrum development framework. In reference to the aims of the study, these issues that have been identified in the literature justify the need to extend and improve agile methods in terms of security requirements and how they are managed by the development team. The key findings are that although there have been attempts to further integrate security and an emphasis on its importance, there is a lack of practical facilitation of these approaches, and this coupled with another key area in the literature about the function and importance of team roles in agile methods contributes to justifying the need to enhance security management facilitated by the addition of a security role.

CHAPTER THREE

RESEARCH METHODOLOGY

Objectives

- Reviewing and comparing the appropriate research methods.
- Conceptualising the research design.
- Presenting the data collection methods used.

3. Research Methodology

3.1 Introduction

This chapter reviews research methods and presents the chosen methods for this study including their development and application. Specifically, their benefits, disadvantages and the rationale for the adopted methods. Empirical research in software development and their guidelines as proposed by Basili (1996) and Kitchenham et al. (2002) are discussed in detail as it forms the key method used in this study. This chapter details the research design, the questionnaire development, sampling and the experiment, as well as a justification for using real experimentation. The results will be discussed in chapter six to evaluate the proposed framework of the study (chapter 4).

3.2 Philosophical Perspectives

Realism refers to an approach that involves recording and reflecting different experiences and is mainly applied in problems related to real life that has a factual meaning.The philosophical position of realism is that reality is independent of the researcher’s mind, in other words there is an external reality consists of things that although are born in the mind, they exist independently of any single person (Sobh and Perry, 2006). This external reality consists of structures that are interrelated, and mechanisms through which these structures interact (Sobh and Perry, 2006). When applying this concept, different human interpretations such as emotional values are not involved; the idea is to exclude emotions such as idealism, feelings, social and other attitudes. This is therefore a way of offering facts and figures in a way that represents different events with reference to a real life situation.

In management sciences, realism is used to show that an organisation is a body with its own cultural and social values. A research based on this methodology will assume that all findings acquired by the researcher will engulf basic social and cultural issues related to an organisation. This type of research often uses deduction in that hypotheses will be established and then verified. In the case of the present study the hypothesis is that "the inclusion of UMLsec and the involvement of a Security Owner will improve security requirements modelling in agile methods". Considering this view, the research in the present study uses experimentation towards verifying the hypotheses. Therefore, realism was considered in this study.

3.3 Empirical Research in Software Development

Basili (2006a) is concerned about the fact that empiricism is considered to be a basic part of scientific and engineering discplines but has not been part of the tradition of software engineering. Basili (1996) says that software engineering should follow other physical sciences in terms of experimentation, this is based on experimentation and process improvement. Basili's approach to experimentation in software engineering is based on the idea that in any discipline progress is based on an understanding of the basic units that are used to solve a problem and requires models to be built in the application domain and an understanding of the techniques that are available for using these models to help solve problems (Basili, 1996). Likewise, Basili and Zelkowitz (2008) apply an empirical approach in order to understand important issues in software development which requires observation, model building and experimentation. In reference to these approaches, the present study considered observation, however, it was deemed inappropriate because observing participants would affect of the results of the study, especially where interactions between participants are important and are studied. Thus it was deemed appropriate to adopt real experimentation.

Basili’s contribution to the empirical study of software development and software quality have included the Goal Question Metric (GQM) approach which has made measuring software possible, specifically, the approach involves deriving measures from goals, limiting data collection to what is required to answer questions, clearly stating assumptions and using appropriate models for interpreting results (Shull et al. , 2006).

The idea that the concepts and practices of software engineering which are taken from experience and observation should have empirical validation is also supported by Kitchenham and Budgen (2002) who state that validation of such ideas would help to link theory with practices. Kitchenham and Budgen (2002) also say that there is a need for empiricism, especially in academic study of software engineering projects. There are practices in software and software development processes that have received little attention which include quality, design patterns and the usability of UML, all of which are relevant to the present study because it will consider the usability of UMLsec and in reference to quality would consider security requirements.

Basili’s idea that empiricism should play a role towards the improvement of product quality and productivity in software engineering has been demonstrated in a study named ‘*Using Measures and Risk Indicators for Early Insight into Software Product Characteristics such as Software*

Safety' (Basili, 2008). This study looked at the relationship between the processes of software development and the resulting product characteristics and was based on the assumption that there is a relationship between processes and product characteristics. Moreover, Basili (2008) recognised that although the characteristics of a system cannot be verified during the development phase, measures for the potential characteristics can be made during development. However, the present study is not concerned with measuring the security characteristics of the developed product, it is primarily concerned with improving the development process in relation to security requirements and evaluating such improvement through the perceptions of those who are engaged in product development. Improving security requirements consideration and modelling by the development team, coordination and teamwork are hypothesised to be improved by the introduction of a Security Owner role and UMLsec.

In software engineering there is a need to model different product characteristics, such as reliability, portability and efficiency, and project characteristics, something which a concern of the present study, which include schedule and cost. Basili (1996) emphasises that it is important to understand the relationships between the process characteristics and the product characteristics. More specifically, there needs to be an investigation of what type algorithms produce efficient solutions in relation to certain variables and how certain development processes produce certain product characteristics under different conditions (Basili, 1996). The present study is concerned with the development process and its impact on the product characteristics, specifically in relation to product security. Therefore, these considerations made by Basili (1996) are relevant to this study.

This study uses some of the ideas posed by Basili to understand how the participants feel that the development process has an impact on the characteristic of the product. Specifically, how product security is impacted. By introducing new process factors can improve product quality in terms of security. This is determined from the opinions of the development team.

Basili (2006b) describes that in software engineering research there has been a move from studies that are based solely on quantitative approaches to experiments which include pre-experimental designs and case studies, the influence and importance of context variables and the domain in the interpretation of results has been recognised. In this study experimentation will be used to investigate the effect of variables, in this case the inclusion of UMLsec and a Security Owner role on agile development in terms of security requirements consideration. The consideration of the influence and importance of context variables has justified the use of real experimentation.

Kitchenham et al. (2002) is more relevant as it offers empirical software engineering research guidelines that will help to improve both the research and reporting. The guidelines are used in this research to improve research planning, implementation and reporting. The guidelines for experimental design included consideration of the population being studied, this was achieved through a sampling technique which included a pre-experiment questionnaire to identify suitable participants based on pre-established criteria. There is a requirement for the justification of the sampling technique, the selected technique was purposive sampling and was justified (Nachmias, 1981, Bryman, 2001, Denscombe, 2003, Vaus, 2004) in light of the fact that participants had to have a certain level of knowledge and experience in UML and agile methods. The guidelines for conducting the experiment and data collection include the defining of software measures. However, this study is not concerned with measuring the software product but instead is concerned with measuring developers' perception and feedback of the extended and improved software development process. The guidelines ensure that if the measures are subjective, in the case of this study the subjective opinions of the participants, then the measurement used should be accurate and not be subject to the bias of the researcher. To follow this guideline Likert scales are employed to increase validity and reliability of results, the present study employs a Likert scale in the post-experiment questionnaire.

Experimentation is used to observe the phenomena and an empirical approach is used to validate the success or otherwise of the framework presented in chapter four. The experiments will be concerned with the structure of the Scrum team and the interactions and communications between its members. Related work by Nagappan, Murphy and Basili (2008) conducted an empirical case study about the influence of organisational structure on software quality which will be used to help guide the methodology in this study. Although this study is not concerned with software quality it is concerned with organisational issues such as communication and interaction between the various Scrum team members and how they are impacted by UMLsec and the Security Owner role and how this affects security requirments modelling.

3.4 Quantitative and Qualitative Research

Research methods can be categorised as quantitative and qualitative (Myers, 1997). All such quantitative or qualitative studies rely on suitable methods and their validity for deriving valid findings. The present study uses both quantitative and qualitative data, mixing these two types of data is commonly referred to as a mixed methods approach and is described in the following.

With quantitative research collected data can be applied for statistical analysis (Wilson, 2001). Statistical analysis will allow the investigator to find out whether the derived data can answer the research question.

This study presents hypotheses about the effects of UMLsec and a Security Owner role on security requirements modelling in agile methods. Therefore, it is appropriate to adopt a quantitative approach through the use of a questionnaire which contains statements on a Likert scale, in order to test these hypotheses.

Qualitative research is broadly applied in social studies and can assist to accumulate the proofs through observation, interviews and open-ended questionnaires (Wilson, 2001). Although the questionnaire used in the present study has closed-ended questions on a Likert scale, at the end of the questionnaire there are open-ended questions which for the qualitative part of the questionnaire. Qualitative methods will be used in this study to give the participants the opportunity to elaborate on the subject in hand. According to Bryman and Bell (2007) qualitative methods are used in research to evaluate mainly the social environment, in the case of the present study people interacting within and engaging in an agile development process.

One key characteristic of a qualitative research study is the ability to recognise various phenomena especially those that exist naturally. Data is one element of a qualitative research study that occurs naturally. According to Silverman (2005) this data helps in linking the "how" and the "what" in a chronological manner. The strength of qualitative study is its ability in recognising the phenomena that occur naturally. In qualitative research, the data is revealed naturally and links sequentially the 'how' and the 'what' the participant means (Silverman, 2005).

3.5 Research Design

In this section the development of the research design is presented, motivating for the adoption of qualitative and quantitative approaches. The research relates to software engineering and adopts some of the empirical research principles put forward by Basili (1996) and Kitchenham et al. (2002) who advocated that research in software engineering should include empiricism and follow other physical sciences in terms of experimentation and process improvement (Basili, 1996). The present study involves understanding issues in software development which requires experimentation through an empirical approach (Basili and Zelkowitz, 2008).

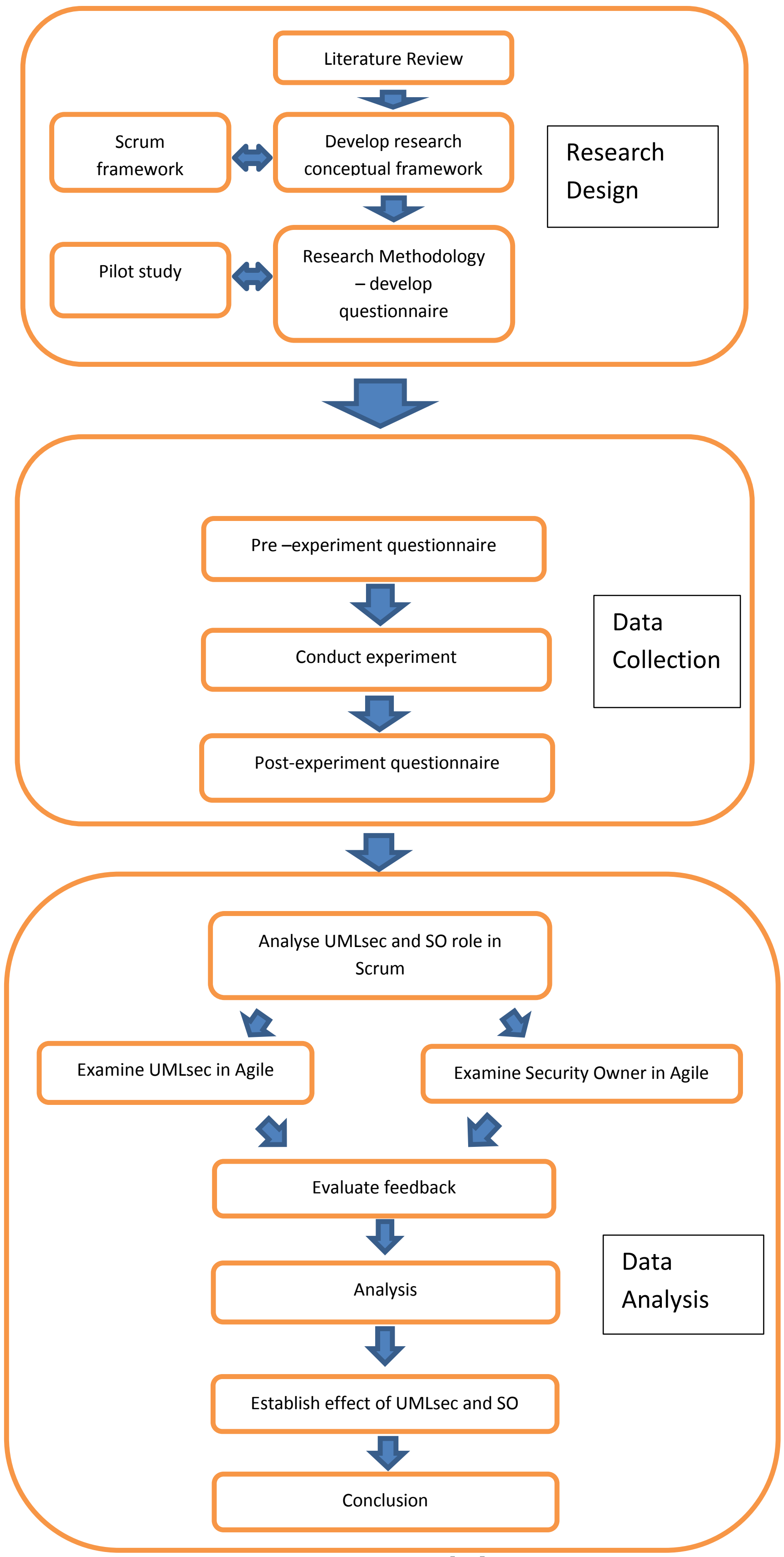

Literature Review
Scrum framework
Develop research conceptual framework
Research Design
Pilot study
Research Methodology – develop questionnaire
Pre –experiment questionnaire
Conduct experiment
Post-experiment questionnaire
Data Collection
Analyse UMLsec and SO role in Scrum
Examine UMLsec in Agile
Examine Security Owner in Agile
Evaluate feedback
Analysis
Establish effect of UMLsec and SO
Conclusion
Data Analysis

Figure 3.1: Research design

Figure 3.1 provides an illustrated overview of the reseaech design for the study. The research design includes a development of the conceptual from which the research methodology was developed. Data collection methods of the research methodology include a pre-experiment questionnaire as part of identifying suitable candidates, the experimentation itself and then the post-experiment questionnaire to derive opinions towards understanding the effectiveness of the proposed approach of the study. These derived opinions were analysed towards confirming hypotheses about the inclusion of UMLsec and a Security Owner role.

3.6 Research Methods

There are varying types of software development research methods; therefore, researchers can apply one or more methods to fulfil their research requirements. Myers (1997) stated that better confidence in the consequences could be achieved by utilising multiple methods, as in this work. There are several factors that play a role in selecting a single or multiple techniques, for example the research queries, the area of research, the researcher's context and the envisioned spectators (Palvia et al., 2003). This research goal is to assess the perception of Scrum team members of the implementation of UMLsec and Security Owner role in agile, hence the researcher selected questionnaires and experimentation as research approaches. The experimentation was used in order to introduce UMLsec and a Security Owner role to the Scrum development framework, and the questionnaire was used to determine if the introduction of these factors would have an impact on the development process from the perspective of the Scrum team members.

3.6.1 Interviews

Consideration was given to the different potential research methods that can be used in this study to derive the perspectives of the participants about the use of UMLsec and a Security Owner role for modelling security requirements in Scrum. Interviews are known for gaining insight into a situation and are also flexible in nature because they are about asking questions listening for the answers. Moreover, they allow for structure towards the purpose of the study while at the same time allow for the researcher to further clarify or probe into points that are raised (Gillham, 2005). Overall, interviews give a clearer picture of how things actually are, however, given that there are more than one hundred participants there are practical implications in this study. The first practical implication is that people are often busy so they do not have available time (Gillham, 2005) for an interview and the fact that there are more than one hundred participants make interviews impractical in terms of time not only for the participants but also for the researcher as well. Interviewing so many participants is not only impractical in terms of time, but also it would be too costly and is not justifiable for research that is medium scale and being carried out by one person. Therefore, the time, cost and practical implications make interviews an unrealistic expectation in this case despite the benefits.

3.6.2 Focus Groups

One of the features of a focus group interview or focus group discussion is that it relies on generating and analysing the interaction between the participants (Barbour, 2008). In fact it is important that the moderator should moderate the discussion while at the same time should pay attention to group interaction and group dynamics, while these may be an advantage to the present study in that it may reveal a concensus of experiences and opinions or may offer an explanation for phenomena (Barbour, 2008). The disadvantages of focus group interview include that one or two participants could dominate the session, participants may not want to reveal opinions that are of a sensitive nature in front of other people and the environment of the focus group would be different to that of the experiment which may influence responses. Like with interviews described in the above, there are practical difficulties in using focus groups for this study. This study involves many participants in numerous experiments which make focus groups impractical.

3.6.3 Questionnaires

Questionnaires (surveys) are utilised for reflecting perspectives of the real situation at a specific point (Galliers, 1992) and they can help to give a variety of perspectives to the real world. This method is mainly used when phenomena are investigated in their natural background (Pinsonneault and Kraemer, 1993). Moreover, surveys can generate large quantities of data in a brief time for a fairly reasonable cost. One weakness of surveys is the probability of predisposition on the part of the respondents or the researcher, or arising from the time at which the research is commenced. Robson (2002) shows that the investigation is a research plan, i.e. a general approach to undertaking research rather than the tactic or a precise approach. Cohen et al. (2000) mentioned that the questionnaires could be applied to define points of view regarding concepts, activities, earlier experience and prospective plans. Moreover, questionnaires also assists to cover a large number of people and gains straightforward information which reveals the viewpoint of the sample of the research (Creswell, 2003, Remenyi et al., 1998). Social researchers also use questionnaires to examine the subjects from a certain perspective, in this case the perspective of Scrum team members. However, "questionnaires are not among the most prominent methods in qualitative research, because they commonly require subjects to respond to a stimulus, and thus they are not acting naturally" (Woods, 2006, p. 15). Nonetheless, questionnaires are a useful way of collecting information from a large sample that cannot be achieved through interviews. This study aims to involve over 100 participants and interviews would not be practical (see section 3.6.3.2). Woods (2006 p.15 - 16) say that "where certain clearly defined facts or opinions have been identified by more qualitative methods, a questionnaire can explore how generally these apply, if that is a matter of interest. Ideally, there would then be a qualitative 'check' on a sample of questionnaire replies to see if respondents were interpreting questions in the way intended".

Surveys are often referred to as sample surveys and the collected data is analysed using some measurement technique. Questions in questionnaires may be closed or open ended and are asked of all participants within a sample group. The aim is to gain data to test pre-determined hypothesed, in this thesis there are six hypotheses to be tested.

Questionnaire are considered an economical way to gather data directed by a single investigator (Bryman, 2001), this benefit is important for this analysis because other techniques such as interviews are expensive and take time. However, this method also has weak points. A questionnaire necessitates simple, easily assumed questions and directions; a questionnaire becomes unable to help the researchers to get further information or to clarify responses; although this is the case in this study, participants are given the opportunity to say how they feel through open-ended questions at the end of the questionnaire, moreover the researcher becomes unable to control the rates of the response (Nachmias, 1981), (Bryman, 2001).

Within the agile techniques, this research utilised a questionnaire to obtain data about the effect of modelling security requirements of UMLsec with a Security Owner role in an agile method. Both Remenyi et al. (1998) and Palvia et al. (2003) stated that questionnaire could assist to capture information from the individuals that is hard to observe. Additionally, questionnaire can be applied to hold the points of view regarding notions, past experiences, upcoming plans and actions.

3.6.3.1 Questionnaire Development

The questionnaires adopted in this study were developed in order to verify the established hypotheses of this study. There were two questionnaires adopted in this study, the first, a pre-experiment questionnaire, was aimed at identifying suitable candidates for the experiment and identifying appropriate roles of these candidates in the simulated scenario, the second, a post-experiment questionnaire was aimed to elicit the perceptions and attitudes related to the inclusion of a Security Owner role and UMLsec in the Scrum development framework, this questionnaire was designed to elicit both qualitiative and quantitative data.

3.6.3.2 Sampling for Questionnaires

In the fields of security and information systems it is important to confirm the population, sample frame, and the sample organisations. The population could be recognized as “the globe of units from which the sample could be selected” (Bryman, 2001, p. 85) in addition with “the cumulative of all cases that follow to some selected set of conditions” (Nachmias, 1981). The sample is “the section of the population that is designated for examination" (Bryman, 2001, p. 85). The frame of the sample is "a list of objectives of 'the population' from which the investigator can make his or her collections" (Denscombe, 2003) and is "the list of all the units in the population from which sample will be selected" (Bryman, 2001, p. 85).

The sampling frame ought to be produced from more than one source which adapts to stipulations and restrictions of the investigation (Denscombe, 2003). Adequate numbers must be cautiously chosen and signify the population. Samples between 30 and 250 cases can utilised with the questionnaire in social investigation (Denscombe, 2003). Furthermore, the capacity of the sample to represent the population ought to be considered, and this capacity is not associated with the size of the sample but must be reliant on accurateness. Despite the intended sample size being more than 100 participants, accurateness in terms of this sample's capacity to represent the population was further increased by the use of a pre-experiment questionnaire design to identify suitable participants for the experiment.

The literature indicates that there are several types of sampling methods, but all come under one of the two main groups: probability sampling, which is frequently known as random sampling, or the non-probability sampling. Probability sampling is created on a coincidental selection processes. The population in this method is recognized as a non-zero probability of being designated (Nachmias, 1981, Bryman, 2001, Denscombe, 2003). The benefits of this method are the decreasing number of errors and removing preference in the sample (Bryman, 2001). Probability can be distributed into four key types: such as simple random sampling, systematic sampling, stratified sampling and cluster sampling (Nachmias, 1981, Bryman, 2001, Denscombe, 2003, Vaus, 2004). Systematic sampling has the similar principles of simple random sampling with the systematic selection of people or the events.

According to the stratified sampling methods, splitting the population into subgroups (groups or strata) and then taking a simple random sample in each subgroup into consideration. Cluster sampling signifies the researcher chooses from the groups (clusters) already persisting in the target population. Based on the individual judgment of the researcher non-probability sampling is measured. Though the probability insertion for each of the members is unknown (Bryman, 2001, Gilbert, 2004), in this method few people have a larger probability of being incorporated into the sample. The benefits of this method have encouraged researchers to choose it in place of probability sampling.

Non probability samples could help in terms of the expediency and economics, when the probability samples are costly and time consuming (Nachmias, 1981). Researchers select non-probability sampling just because of the population (Nachmias , 1981, Vaus, 2004). Non-probability sampling is split into four categories: snowball sampling, convenience sampling, quota sampling, and purposive sampling (Nachmias, 1981, Bryman, 2001, Denscombe, 2003, Vaus, 2004). The convenience sampling method comprised of a population who are simply situated and ready to take part.

Quota sampling is like stratified sampling, where the population is split into categories according to some fixed proportion. The Snowball sampling method permits the researcher to link to a small group of people who are relevant in the study and that also permit the researcher to build contacts with the others. Purposive sampling is the method where the researcher chooses the sample with a particular purpose in mind. Since the accurateness of a sample relies mostly on the sampling structure, the researcher must make sure that there is a high level of correspondence between both the sampling frame and the whole population.

Purposive sampling was chosen for the study because it is a sampling technique where participants can be deliberately chosen based on their qualities (in this case those with experience of security and software development), the researcher makes the decision of what needs to be known (the effect of the inclusion of UMLsec and a Security Owner on security in Scrum) and selects participants who are able to provide that information based on their experience or knowledge (Tongco, 2007). The study adopts purposive sampling because the research is not interested in what most people in a population think about a particular issue but is interested in how a particular type of people (those who engage in agile software development) how they these attitudes are constructed and the role they play in an organisation or group (Palys, 2008).

In reference to the total number of participants in the experiment, because the experiment reflected a real Scrum development situation it was necessary that the number of participants for each experiment reflected this. Therefore, between five and ten participants were included in each experiment. The pre-experiment questionnaire will identify suitable participants, secondly, to gain background information such as length of experience for analysis purposes. The researcher had to consider participants interested in security, UML modelling, and agile development processes rather than security engineering in general. In light of this need to select participants with specialist knowledge of the subject being researched and based on the aforementioned criteria of selection, it was appropriate to adopt purposive sampling as mentioned in the above. The post-experiment questionnaire will be distributed to all the participants who took part in the experimentation.

3.6.3.3 Pre-experiment Questionnaire

The researcher contacted students from Information Systems and Software Engineering departments at three universities in Saudi Arabia where the researcher had ease of access. These universities were Imam Muhammad Ibn Saud University in Riyadh, Shaqra University in Riyadh and King Abdul Aziz University in Jeddah. The researcher explained the nature of the study and invited students to complete the pre-experiment questionnaire. The pre-experiment questionnaire was designed to identify students who have knowledge or experience of agile development methods and UML modelling.

3.6.3.3.1 Sampling for Pre-experiment Questionnaire

In order to test the hypotheses of this study it is important that the sample is drawn from a well-defined population and therefore representative of that sample (Easterbrook et al. 2007). Not only does the researcher have to ensure a sufficient number of responses to the pre-experiment questionnaire based on a response rate of 50 percent, but the researcher also had to consider that only a number of those who do respond will meet the criteria to participate in the experiment. Given the researcher wanted a minimum of 100 participants for the experiment 300 questionnaires should be distributed, assuming the 50 percent rate 150 would return the questionnaires of which at least 100 would fit the criteria. The assumption that a majority of those sample will fit the criteria is based on the fact that questionnaire were distributed to students from IT departments who should meet the minimum criteria.

The number of participants that received the initial pre-experiment questionnaire exceeded the 100 figure that is required for the experimentation, the reason for this is that not all recipients will respond, and even where they do respond to the sampling questionnaire, they may not proceed to the training or simulated scenario experiment.

3.6.3.3.2 Access and Permission – Pre-experiment Questionnaire

These people first had to be identified; this was achieved through liaising with a contact, in this case the head of department at each university, moreover, the contact forwarded emails to the prospective participants. Informed consent was achieved (see section 3.9).

3.6.3.3.3 Participant Criteria for Experimentation

In response to the pre-experiment questionnaire if a respondent meets a number of criteria they will be deemed suitable to proceed to the scenario training and the scenario experimentation. The following are the criteria for the sample:

1. Knowledge about security in software development
2. Knowledge about agile development methods
3. Knowledge of modelling using UML
4. Experience or knowledge using Scrum

The pre-experiment questionnaire can be found in appendix and contains six questions related to the abovementioned areas, if they answer 'no' to questions 2, 4 and 5 they will still be consider for participation in experimentation.

3.6.3.4 Post Experiment Questionnaire

For the post-experiment questionnaire, evaluating the agile process with UMLsec and the Security Owner role was aimed to elicit the respondents level of agreement with statements that would the hypotheses. Questions were designed to gain the respondents' degree of agreement or disagreement with ideas put forward about the use of UMLsec and Security Owner in agile methods, this formed the quantitative element of the questionnaire, and there are two questions that were open to elicit perceptions of UMLsec and the Security Owner role, this formed the qualitative part of the questionnaire.

A questionnaire is distributed to the participants in order to derive the perceptions of the inclusion of UMLsec and the Security Owner role. The questionnaire is designed reveal the perceived advantages and disadvantages of these inclusions towards verifying the hypotheses of this study. A number of examples are shown here in order to demonstrate how the questions were developed based on the hypotheses. The first hypothesis reads as follows: The introduction of a Security Owner role helps the Scrum team in terms of effectiveness when considering and managing security requirements (when using UMLsec) (H1). The associated statement to support this hypothesis include the following:

You found it easy to identify security requirements (H1, H2)

You found it easy to prioritise security requirements (H1, H4)

You were able to manage security requirements effectively (H1)

Another hypothesis reads as follows: *Scrum as an agile process remains agile when UMLsec is used when / if facilitated by a Security Owner.* In order to confirm this hypothesis the following statement was used in the questionnaire:

The agility of the team was maintained in the agile process (H6)

In order to verify the hypotheses participants were invited to provide their level of agreement or disagreement with these statements. A list of all of the hypotheses are provided in Table 3.1 and the mapping between the questions in the questionnairte and the associated hypotheses are presented in Table 3.2.

Hypothesis no.	**Hypothesis statement**
H1	The introduction of a Security Owner role helps the Scrum team in terms of effectiveness when considering and managing security requirements (when using UMLsec)
H2	The Security Owner role helps to identify security requirements
H3	The Security Owner role will help team members with UMLsec in terms of team working
H4	The Security Owner role will help team members to prioritise security requirements using UMLsec
H5	UMLsec is an appropriate tool for modelling security requirements (when facilitated by a Security Owner)
H6	Scrum as an agile process remains agile when UMLsec is used when / if facilitated by a Security Owner

Table 3.1: Hypotheses Statements

	Question	Hypothesis	Metrics / Measurement
1	You found it easy to identify security requirements	H1, H2	Likert Scale 1 – 5 Strongly disagree – strongly agree
2	You found it easy to prioritise security requirements	H1, H4	
3	You were able to manage security requirements effectively	H1	
4	You found it easy to model security requirements	H5	
5	The methods you used for modelling security requirements were appropriate	H5	
6	You were helped to identify security requirements	H2	
7	You were helped with modelling security requirements	H5	
8	As a team you could effectively implement security into the product	H1	
9	You received effective advice about security requirements to help with security requirements management	H1	
10	You found that you and the team could effectively negotiate security requirements with the Product Owner	H3	
11	There was effective team work regarding security requirements in the Scrum meetings	H3	
12	There was effective communication about security requirements in the Scrum team	H3	
13	The team effectively identified security requirements	H2, H3	
14	The Product Owner effectively prioritised security requirements	H4	
15	Security requirements were effectively managed by the Scrum team	H1	
16	As a team you could manage you own security requirements / considerations without assistance	H1	
17	The agility of the team was maintained in the agile process	H6	
18	You required help with the technique you used for modelling security requirements	H5	
19	In consideration of the approach to security requirements that you engaged with in the experiment, what are your opinions about the security requirements consideration?		Qualitative analysis
20	Please give you overall opinion of the team and how well you work with them and whether you think they were effective in modelling security.		Qualitative analysis

Table 3.2: Mapping between Questionnaire and Hypotheses

3.6.3.4.1 Piloting – Post Experiment Questionnaire

To explore any probable deficiencies before it is conducted, a questionnaire instrument should be pre-tested (Remenyi et al., 1998). A pilot study is a vital step to find out any issues that can hamper the suggested technique of data collection. It can assist to evaluate the research questions or the hypotheses; and may lead to enhance them or create new ones. A pilot test can help the researcher in terms of methods, opinions and signs. It can ensure the words in the questionnaire are comprehensible and clear. The questionnaires in this study were considered in terms of vocabulary, that items were not opaque, biased wordings, the capability for response to questions and their comments and proposals to update the questions. Questionnaires were distributed to ten people which included IT students and staff and because the questionnaires were in English they were asked if all of the words were clear and that the questions made sense, this was especially important as English is not the first language of the participants. The researcher also discussed with staff in order to determine if they had the correct intepration of the questions which was find to be true. Generally, only a few modifications to the questions had to be made which related to simplifying the structure of the sentences. Comments and suggestions from participants in the pilot study were accepted and the questionnaires were modified according to their comments.

3.7 Experiment

The participants are to take part in a scenario experiment where they were engaged in a scenario where they have to resolve an issue within a simulated agile development situation. Participants will be presented with two different scenarios, one will be using agile methods to develop software and will focus on the security aspect of the development, the other will be the same as the first but will include the addition of UMLsec and the Security Owner role, the former being the control experiment. Detail of the experiment is provided in Chapter five.

3.7.1 Justification for Real Experimentation

The experimentation of the Scrum method with and without the inclusion of the Security Owner is something that clearly involves experimentation of the interactions that take place between team members and between team members and the Security Owner. This was a consideration of the approach to the experimentation, specifically, whether to select a simulated online experiment or a real scenario experimentation, the latter was selected and a justification is provided here. This interaction and collaboration that is taking place in a software development scenario includes body language, tone of voice and facial expressions as part of the interaction between team members. Moreover, in reference to the principles of agile development one of the principles is that the most effective and efficient way of relaying information to and within the team is face-to-face interaction. Although these are not measured in the study, they are essential as part of the simulation of a Scrum team development including all its associated interactions and process in a real time, real life situation. The study aims to see the effect of the inclusion of the Security Owner and UMLsec, which are factors that interact in this simulated, real time situation and therefore, a simulation may be more appropriate than an online simulation where real human face-to-face interaction is lost. Here the various issues related to the decision as to whether or not real experimentation instead of online experimentation should be used are addressed.

Dandurand et al. (2008) say that one of the disadvantages of doing online experimentation, as opposed to lab methods, is that there can be a high dropout rate. Nedic et al. (2003) say that although there has been numerous software packages designed for the simulation of real experiments, they are not as effective as undertaking real experimental work and they are a poor replacement for real lab work. Moreover, Nedic et al. (2003) also say that the lab environment provides opportunities for testing conceptual knowledge, working collaboratively and learning by trial and error, an another interesting issue raised by Nedic et al. (2003) which would be relevant to the present study is that in real experimentation there are time and physical boundaries, the present study aims to introduce a new role that will interact with existing Scrum team members and that this human interaction will take place in an agile development framework that has time and physical boundaries.

One of the main disadvantages of web-experimentation is that there is a reduction in the experimental control, because all of the participants are at their own location there might be environmental noise or distractions and we do not know the characteristics of the participants or their psychological states of mid (Pollanen, 2014). In the lab experiment situation the conditions can be controlled and the lab environment can be polished, scrutinised and tested.

3.7.2 Sampling for Experiment

It is preferable that the number of members in an agile team does not exceed 12; the minimum for the experimentation is five members. In light of this the number of participants ranged between 5 and 11 for the experiments. Moreover, it was necessary to have approximately half of the experiments without the inclusion of the Security Owner and UMLsec and half the experiments with the inclusion of the Security Owner and UMLsec. Chapter 5 provides more details of the actual sample numbers for each experiment.

The identified participants completed the questionnaire, the responses to which will identify if they are suitable for the study, the criteria for selection has been described in section 3.6.3.3.3.

This study is concerned with the perceptions of the team members working in an agile software development method, namely, Scrum using a survey approach. Similar studies have addressed developer perceptions of using agile methods. Mannaro et al. (2004) surveyed developer perceptions of agile methods in software companies, their study adopted a questionnaire distributed to 122 participants. Santos et al. (2011) assess the perception of agile team members about how agile methods contribute to the quality of software, their study used questionnaires with 109 participants. Vijayasarathy and Turk (2012) look at the drivers for the use of agile as a development process from the perspective of the advantages and disadvantages, they used an online survey with 98 participants which they stated was a reasonably good size. Therefore, in consideration of these studies and the fact that real practical experiments were being conducted and not online surveys, that the number of participants for the scenario experiment is 110.

3.7.3 Preparation Training

The training course will prepare the participant for the simulated scenario experimentation. It will familiarise them with software development using agile methods and modelling with UMLsec and explain to them the procedures for the experiment. Details of the experiment training are provided in Chapter five.

3.8 Data Analysis

Data analysis will be conducted in order to test the hypotheses of this study; the hypothesis will be tested using the significance level. To conduct other analysis to derive other findings, for example relationships between different values. To achieve these analyses data will be inputted in to SPSS.

T-test will be conducted using SPSS to check if there is a significant statistical difference between the means in responses between the two different experiment groups. This will be conducted in order to verify the hypotheses of the study.

Analysis of the qualitative will include collecting and organising the data and then carrying out an analysis in order to identify emerging themes; this will involve coding of the data from the responses of qualitative questions. The evaluation of the implementation of UMLsec and the Security Owner role in agile development methods should be based on the aforementioned analysis. Questionnaire data analysis assisted in this assessment. Through evaluating the strong and weak points of the implementation UML in agile this assessment can provide recommendations to development.

3.9 Ethical Considerations

Because the study involved human participants it was necessary to consider ethics and gain ethical approval from the researcher’s university. Ethical approaval was granted. Towards this informed consent was conducted whereby participants were informed about the purpose of the study and their consent was gained. The participants were also informed about their right to withdraw from the study at any time.

3.10 Chapter Summary

This chapter presented the methodology employed in this study. Specifically, there was a review and comparison of the available philosophical and methodological approaches to achieve the aims of the study and a consideration and justification for adopted methods. There was consideration of the different methods that could be employed to achieve the aims of the study, including interviews and focus groups, and while these were shown to have advantages for eliciting perspectives and opinons, the associated disadvantges were discussed including that they were highly impractical given the number of participants. The chapter showed how the selected method, questionnaires, were justified and how they formed part of the overall experimentation research and assisted in deriving data about its success or otherwise in terms of the inclusion of UMLsec and the Security Owner role. The development and administration of the select method was also presented. Additionally, the population and sampling methods were also explained. The core contribution of the present study which is the integration of UMLsec and the Security Owner role within an agile process is presented in the following chapter.

This chapter has presented the methodology to show how it is appropriate for achieving the aims and objectives of the study and specifically, how the development of the research instruments considered the hypotheses that they are developed to verify. An overview of the research design provided a visual illustration of how the adopted methods are applied to achieve the study objectives.

These methods, the questionnaires and the experimentation are applied for analysis in chapter six. The experimentation of the proposed framework is to extend and improve agile process in term of security requirements consideration, modelling and management. This framework will now be presented in the following chapter.

CHAPTER FOUR

FRAMEWORK OF INTEGRATING UMLSEC IN AGILE DEVELOPMENT METHODS

Objectives

- Illustrating proposal of extending Scrum.
- Integration of misuse case analysis and agile security analysis techniques.
- Adopting UMLsec in the extended Scrum framework.

4. Framework of Integrating UMLsec in Agile Development Methods

4.1 Introduction

The literature (chapter two) reviewed some studies and researchers' attempts at integrating security activities with agile software development methods (Martin, 2003), and motivated the need to integrate security into these methods. The following sections describe the core contributions of this thesis which are to integrate UMLsec as a security requirements technique and a Security Owner role to facilitate UMLsec in an agile development method. The discussion concentrates on how to apply UMLsec and its facilitation by the Security Owner in software projects that are controlled by the Scrum management framework.

The Scrum framework is the most popular for organisations that consider transitioning to agile development due to its rapid requirements change (Eliza et al., 2010, Jakobsen and Sutherland, 2009). The Scrum framework was chosen as an agile development method primarily because all processes within the framework involve communication, interaction, discussion and planning within a team working environment and the study aims to assess the impact of a Security Owner role on teamwork (chapter two; sections 2.6.1 and 2.6.2) The backlog refinement in Scrum allows team members to discuss thoughts and concerns as well as understanding workflow. Sprint planning meetings involve a team discussion to prioristise stories and how these stories can be broken down into tasks. The daily Scrum meeting is designed to ensure effective communication and that all members are on the same page and there is an explanation of progress, tasks to be completed and obstacles to process by each team member. The Sprint review meeting involves the team presenting the product to the Product Owner and finally, during there is a Sprint retrospective meeting where the team meet with the Scrum Master about what went well, what went wrong and how improvements can be made in the future. In the following the role of Security Owner is described and linked to the artifacts and processes of the Scrum framework reviewed in chapter two; section 2.8. Therefore, all of these processes involve the human factor, the framework depends entirely on human interaction, so in terms of security and improving security requirements through the introduction of UMLsec this would require a human solution to facilitate security requirements consideration.

One of the objectives of this study is to determine how the Scrum team can use UMLsec to prioritise and apply their security requirements. The Scrum framework defines three main actors/ roles that should be available in Scrum. The Product Owner makes a decision of what the development team will build next, whereas the Scrum Master facilitates the development team's work. The development team are professionals who actually build the product (Schwaber and Beedle, 2002, Schwaber, 2004). Normally, the responsibility of securing the product lies with the development team. The prioritisation and identification of security relevant stories is a negotiation between the Product Owner and the development team. This thesis proposes the addition of a specific Security Owner role who is tasked with helping the Scrum team to apply best security practices and influence the build in order to create a more secure product through the use of UMLsec.

4.2 The Security Owner

The study proposes a Security Owner as a role to help the Scrum team produce more secure increments when they are dealing with security requirements through the inclusion of UMLsec. The Security Owner will help in the identification, consideration and management of security requirements that are related to the system to be developed. In consideration of existing Scrum roles, they are not suitable for the identification, consideration and management of security requirements. The Product Owner may not have knowledge of the best security practices and they are more focused on functional user requirements and ROI. Although some Development Team members may be experts in security, they do not have decision making authority and are primarily concerned with functional requirements development. Finally, as for the Scrum Master, their role as the leader is to assist the team and remove or mitigate obstacles. Therefore, the Security Owner, who understands best security practices, is a suitable solution for the regulation of security requirements.

4.3 The Security Owner's Responsibilities

The role of the Security Owner is facilitating the work of the Scrum team in relation to security requirements using UMLsec in order to produce a more secure product. In reference to the different processes within the Scrum development method, the role of the Security Owner is described in the following:

- *Sprint planning meeting:*

The Security Owner is involved in the sprint planning meeting where the Scrum team together with the Product Owner determine a list of all functional requirements of the product and prioritise them in terms of importance in the Product Backlog. During the sprint planning meeting the Security Owner will be responsible for helping identify and prioritse security requirements to be modelled in the following sprint towards assuring the security of the system. These requirements include security requirements with other requirements in the planning poker game. An important part of the Security Owner’s role is to maintain a balance between the need to implement security and the need to maintain the agility of Scrum which would require managing heavy security requirements that may affect the agile process or the time boxes of Scrum sprints, which could negatively affect delivery releases of the system to the customer which could lead to tension with the Product Owner.

- *During the sprint:*

The role of the Security Owner involves discussion and assistance in the estimation of the approximate time and effort for security requirements, contributing to the implementation of those requirements using UMLsec and providing advice to any members if required.

- *Sprint Retrospective:*

Security Owner as a member of the team, will attend the sprint retrospective meeting to discuss the following issues:

- What went well in terms of security during the sprint? Or what security requirements have been checked and verified?
- Where can improvements be made in relation to security in the following sprint? Or what security requirements need to be checked in future sprints?
- Expected Threats and Vulnerabilities.

- *Considering security in daily Scrum:*

 Due to the extra security tasks discussed by the team with the Security Owner in the Scrum review and Scrum retrospective meetings, they also have to consider them in the following daily Scrum meetings. While Scrum is an agile methodology, it creates collective accountability and continuous progress of these security issues triggered or discussed by the Security Owner with the team each time they meet.

4.4 The Security Owner's Authority

The study is based on the idea of the incorporation of a security philosophy in the Scrum development framework. The table below illustrates the tasks that the Security Owner is authorised or not authorised to do while acting in the Scrum agile software project:

Authorised	**Not authorised**
Identify and specify security related requirements to the proposed system	Prioritise and manage the whole system Product Backlog
Build, present and manage Security Backlog (some security requirements could not be addressed by the Product Owner)	Identify unrelated security issues in a Security Backlog that effect on the system agility
Negotiate with the Product Owner to produce the refined Product Backlog	Producing the final refined Product Backlog alone
Support the development team in security testing	Interference with Scrum Master role

Table 4.1: Authorised and non-authorised activities of the Security Owner

4.5 The Security Owner's Practices

As mentioned in the above, there are some security requirements cannot be addressed by the Scrum team or the Product Owner, thus this study proposes the Security Owner to deal with the security artefact 'Security Backlog' which is an agreed list of the security requirement items. The Security Backlog can be used for security requirements prioritisation and implementation in the Scrum.

- *In the sprint planning meeting:*

While the Product Owner can specify and prioritise system functional requirements in the Product Backlog and then the development team can select and build their portion of work in each sprint (Sprint Backlog). The Security Owner, on the other hand, can specify security requirements in a Security Backlog using UMLsec and prioritise security requirements with the Product Owner. The Security Owner facilitates the Scrum team, Product Owner and Scrum Master in refining both backlogs to be functions or actions in the Product Backlog with security tasks and activities in the Security Backlog.

- *In the daily Scrum meeting:*

In daily Scrum, the Scrum team initiates a follow up meeting to discuss the daily tasks and impediments (chapter two; section 2.8.5). The Security Owner can engage and observe the Scrum development team when they discuss the required security in the daily Scrum meeting and suggest what they should do before next daily meeting. The Security Owner can assist in improving the security level of the current product or software increment. Consequently, all Scrum team members will be more knowledgeable about any changes in security requirements.

- *In the sprint review:*

This review includes all the work done according to 'definition of done' and all security issues addressed that are presented to the stakeholders. Moreover, this process includes all essential updating of the checklists. Updating will be refined at each integration of the increments. For example, the following are some of these requirements that should be checked in every sprint and will be the responsibility of the Security Owner:

- Access control (AC).

  When the proposed system needs to be checked for access control (AC) because it is a critical resource system (e.g. Health Care Information System). However, adding more security requirement testing with risk and threat analysis would be a limitation and challenges the Scrum team.

- Authentication

To authenticate users of the software product that they are who they claim they are.

- Input validation

Testing and validating inputs to know what is providing input to the system.

- Output encoding

Encoding processes to be outputted to ensure dangerous characters are made safe.

- Cryptography.

Involves the protection of information through encryption into an unreadable format.

- Data protection

Protecting data during storage and transfer from unauthorised access.

- Error handling

The anticipation, detection and resolution of errors in the software.

- Communication security

To ensure the safe transfer of system messages and to avoid unauthorised access and intervention

- OWASP Top Ten (The Open Web Application Security Project)

A list of the top ten most critical web application security risks including examples of vulnerabilities and attacks and guidance on how to avoid them.

The above requirements are checked according to the Definition of Done which is discussed in the following section.

4.6 Achieving 'Definition of Done'

Definition of Done 'DoD' is a checklist of valuable activities required to produce the product. With the knowledge that 'Definition of Done' is based on two conditions:

1- Acceptance criteria that have been met.

2- Quality agreements.

All required notations are produced and modelled in an incremental manner (could be updated each iteration). It is essential to understand when a product increment can be called 'done' by Scrum teams and stakeholders.

Achieving 'definition of done' in misuse cases by the Scrum team is more difficult than the case with elements of the system functions/stories in the Product Backlog. For example, the completion of recording patient data in the medical record system (function) differs from those of data protection and ensure that no misuse or penetrate the created system. The reason is the possibility of misuse of the system resources occurring repeatedly during the sprints development and construction of the product increments. Thus, the Scrum team must review and check/test these misuse cases all the time every iteration. According to Ayalew et al. (2011), security activities should be to prevent/ mitigate these misuse cases in conjunction with the concepts of agile methods. The Security Owner is instrumental in establishing the definition of done through identifying security requirements or considering issues, they will help the team in this matter through the negotiation with the Product Owner.

4.6.1 "Undone" work sprint (for security check)

Delivering a done increment of working software is important to being successful in agile software development. In the Scrum framework, most Scrum teams produce partially done and incomplete increments of the Product Backlog requirements due to limited time boxes. At the last sprint, Scrum team members will investigate and check the whole required security requirements. This will be carried out with the assistance of the Security Owner. They will test any potential vulnerability that could be a gate for an attack with malicious code or misuse of the system. Some of these vulnerabilities don't appear in early sprints (could appear in integration or at the release level). As a result, the Product Backlog will be empty because all completed items are removed. All Security Backlog items will remain, but will be checked. Consequently, they will deliver a shippable secure software product to the customers for acceptance testing in the real environment.

4.7 Maintaining a Security Backlog

The Security Owner role will be instrumental in maintaining the Security Backlog. Dealing with the agility aspect that the team should preserve according to security issues is difficult, especially to fit and manage those security requirements in the short and limited sprints' time boxes, achieving all of them in every deliverable and shippable release product to be produced for the customers. Some of the security features are crucial and cannot be disregarded for any release. Notably, that in the release backlog; which is a prioritised list of the features (subset of the Product Backlog) for the targeting release; they have to consider threat analysis to prioritise security tasks in those limited time boxes, which the Security Owner will help to do. Besides, any additional or considerable security task in Sprint Backlog will hinder the team to control them in the specified time box and consequently result in increased cost and time. This means that all security tasks for a sprint should be made clear for the Scrum team before that particular sprint starts. For instance, checking a payment is <<provable>> and that there is non-repudiation for a digital signature to a contract or agreement in a purchase system, if they were taken as a higher priority in the following sprint. If the Scrum team considers performing security dependencies as an extra security requirements in the same sprint, this will impede the team in controlling their scheduled tasks.

Therefore, the Security Owner's role is required to manage the security requirements because the team cannot manage or incorporate all of the security requirements with all agile activities in a limited time considering the agile principles and conditions of right conducting, except if they can decompose/ manage and consider them from the early project planning.

4.7.1 Updating Backlogs in a Secure Scrum

Secure Scrum is an agile Scrum framework that integrates security with its processes and practices (Mougouei et al., 2013). Mougouei et al. (2013) proposes an approach to improve Scrum through introducing security analysis and design activities into the Scrum iterations, however, in contrast to the proposal of the present study such integration of security requirements consideration is not achieved through the inclusion of a Security Owner role. Building on the work of Mougouei et al. (2013) the addition of a Security Owner role has the following effects and the most important treatments to its related backlogs will now be demonstrated.

- *Product Backlog:*

The Product Owner with the Security Owner should identify related security requirements and the Product Owner can prioritise them according to the business value or return-on-investment (ROI). This is facilitated by the UMLsec method.

- *Security Backlog:*

The Security Owner who manages this backlog will include all agreements related to security issues and potential threats after they are prioritised by the Product Owner and placed in the list. While threats analysis and modelling take place in the sprint planning meeting, their mitigation or prevention strategies, thereby could be placed on the Sprint Backlog. The modelling of security requirements is facilitated here by the use of UMLsec with the Security Owner role.

- *Sprint Backlog:*

During the sprint planning meeting in secure Scrum, the development team can select and estimate some of the security requirements by UMLsec from the Security Backlog facilitated by the Security Owner; - refined Product Backlog – more specifically as a slice that can be modelled and tested in the specified sprint time box. They can estimate the items after breaking the big security issues with the Product Owner or either they can estimate their impact on the next sprint.

Hence, the Scrum team has to consider and deal with security requirements in daily Scrum, sprint review, and retrospective meetings to improve satisfying all of the required and changed security features and it is in these activities where the Security Owner will provide assistance.

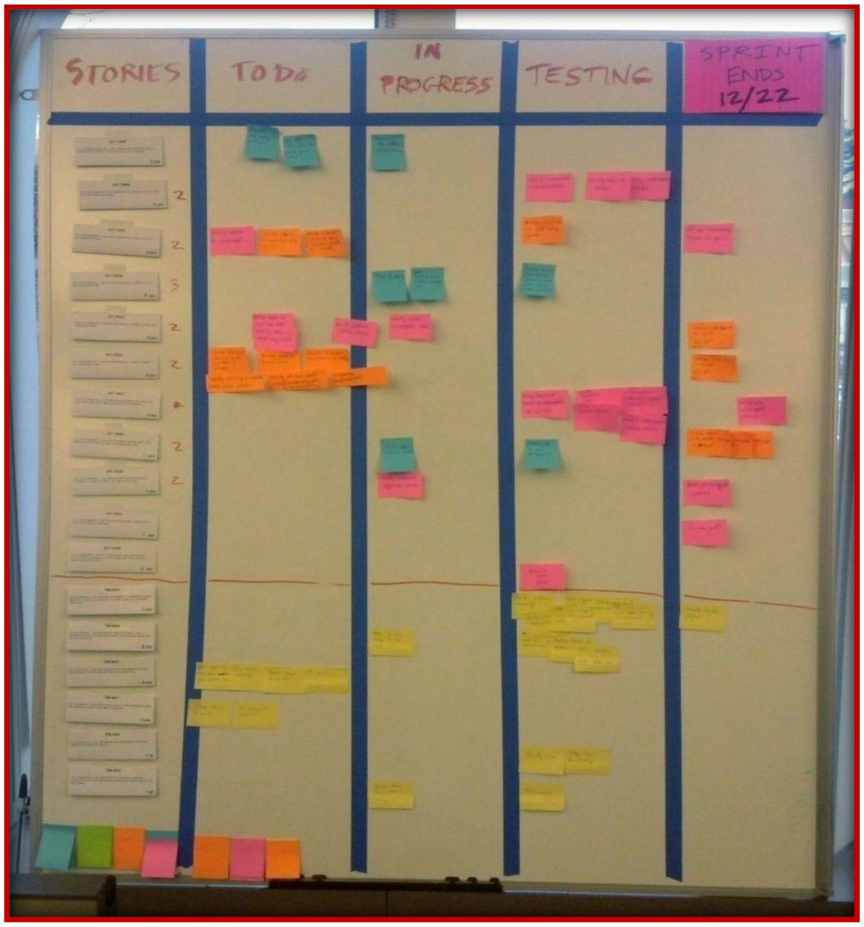


Figure 4.1: Scrum Task Board (from: Scrum (Software Development))

Note that the security requirements cards could be green cards and posted in Scrum task board for implementation of their queue of priority (Figure 4.1).

4.7.2 Security Backlog

The proposed *Security Backlog* is a backlog to consider security issues that are related to the main user requirements of the proposed system. This process will be in the agreed and follow the *Product Backlog* suggested and prioritised by the Product Owner. The Security Backlog should preserve the system agility when deciding to use an agile or Scrum framework and is controlled and maintained by the Security Owner. The Security Owner has to specify potential risks from a security point of view and according to essential documentation required to be presented in the Security Backlog. The Security Backlog should be simple to create simple documents without affecting the principle of agility. During each sprint, the Security Owner has to implement security testing for that feature or security task. Specification of risks by the Security Owner is similar to risk management in Scrum whereby risks are managed proactively rather than ignoring risks until they become an issue.

Figure 4.2 illustrates how the Security Owner deals with the Product Backlog while considering and incorporating security requirements. As a result of the Product Owner and Security Owner negotiation, they will produce a new *refined Product Backlog*. The refined Product Backlog will include all the essential and considered security requirements before selecting the release backlog for the coming release. UMLsec will help the Scrum team and the Security Owner to model these security requirements which are listed in the Security Backlog.

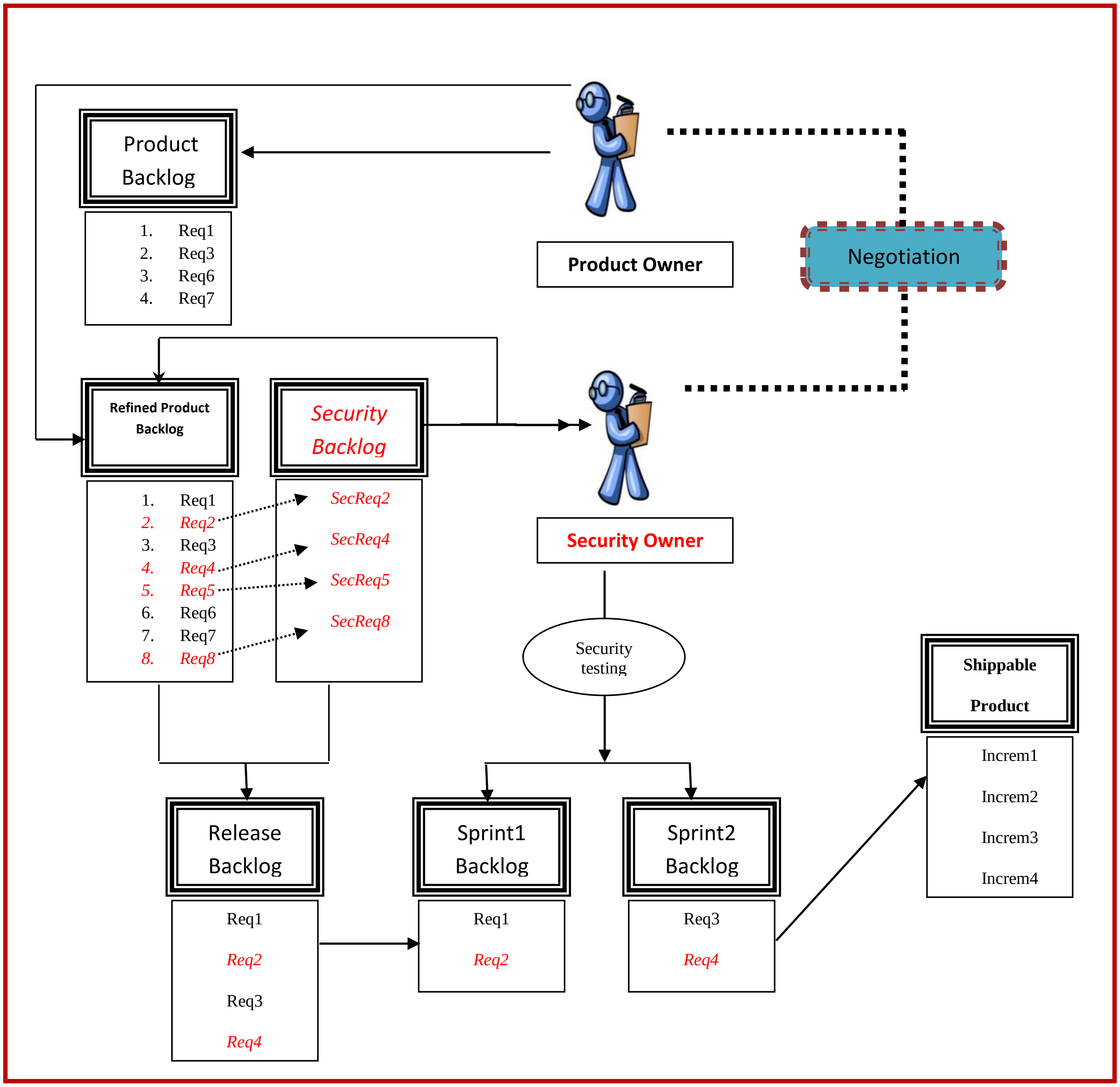


Figure 4.2: Refined Product Backlog (by Security Owner and Product Owner negotiation).

The Security Backlog artefact produced is different from the Product Backlog and the team Sprint Backlog. It remains as it is and they will not remove the tasks from the top when an increment product or software is built. Instead, its items can be checked as done or tested taking into account that it could be tested again. This means that the Scrum team should consider security issues all the time during the sprints. This is another part of the Scrum process where the Security Owner can facilitate the Security Backlog in the sprint.

Together with the Security Owner and their input, in sprint planning, when the development team agree on tasks that should be done in the current sprint (Sprint Backlog), they also agree on Security Backlog items that should be tested and checked during this sprint (the highest priority items from the refined Product Backlog).

4.7.3 Security Stories

A security story is a story proposed by the Security Owner to mitigate the potential risk or prevent misuse. For instance, ‘As a car owner, I want to lock the car, so that I can reduce the probability of theft'.

For security use cases, their work and function is centred on minimizing, mitigating or preventing risks and potential threats initiated by penetrative systems, attackers or the misuse of available resources required to complete developing the software and system functions.

Therefore, the most appropriate place to represent these security use cases will be in the middle – between use and misuse cases, while modelling system requirements, i.e. it is among the functional requirements to strengthen and develop functional and user stories in the system and between the cases of misuse to prevent or mitigate the risks and/ or potential threats. The following figure (Figure 4.3) shows how these security use cases can be presented in double circles as the drawing by an agile Scrum team.

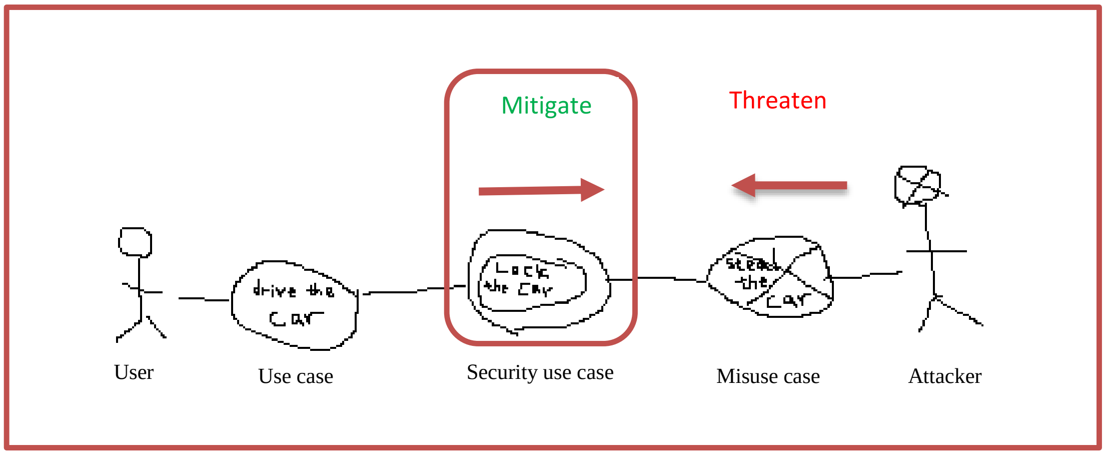


Figure 4.3: Security use case

The following are the proposed shapes of these above mentioned entities:

- An actor (user) can be represented by a normal stick with an empty circle as a head.
- An actor (attacker) can represented by a normal stick with a cross sign inside the circle as a head.
- Use case: an oval shape (hand drawing) that represents a function which the system has to perform it.
- Misuse case: system misbehaviour, which is same as the use case shape, but with a cross sign inside.

The following cards are examples of the above figure that the Scrum team uses while considering security in their planning. Misuse stories/cases will be mitigated by security cards placed on the task board.

- Example of a user story card:

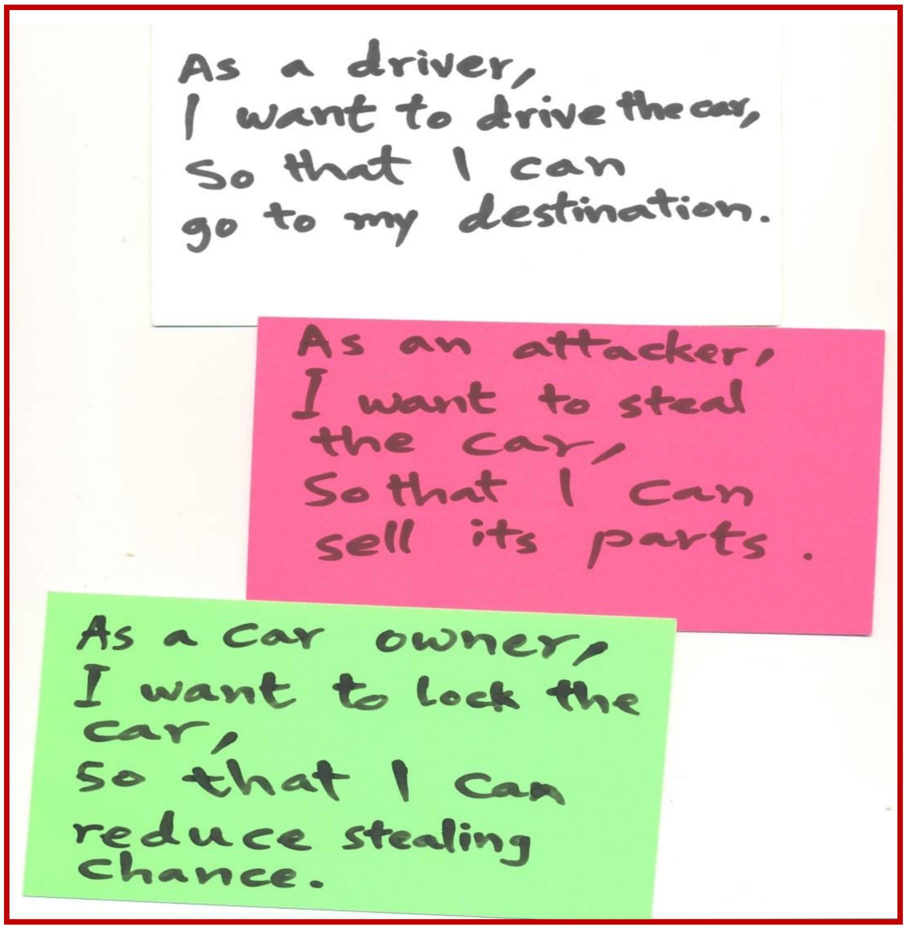
'As a driver, I want to drive the car, so that I can go to my destination'.

- Example of misuse story card:

'As an attacker, I want to steal the car, so that I can sell its parts'.

- Example of security use case story card:

'As a car owner, I want to lock the car, so that I can reduce the probability of theft' (Figure 4.4).


As a driver,
I want to drive the car,
So that I can
go to my destination.

As an attacker,
I want to steal
the car,
So that I can
sell its parts.

As a car owner,
I want to lock the
car,
So that I can
reduce stealing
chance.

Figure 4.4: User/Misuse/Security cards

4.7.3.1 Security Stories Cards

An agile team can deal with the functional requirement (user stories) cards as white cards whereas security requirements stories are in a different colour (e.g. Green). The reason behind the different colours is so that each team member can know when those cards are attached to the board, the ratio and the amount of such security requirements that have been discussed in relation to the rest of the cards and the functional elements and therefore, team members can visualise the security of the system. However, it is not necessarily the number of security story cards, which have been stuck on the board, that reflect the quality or the level of system security.

The Scrum team can incorporate security requirements of the UMLsec facilitated by the Security Owner as a step towards in analysis and model them as their importance to be ordered in iterations.

4.7.4 Misuse cases in the Scrum framework

Misuse cases are a technique that team members and the Security Owner work on together. Misuse cases have been illustrated in previous chapters and reflect the issues or features from the attackers point of view. They describe breaches security principles or the misbehaviour of the system that has fallen under attacker or malicious code control. Misuse cases is a cornerstone in the planning game. The Security Owner helps identify and negotiate misuse cases in the release planning meeting. The team agrees of the security features list (Security Backlog), this list will be pasted on the task board after finishing user and security story cards. This process, that includes functional user stories and functional and non-functional security story cards, are implemented by team members with the Security Owner during the *planning poker* (Williams et al., 2009) for all the required issues that are related to the systems.

When conducting planning poker in the meeting, potential misuse cases relevant to the current increment are discussed, negotiated and listed on the board according to their priorities and importance, the most important misuse case is placed at the top. The Scrum team together with the Security Owner can create misuse cases in a different colour; red was proposed for each of the potential written misuse cases. For each misuse case, the Security Owner helps the Scrum team to address and suggest security tasks.

The following table (Table 4.2) differentiates between different issues to help the agile team while modelling misuse stories and security requirements:

The issue	**Card colour representation**
User story	White card
Misuse story	Red card
Security story	Green card

Table 4.2: User/Misuse/Security stories representation

Examples of misuse cases notation in Scrum:

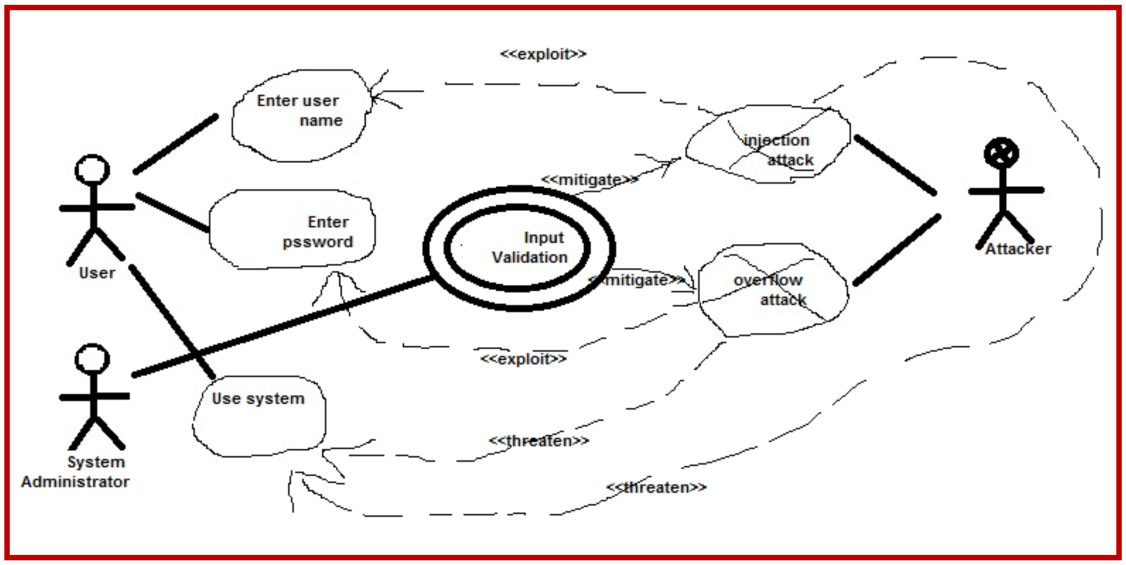


Figure 4.5: Misuse case model

Figure 4.5 depicts an example of the misuse case model using notation proposal to illustrate (mis) use cases in the target system as well as security use case. The Scrum team should know how use cases of the system work and what are the expected misuses. Then, with the Security Owner, they can determine and develop the required countermeasures of the security use cases. The Scrum team can produce the following artefacts in their analysis:

- Abuser stories: these identify how an attacker could abuse the system and jeopardize stakeholders' assets (Peeters, 2005, Neher, 2012).
- Misuse cases: as discussed above (Clagett II, 2009).
- Attack trees: which help to identify all possible attacks of any system threat (Clagett II, 2009).
- Threat trees: in more abstraction than attack trees, threat trees assist the Scrum team to see a variety of the system expected threats (Clagett II, 2009).

In fact, support from management and the Product Owner would be needed in case some security issues could not be addressed in the sprint. The Product Owner and the team will do "Release Planning" before starting the first sprint and "Product Backlog Refinement" after each iteration. This will help the Scrum team’s checking and testing within the sprint's time box, not to delay them. This justifies the role of the Security Owner and the necessity of existing security professional/specialist to help solve these issues in each Scrum cross-functional team.

4.8 Security Requirements in Planning Poker

The focus of this section, in illustrating how the Scrum/ agile team play planning poker, is the *security poker* where each member estimates a value for the current presented security issue. With the help of the Security Owner, the team members negotiate and justify the estimated highest and lowest security requirement value when a turn is initiated during the meeting. Alternatively, team members can play *Security Risk Poker* using different colour cards "red – orange – green – white" that reflect the value of the potential risk after estimation agreement (Jurian van de Laar, 2012), (its value = the probability of risk that could happen * impact of that risk when happened).

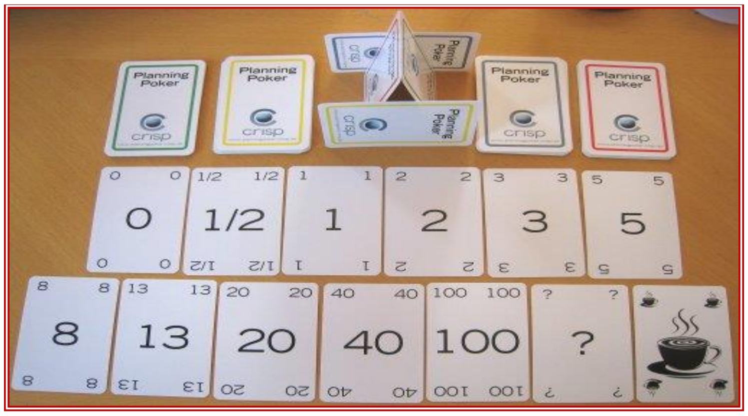


Figure 4.6: Planning Poker deck of cards (planning-poker)

Protection Poker (Williams et al., 2009, Williams and Meneely, 2010) is an informal activity of misuse and threat modelling development that the agile team can use using the planning poker cards as shown in figure 4.6 to estimate their effort in addressing potential items.

4.8.1 The Product Backlog effects on the Security Backlog items and vice versa

Scrum teams are self-organised and their work is integrated to produce the final product. Any change in any element in the Product Backlog will affect the work of the team as a whole and the results expected by customers. The addition of any element to the Product Backlog must be considered from the perspective of security and its impact on the security of the system and subsequently will result in a change in the Security Backlog. Consequently, the team, including the Security Owner, discusses and works on the added item and amends the Security Backlog. This will be applied to the items in the Security Backlog itself and the extent of the impact tested to add functions and elements of the Product Backlog.

4.9 Using UMLsec in agile

The current question is how to integrate and include some of these essential security requirements of the system in an agile process and rapid development (McConnell, 1996), for instance the unified modelling language (Larman, 2005), and preserve security in the short iterations without affecting time or customer satisfaction producing this integration, a proposal of Jurjens (2002) in the framework of the Scrum (Shore and Warden, 2008). It is the modelling of security requirements using UMLsec that the Security Owner will be involved with.

The limitation of time boxes is related to the fact that dealing with some security issues is difficult and cannot be included as a whole in these time boxes in agile iterations. However, the inclusion of a Security Owner role would help to decompose the security issues so that they can fit into the time-boxes. For instance, role based access control <<RBAC>> requirements in the patient health care system for all of the departments with only two weeks of a time box. Those security requirements should be decomposed or analysed with the support of the Product Owners to control within time boxes. Decomposition of the RBAC model to assign roles, grant permissions and access authorisation to the resources.

Scrum teams must manage carefully their specified Security Backlog before producing the refined Product Backlog. For example, the Scrum team can decompose the system RBAC requirement in the Security Backlog into three requirements (assign roles, grant permissions and access authorisation) in the refined Product Backlog and manage them in different sprints. So, they can provide a valuable and more secure product when they can decompose and manage such large security requirements and be aware of what can be misused in the next iteration. For instance, expected unauthorised access to patient records.

4.9.1 Applying UMLsec in agile methods (The Integration)

Developers and customers are more focused on functional requirements rather than on the security of the system and developers may not be capable of finding security requirements related to the functional requirements and leave them for later phases. It is for these reasons that there is a need for the inclusion of a Security Owner role. In UMLsec and Secure UML, several solutions were proposed, but they were proposed for a specific organisational problem. If the developers try to implement the proposed solution – UMLsec/ Secure UML solution - in another organisation-problem area, this proposed solution will not work well. Customers are not aware of all threats related to the software, for instance, the aforementioned unauthorised access to the patients records. Furthermore, developers are not experts in finding threats and vulnerabilities. Therefore, security experts, in this case the Security Owner, are needed in risks and threat scenario modelling for security testing and vulnerabilities terminating. The Security Owner is needed in solving this kind of work through providing the Scrum team with the expert solutions.

In Agile methods, there is the concept of sprints, product increment, Product Backlog and Sprint Backlog to be maintained to keep track of what is required and what is in the process of development in each sprint. Therefore, the overall use case diagram can be divided on the basis of sprints. This makes the problem of the required system simpler since there is a decision required if a misuse case should need to be handled at the Product Backlog level or at the sprint level. However, the experimentation proposed in the present study is to see if it would be useful to identify misuse cases using UMLsec at the overall product level or at the individual sprint level. The latter would require identification of either additional artefacts or connections so that the code developed as a result of identification of misuse cases is logical (the increments will be built in a logical way). Since other team members build these artefacts, they need assistance in identifying the security issues.

4.9.1.1 Integration Parts

Depending on an agreement and 'definition of done' required by the agile team, most parts of this integration in modelling and diagrams' representation are as follows:

- The Misuse cases
- Simple class diagram (Figure 4.11)
- Sequence diagrams (Figure 4.10)
- Simple state-chart diagram
- Simple picture of deployment diagram

4.9.2 Prioritisation

Through the assistance and involvement of the Security Owner the Scrum team and the Product Owner can deal with security high level design at release planning. They can inherit the benefits of *risk poker* (Williams and Meneely, 2010, Jurian van de Laar, 2012) and *protection poker* (Williams et al., 2009) techniques in prioritising important and crucial security requirements and stereotypes of UMLsec profile. For example, a member can record <<critical>> or <<data security>> stereotype in a red card with its degree to negotiate with other members if it is most important as a security issue in the meeting from his point of view. User story cards with security cards would be stuck afterward on the task board after prioritising. A selected slice of work in each sprint would follow the same procedure for estimation around the development team in the sprint planning meeting. However, because team members do not normally specialise in security requirements, they would need help from a Security Owner in coordinating security issues in this way. The Security Owner role, as a proposed role in the present study, will assist the Scrum team in the consideration, modelling, coordination and management of these security requirements.

4.9.3 UMLsec in Agile Advantages over the Traditional Process

The following points illustrate some advantages of using UMLsec in an agile development methods compared with the traditional waterfall method:

1- Time of security - In agile, every iteration considers security requirements, but is not limited to security, while in traditional waterfall methods distinct security-focused project phases to deal with UMLsec security requirements are often located at the start or at the end of a project.

2- Resources of security - In agile, although team members are not security specialists and need help with security requirements coordination, security skills are embedded in the team, while in traditional process security skills are externally sourced, often isolated from development and test resources.

3- Validation of security - In agile, there is hybrid UMLsec security and functionality testing throughout the project, while in traditional methods there is often an explicit security testing phase, often at the end of the project.

4.9.3.1 An example of Using UMLsec Security Requirements

The following are some security properties that the Scrum team can consider while modelling UMLsec in an agile process with the assistance of the Security Owner role:

- Secrecy: which means the data should be read only by authorised parties.
- Integrity: which means the data should be modified only by authorised parties.
- Authenticity: message authenticity means data can be traced back to its original source; and entity authenticity means identifying the participants in the protocol.
- Freshness: to make sure that the message has been created during the current protocol execution.
- Secure information flow: to insure that there is no leakage of sensitive data, for example, <<no down-flow>> and <<no up-flow>>.

4.9.3.2 UMLsec Notation and Diagrams in the Scrum Framework

One of the proposals of the study is to include UMLsec towards improving security requirements modelling in Scrum. Therefore, it is necessary to show the notation of UMLsec used in an agile development process. The following is an example of UMLsec notation in a secure channel subsystem drawn by the Scrum team during their agile process (Figure 4.7) which is incremental in development. Here the Scrum team assemble the diagram to firstly show the sender and receiver as an activity diagram, with the help of the Security Owner they add a state chart diagram of sender and receiver followed by establishing UMLsec stereotypes and interfaces for both sender and receiver as class diagram. With the help of the Security Owner a deployment diagram which specifies stereotypes and security features that are required and the interactions between them.

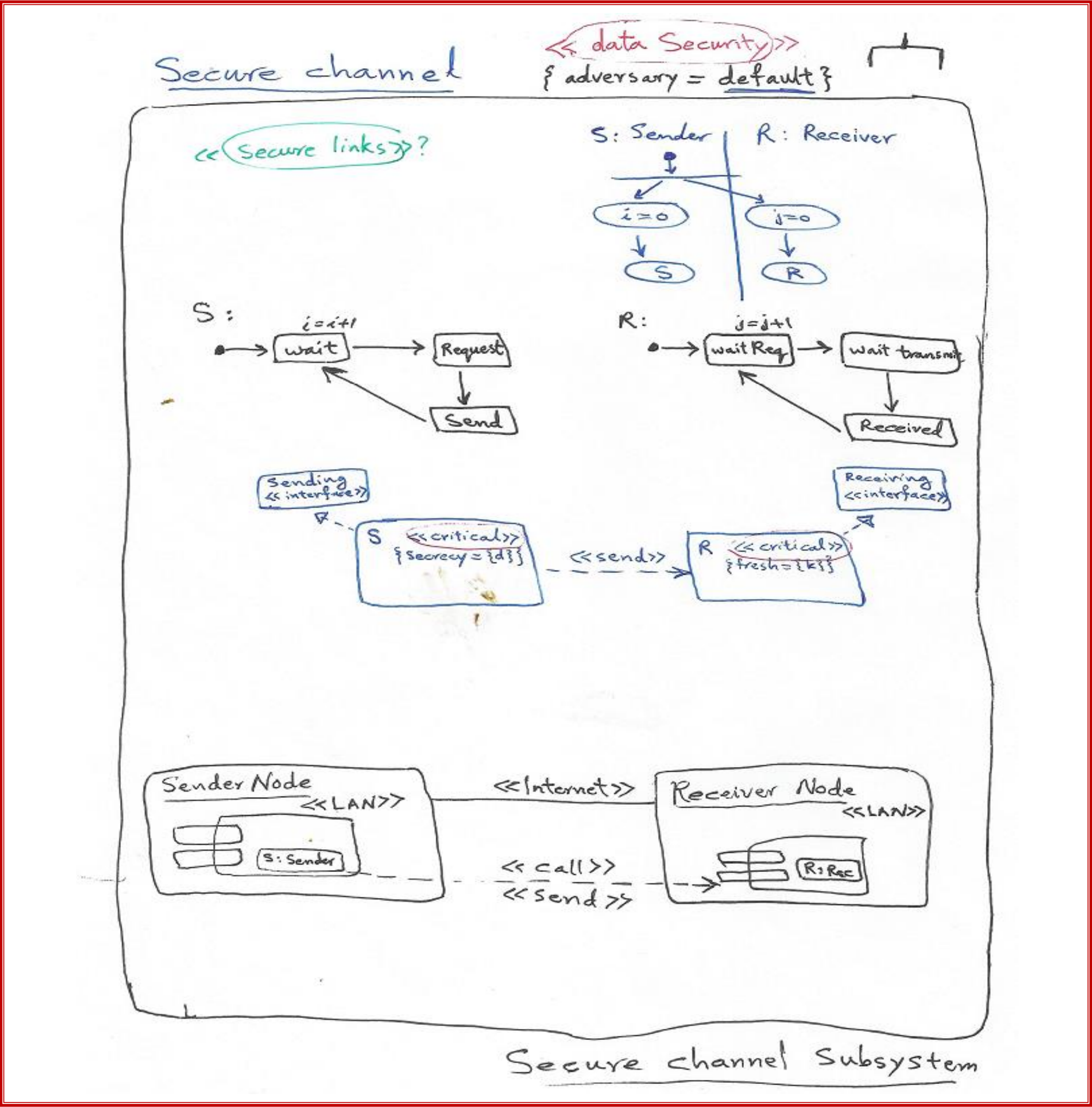


Figure 4.7: An example of Secure Channel Subsystem by the Scrum team.

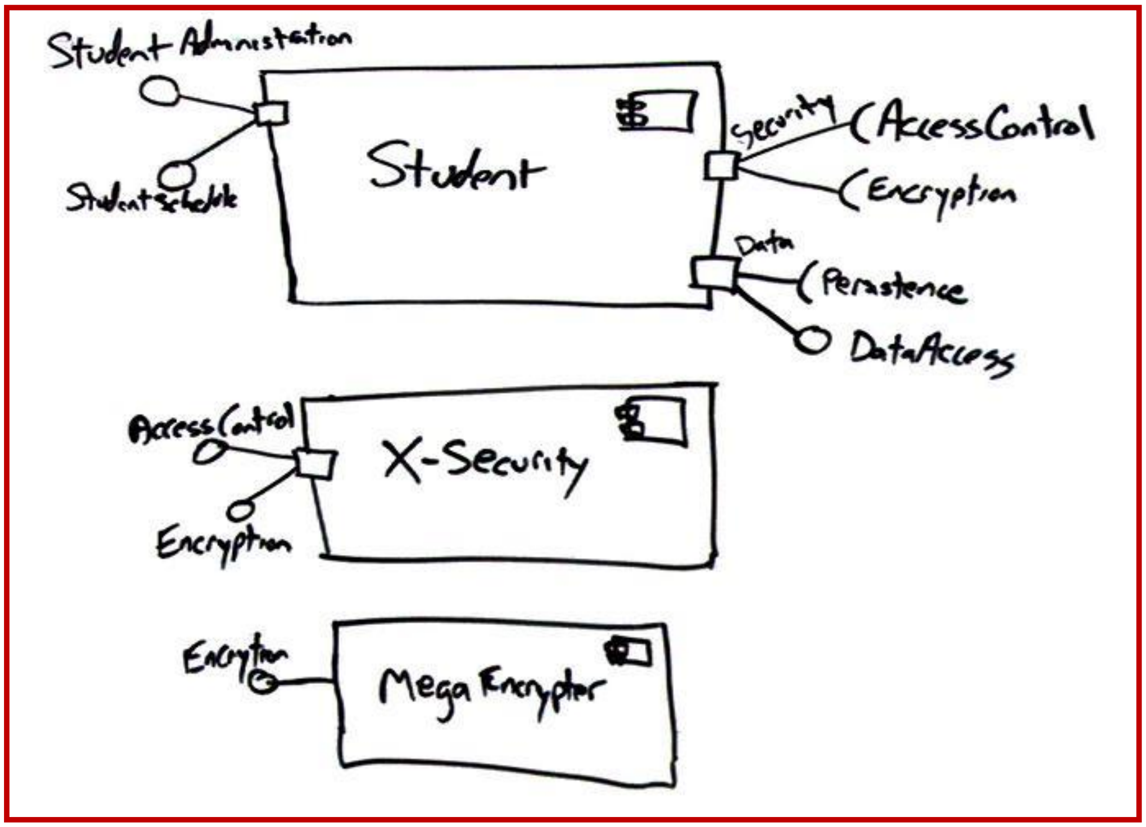


Figure 4.8: An agile component diagram

Figure 4.8 represents an example of an agile component diagram which shows the various components of the system being developed and the security considerations for each component. In UMLsec components and their associated security requirements are modelled in this way. In the example (Figure 4.8) the student component comprises of the facilities of student administration and student schedule which would require access control and encryption as security features. This security feature, denoted as X-security in figure 4.8, is then considered alone with the associated requirements of access control and encryption, in the example here, thereafter, the encryption requirement is then considered shown as mega encryptor. The addition of a Security Owner role can assist the agile team in the development of this component diagram using UMLsec.

with its ports and interfaces to model security requirements which is obviously understandable by the development team in the Scrum and other stakeholders including consumers, non-technical domain experts, testers and also by management.

For the sake of modelling security requirements in UMLsec, the Scrum team can work together with the Security Owner for the generation of ideas in one unified form understandable by all members of the team in order to implement the requirements. Some examples are reviewed below (Figure 4.9, 4.10, 4.11, 4.12) which explain how the agile team modelling and representation of diagrams required in the preparation of requirements' implementation, including security requirements, and how they are put up for discussion and continuous adjustment and improvement. It is important to consider the nature of agile products development in agile, and this reflects that the team at each stage or iteration makes a simple addition within the specified time boxes as an increment to the final model. Figure 4.9 shows how the Scrum team can represent the use case diagram. In this example, the use case represents that the student needs to enroll in the university in order to search and register for seminars.

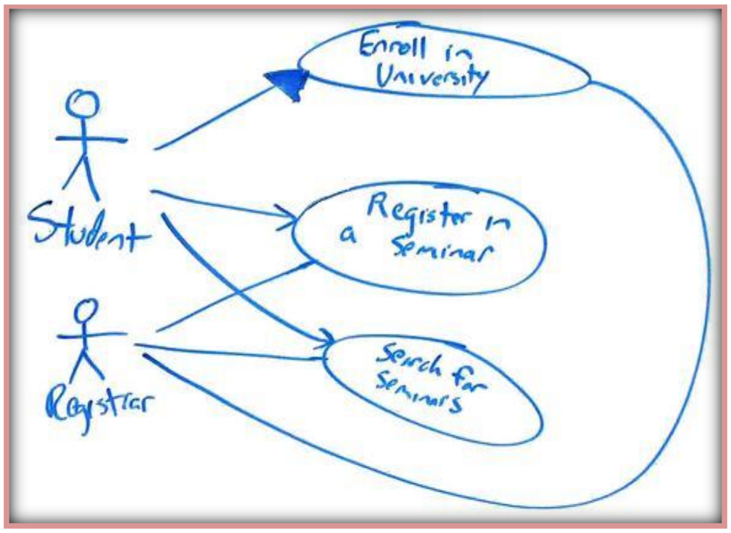


Figure 4.9: An example of agile use case

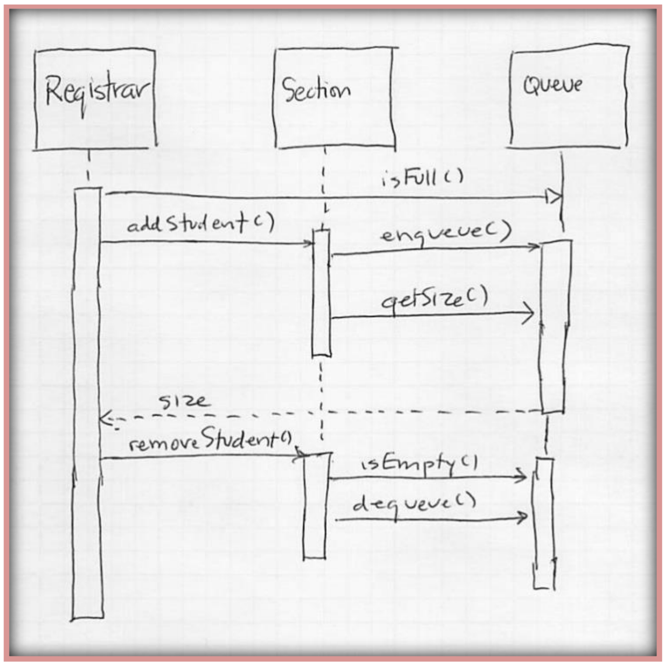


Figure 4.10: An example of agile sequence diagram

In an agile sequence diagram, as shown in Figure 4.10, by using UMLsec additional security requirements could be added. It is proposed in this study that not only will the inclusion of UMLsec improve security requirements modelling but that this use of UMLsec can be facilitated by a Security Owner role. This applies to all aspects of the modelling of requirements, which includes the initial use cases (Figure 4.9), the sequence diagram (Figure 4.10) and the class diagram that shows the relationships between the different classes (Figure 4.11). For the latter the Security Owner can help the team to develop and visualise the security issues required for the class diagram.

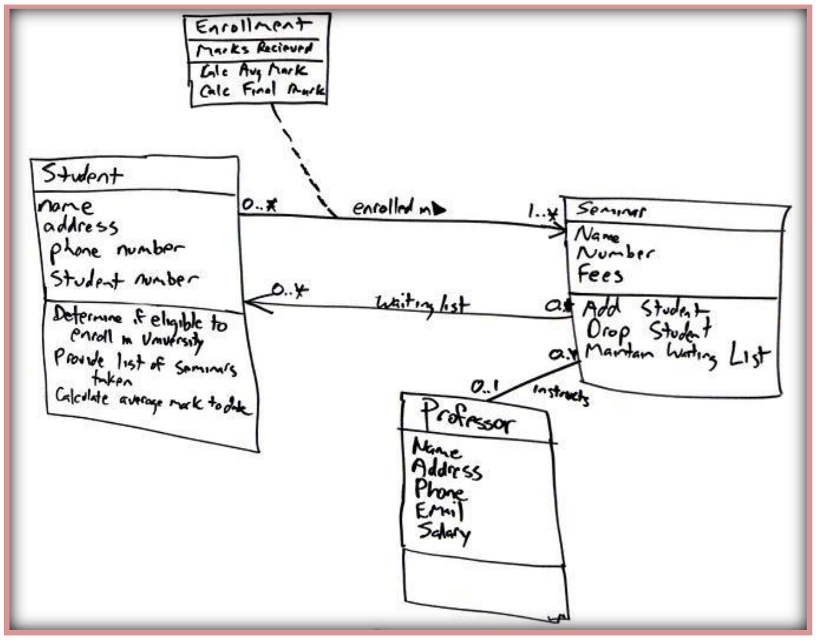


Figure 4.11: An example of drawing class diagram

4.9.4 Collection of Simple Diagrams

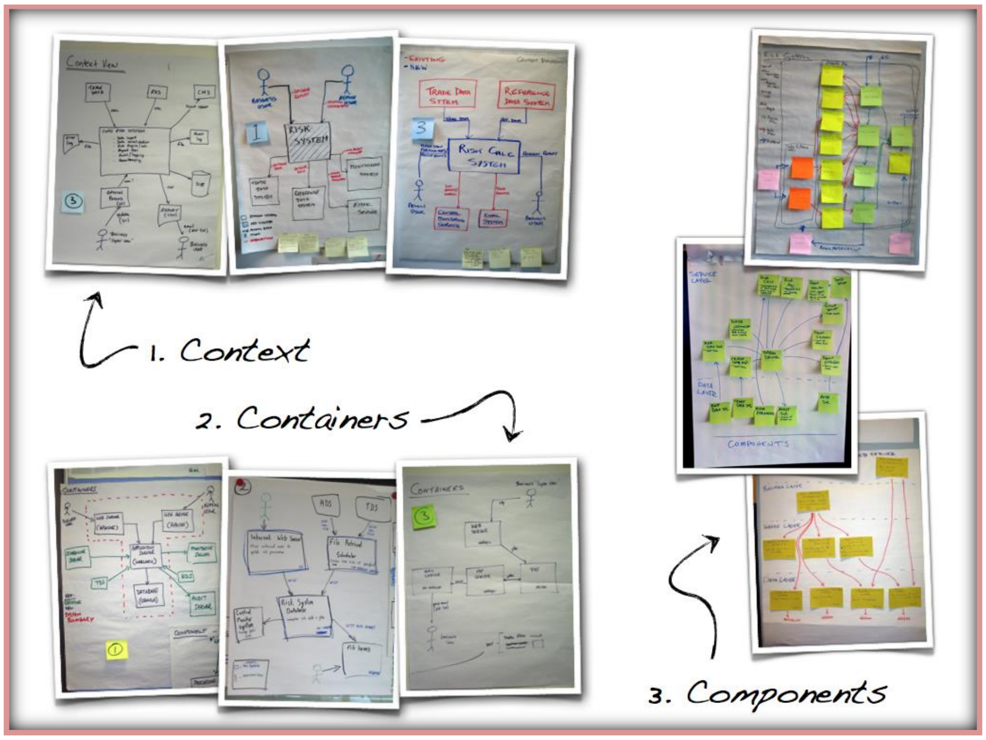


Figure 4.12: Collection of simple diagrams

As shown in the figure above (Figure 4.12), a collection of simple diagrams allows the Scrum team to describe the software from several perspectives to include different elements describing the whole system, compared to a single presented diagram. This is a simple way to visualise and understand the whole picture of the proposed software by all of the team members. Here the Security Owner role can help the team to negotiate and emphasise the security requirements through including and organising individual diagrams towards helping the team to visualise the entire system. For distributed Scrum teams, where teams are distributed geographically, these diagrams can be presented and amended online using online Scrum tools or software such as ScrumDesk.

4.9.5 Prioritise UMLsec Security Requirements with Kanban in a Scrum Framework

Kanban uses white board and visual cards to represent the items as a Product Backlog, which is also ordered in columns (new, in progress, and completed) in a simple and efficient way to smooth the workflow. It could be used as well to include the Security Backlog items stocked and ordered in the same Kanban by completing the specified security issues (Table 4.3). With this process, team members can deliver the completed work and features to the customers each time to get their feedback. To ensure of work progress, Kanban limits the number of items under the column of work-in-progress (which is called WIP-LIMITS). They can be set with work remaining/time chart or Scrum burn down chart smoothly and in a consistent way. Consequently, with a collaborative team, they have to finish one item in this limit before starting to add or move a new item from the column. WIP-Limits help to keep work flowing, save time by eliminating too much task switching and complete tasks (on-demand improvement).

Kanban is a lean and agile methodology that can be used to improve any software development practice like Scrum. The Scrum team can use Kanban in conjunction with a Scrum development process (Scrumban) (SwitchingToScrum.com, 2013) to prioritise and deal with security requirements when including security items to be implemented or in-progress for the specified system. Kanban will be more valuable if an agile team uses this technique within a Scrum project management framework. The Security Owner will assist the Scrum team to arrange UMLsec security requirements and constraints and prioritise them using this technique with both the team and the Product Owner. The Scrum development method provides the structure for organising feedback, short time planning, stack ranking, and divide the work in manageable sprints. In Scrum sprints, some challenges such as unclear development steps, task switching and partially done work could appear. These would be sorted out using a Kanban technique through splitting and dividing tasks into columns, guaranteeing a hundred percent task completion in comparison to Scrum with only eighty percent task completion, in addition to streamline delivery task by task. OnTime Scrum is an example of the several tools that the agile team can use during development to visualise features, backlogs and dashboards. The team can use these tools to prioritise and model security requirements in a similar manner and scenario of modelling those requirements and arranging them in Kanban columns.

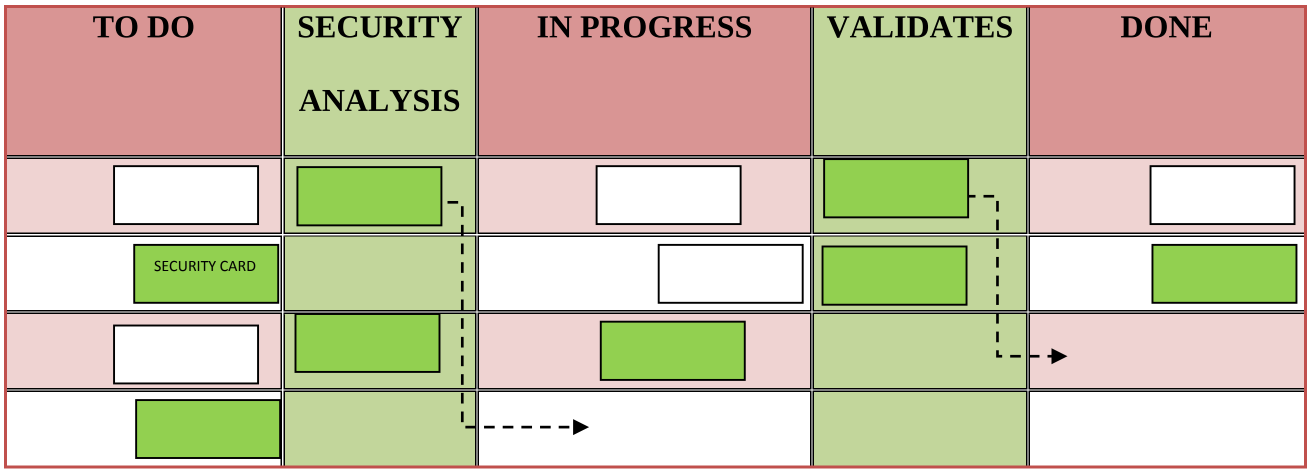


Table 4.3: Kanban schedule system with security cards

4.9.6 Planning Poker with UMLsec for Proposed Security Requirements

The adaption of gamified techniques such as Planning Poker to security is useful and described in the following. It was pointed out in the above that the security poker is a method when dealing with UMLsec security requirements. Security poker has the following steps in the planning meeting:

- The Product Backlog stories are presented by the Product Owner.

- The team has a time-boxed conversation to reach a user story common understanding.
- Each participant selects a card depending on his/her estimate, which is hidden from other players.
- All cards are turned over at the same time.
- If their estimates are close to each other, they select the highest frequency estimate; else they have a time-boxed conversation to discuss the estimates focusing on the outliers. Then the team plays again until all estimates converge.

The following steps are added when considering Security Poker for the proposed security requirements:

- The Security Backlog stories are presented by the Security Owner.
- For each story, the Security Owner identifies the related security issues.
- The team delivers a time-boxed conversation to make a security story common understanding.
- Each participant selects a card depending on his/her estimate of this security issue, which is hidden from other players.
- All cards are turned over at the same time.
- If their estimates are close to each other, they select the highest frequency estimate; else they have a time-boxed conversation to discuss the estimates focusing on the outliers. Then the team plays again until all estimates converge.

The refined Product Backlog is produced after security poker game. It includes all ordered items and security tasks that the Scrum team has to accomplish. Again the Security Owner will assist the Scrum team to prioritise UMLsec stereotypes and requirements through the above security poker game. For example, the team can rank <<rbac>>, <<data security>> and <<secure links>> at the subsystem level, or they can order <<critical>> and <<guarded>> in the refined Product Backlog at the object level (Figure 4.13).

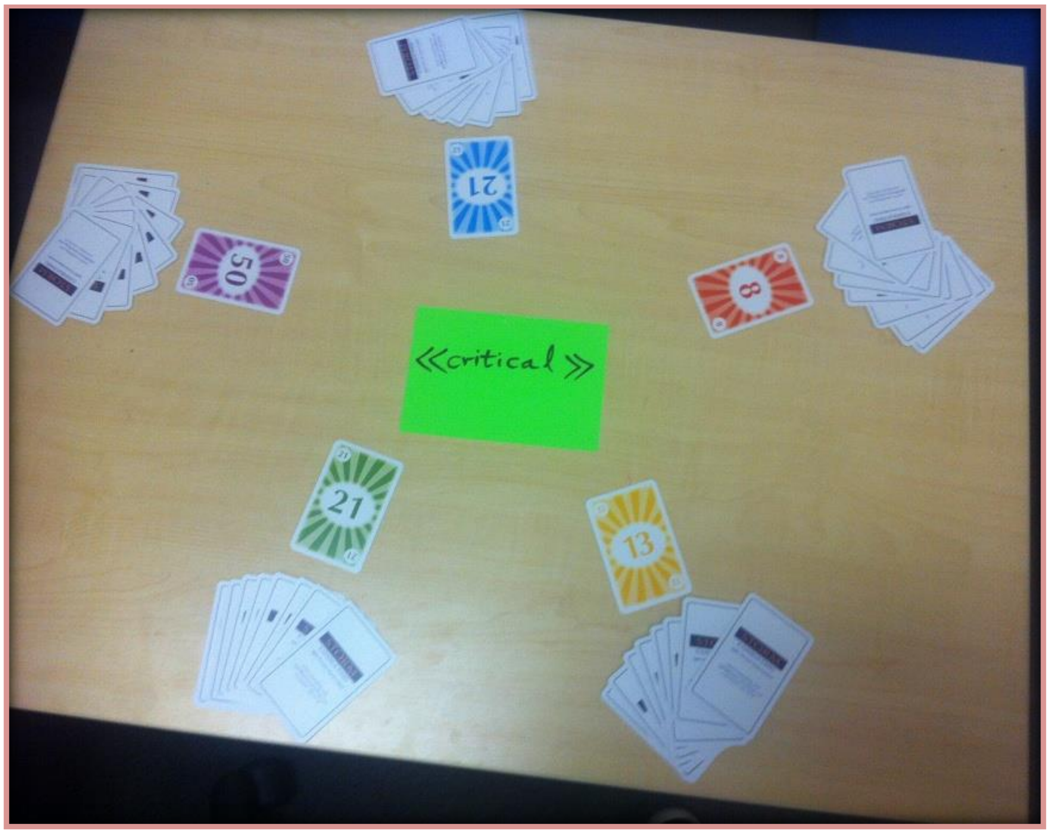


Figure 4.13: Scrum security poker of UMLsec <<critical>> stereotype

4.10 Security Requirements Burn Down Chart

As an artefact the Scrum burn down chart is to illustrate for the Scrum team the relationship between remaining work or tasks and the specified time line to finish and complete these tasks. Alternatively, a security burn chart as another artefact is proposed that demonstrates to the team, Product Owner, developers and Security Owner the current state of security tasks remaining to be performed and checked in the specified time-boxes before producing a deliverable increment to customers. The following diagram (Figure 4.14) is an example of the diagram that is supposed to be negotiated by the team in every meeting that takes place. At the end, when all security requirements are checked, the security burn down chart will reach the zero level. All security items remain in the Security Backlog, but are checked.

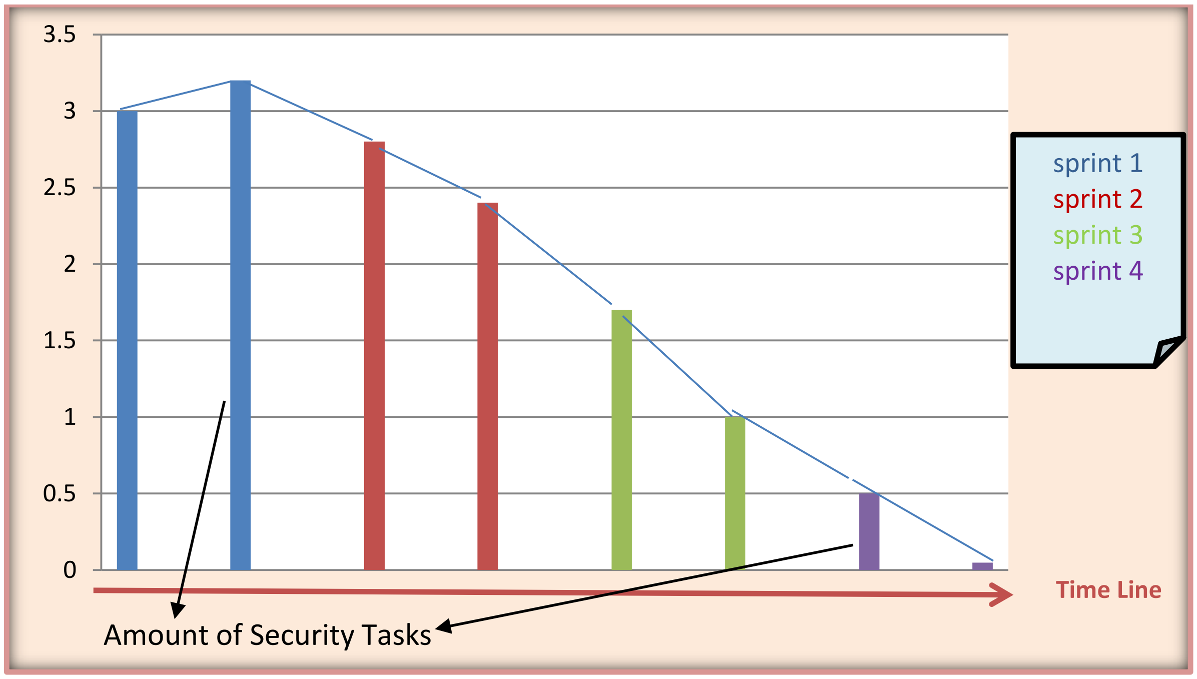


Figure 4.14: Security burn down chart

Another tool in addition to the security burn down chart is the risk burn burn down chart (Figure 4.15) which provides a representation of the risks' status across the iterations. This approach offers an indication of how risks are controlled and managed.

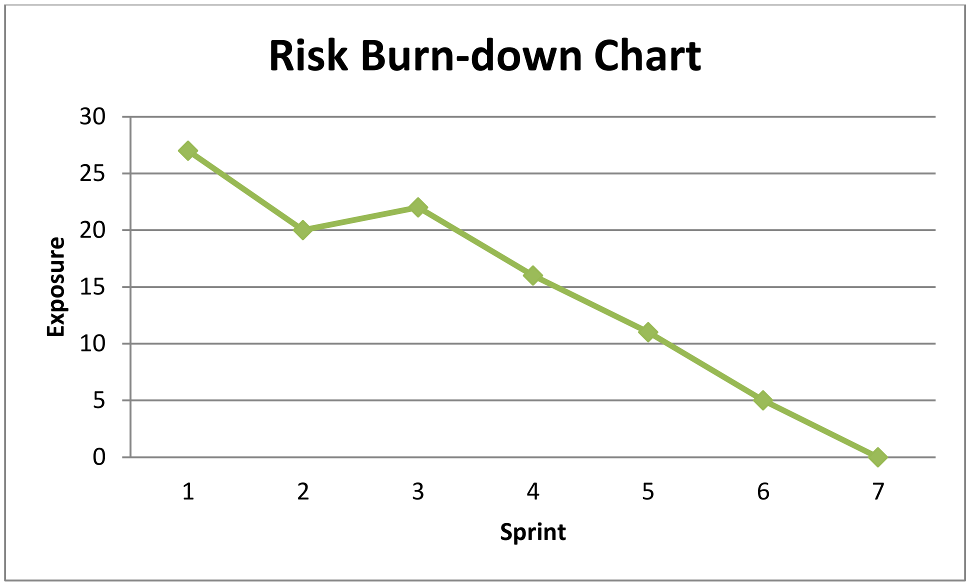


Figure 4.15: Risk Burn-down Chart

4.11 Awareness of Security Change

As an education and knowledge share, the whole team should perform some training for stakeholders and Product Owners with the Security Owner (such as SDL training) as awareness of potential security issues to address all needed security requirements. Secure design and coding, threat modelling and security testing are candidate topics for the Scrum team training session. The Security Owner has to make sure each time, for each sprint’s iteration, that the current users’ or customers' security requirements are fulfilled and satisfied. The Scrum development team can then follow the work progress with the respect to the security change phenomenon. The Scrum team should participate with the Security Owner taking the security requirements change in each iteration keeping users in mind. So, they have direct impact and feedback in their modelling of security requirements and the Security Backlog that affects the whole refined Product Backlog.

4.12 The Integration Framework

The following diagram (Figure 4.16) illustrates the integration process by the Scrum Team when using the security poker technique. The Scrum Team can use security poker to deal with misuse case artefacts. Furthermore, it can be used in risk identification (Risk ID) and threat analysis (Thread A.) and UMLsec security requirements. The Security Owner can specify the related Security Backlog of the system in order to produce the refined Product Backlog with the Product Owner.

Figure 4.16 is an illustration of the proposed extended Scrum framework. The artefacts include misuse case artefacts and Risk Identification and Threat Analysis, however, as part of the extension of the Scrum framework in this study the artefacts also include UMLsec stereotypes (profile). The techniques used in the extended Scrum framework include the two aforementioned security poker games Risk Poker and Protection Poker, these are in addition to the use of white boards and story and security cards.

All of these tools are used for security requirements consideration prior to the next stage which is the modelling and prioritising of security requirements which is achieved through the addition of UMLsec to model security requirements and facilitated by the proposed role, namely Security Owner role, as part of extending the Scrum framework. During the product development stage which includes the Product Backlog, the Security Backlog and the Sprint Backlog which can be managed with the assistance of the Security Owner role, as an additional extension to the Scrum framework, there is facilitation of these artefacts by the Security Owner through liaising with the Product Owner and the Scrum team.

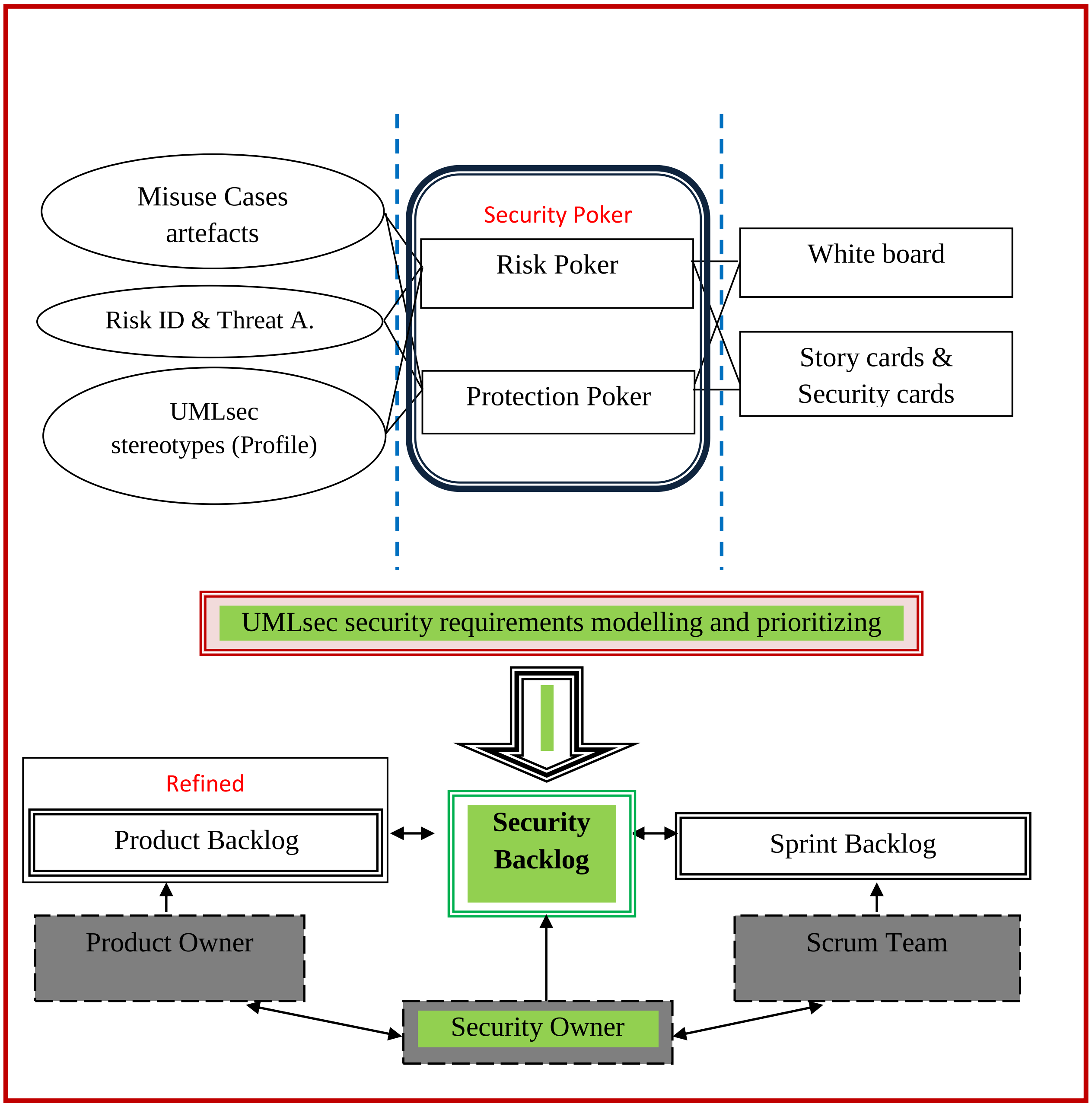


Figure 4.16: Security Owner and UMLsec Profile in the Scrum Framework.

4.13 Chapter Summary

Towards illustrating the objectives of the study which include to extend the Scrum framework through including a Security Owner and UMLsec, this chapter has introduced the framework for integrating the Security Owner role and UMLsec in the Scrum. The framework is designed to achieve the objective of the study which is to improve the security requirements throughout the modelling and development processes. The framework demonstrated how the Security Owner can assist team members to consider security requirements through their coordinating role which was specified as the first objective of the research. The role of the Security Owner, their responsibility and authority has been discussed. Specifically, the chapter has discussed how the Security Owner deals with security requirements of the proposed system, misuse cases and security story cards in order to prioritise these items in a proposed artefact 'Security Backlog' and integration with the Product Backlog to produce the refined Product Backlog with the Product Owner.

One of the objectives of the study is to include a Security Owner role, and it has been important here to show how that role fits into the processes and functions of the Scrum framework, and where and how the role can facilitate and improve the consideration of security requirements. This chapter has discussed how the misuse cases can be used in Scrum in addition to how the Security Owner can identify misuse cases within the Scrum framework. In reference to the third objective of the study which is to investigate the suitability of UMLsec for modelling security requirements Moreover, it has explained how the analysis of the security requirements for UMLsec is prioritised based on the results of the Scrum team's Security Poker. The chapter presented the Security Owner role to facilitate the work with the security requirements, the Security Backlog as an artefact for security items, and the work of the adoption of UMLsec in agile methods, ending with the proposed integration framework.

As an extension of what has been mentioned in the previous chapter (Chapter 3) about the research methods used (experimentation and questionnaires) to support the proposed adoption of UMLsec and Security Owner role in agile processes, in this research, the next chapter (Chapter 5) in accordance with the fourth objective of the study, will present the experiments to be conducted and the subsequent questionnaires in detail in order to assess the effect of the inclusion of a Security Owner role and UMLsec.

CHAPTER FIVE

EVALUATION EXPERIMENTS

Objectives

- To validate the approach of the study through experimentation
- Illustrating the related Training Course.
- Piloting of experiments
- Training of participants

5. Evaluation Experiments

5.1 Introduction

This chapter presents the experiments which are used to determine the validity of the hypotheses of the study identified in chapter one as follows:

H1. The introduction of a Security Owner role helps the Scrum team in terms of effectiveness when considering and managing security requirements (when using UMLsec)

H2. The Security Owner role helps to identify security requirements

H3. The Security Owner role will help team members with UMLsec in terms of team working

H4. The Security Owner role will help team members to prioritise security requirements using UMLsec

H5. UMLsec is an appropriate tool for modelling security requirements (when facilitated by a Security Owner)

H6. Scrum as an agile process remains agile when UMLsec is used when / if facilitated by a Security Owner

This is achieved through experimentation using a scenario of the Scrum agile development method both with and without the inclusion of the proposed Security Owner role and UMLsec. In order to test these hypotheses two different types of experiment are conducted, one with and one without Security Owner and UMLsec. The reason for two types of experiment is that the experiment which does not include UMLsec and the Security Owner will act as the control experiment. The results of each experiment are derived from questioning participants upon completion about their experience and the results from the two different groups are used to confirm the hypotheses. Results from the experiment with UMLsec and the Security Owner and the results from the control experiment without, through comparison will increase the validity of the experimentation.

To ensure that the participants understood the Scrum procedure that was used in the experiment and the procedure of the experiment itself a training session was session was conducted. This training session included software development and agile development methods, knowledge of modelling using UMLsec.

In order to ensure that the training was effective in informing the participants about the procedure of the experiments and to ensure that the design of the experiment did not include any barriers or misunderstanding or that the participants faced difficulties, a pilot study of the experiments was carried out.

5.1.1 Aims of The Experimentation

The aims of the experimentation are to allow the participants to take part in modelling security requirements, within the Scrum development framework, to determine if the inclusion of UMLsec facilitated by the presence of a Security Owner would have an effect on the identification, consideration, management and modelling of security requirements in an agile method. These aims are towards answering the research questions of this study which include whether or not the introduction of UMLsec and the Security Owner role will extend and improve agile methods, specifically Scrum, in terms of considering and managing security requirements, whether the Security Owner will help Scrum team members in relation to security requirements and whether Scrum as an agile method remains agile. Therefore, the experiments are designed to validate the approach of the study, to include UMLsec and a Security Owner, to improve security requirements consideration and modelling in Scrum towards validating the hypotheses of the study as mentioned in the above.

5.2 Training

Participants took part in a training course prior to the experimentation to ensure that the participants as Scrum team members understand the Scrum development process that will be used in the experiment. Required knowledge was already established in the participant criteria described in Chapter Three as part of the selection process for participation in the experimentation, however, the training is concerned only with explaining the Scrum development process that will take place in the experiments. Specifically, the experiments included an agile and iterative software development process that includes UML modelling techniques for security requirements for the development of object-oriented software. Thus it was necessary through the training to ensure that participants understood this agile development process, specifically Scrum, and how to function within this process, to achieve this the training and experiment included the following:

1. Introducing participants to a Scrum software development process through experiment training.
2. Giving them the opportunity to work in teams to design a secure system.
3. Introducing participants to security and modelling issues, UMLsec, in software engineering.

In such experiments there are several forms of bias that can influence the results. Examples of bias include that the researcher is evaluating the results of the experiment of an approach that the researcher has created (Kitchenham et al. 2002) and is proposing will improve and extend agile methods. Additionally, there is selection bias whereby non-random selection may result in bias and internal validity issues (Kampenes et al., 2009), for the present study this selection bias is reduced because the same selection criteria is used for both experiment groups.

In the experimentation of the present study research bias can be as a result of the experiment group with UMLsec and Security Owner role having knowledge of the group without, this would undermine the control experiment. Therefore, each group will receive training separately so they are not aware of each other. For the group with the Security Owner role and using UMLsec there are two additional stages at the end, namely; UMLsec and adopting in Agile methods and Proposed Scrum Framework as shown in table 5.1. This group will be introduced to how to apply UMLsec into agile and use the new proposed technique. Thus, both groups, i.e. those participating in scenarios with, and those without, Security Owner role and UMLsec received the same training, in different sessions, up to a point.

In addition, the questions in the questionnaire are the same for both groups in order to reduce the bias and increase the validity of the output. Bias was further reduced by ensuring that the labels of the different questionnaire responses and participant's information were anonymised such that the researcher when performing the analysis of the results does not know which result data relates to whom.

5.2.1 Course outline

The following is the course outline for the experiment:

(steps 8 and 9 indicate the additions for the group that include the Security Owner role and UMLsec).

Step	Item	Expected Duration (Minutes)	Specified Group
1	Introduction to Agile methods	5	All Groups
2	Analysis and Design in Agile methods	5	
3	User stories and Use cases	5	
4	Activity 1 – five software projects' example : see the Appendix.	10	
5	Scrum Development Framework and its roles	10	
6	Activity 2 – organising the Scrum team.	5	
7	Misuse cases and Protection Poker	10	
8	UMLsec and adopting in Agile methods	10	The Group with S.O. and UMLsec.
9	Proposed Scrum Framework	10	
Total		70	

Table 5.1: Course outline for the experiments

5.2.2 Description of the course

The course first introduced the students to agile and UML artefacts for object-oriented designs and the proposed Scrum framework, it then linked the idea with designs for agile methods. Thereafter, the students were given a tutorial which included a hands-on introduction to design process. This tutorial was designed to develop the decision making ability of the students as an agile team in terms of whether the provided scenarios can be managed in an agile context.

The researcher started with the modelling of security requirements using UML in the proposed agile Scrum framework. After the 'introduction to agile project management' and before the actual experiments, the researcher introduced the participants to the training course. They presented with the outline as follows:

- Introduction to Agile Software Development and The Agile Manifesto
- Analysis and Design and characteristics of agile methods
- User stories and Use cases as elicitation and requirements engineering techniques
- Plan-driven Vs Agile Developments and when to use agile.

The training included the participants were engaged in a time-boxed activity of ten minutes with five projects' examples. Each example had a two minute negotiation time-box and at the end of each two minute time interval the participants will decide if the example can be dealt with using agile methods. In making this decision, the participants considered the factors high risk with competition, and uncertainty with ambiguous and changing requirements.

After this exercise the participants were introduced to the Scrum development framework and its roles, artefacts and ceremonies. The actual experiments included the participants organising themselves into individual Scrum team members (Product Owner, Scrum Master and development team), this took approximately five minutes. Knowing that, the group which will include a Security Owner will nominate one of its members to be as a Security Owner when they are considering security requirements, applying and modelling the two scenarios described later in the experiment process. Next, the researcher explained misuse cases to the participants from the point of view of the attacker and under what situations they can be used. For this purpose a simple example was drawn on the whiteboard for clarity and illustrative purposes.

After introducing UMLsec, as an extension of UML to include some security requirements and its notation with a small example to illustrate stereotypes, tag values and constraints, the group which included the use of UMLsec used the proposed approach of UMLsec with agile. The motivation and the objectives, the proposed Security Owner role, Security Backlog and some examples to clarify how to deal with the potential security issues were considered.

5.3 Experiment Development - Scenarios

As mentioned above, the training course was conducted to ensure that the participants were familiar to engage and follow the Scrum development process used in the experiments. The next stage in the experimentation involved the participants engaging in a scenario exercise which forms the experiments followed by the post-experiments questionnaires (chapter 3, section 3.5.3) in order to analyse their perceptions and data. Activities and tutorials that given during the course to the participants assist them to proceed with the experimentation as required to work as a Scrum team, model security requirements through the use of UML modelling tool.

During the development of the experiments steps that reflected the concepts of modelling security requirements in Scrum were included within a scenario. The scenarios, described below, were developed and included in both experiment types, i.e. with and without UMLsec and the Security Owner role. The experiment uses the UML features, defined in Chapter Two, of agile software development, namely UMLsec for modelling security requirements in the experiments. The scope of the scenarios includes the initial stages of an actual Scrum development process which include analysis and design of security solutions as part of each sprint.

The following are the two scenarios used to test the effect of the integration of UMLsec and the inclusion of a Security Owner role in agile development methods. Both scenarios include security requirements which the agile team should consider when they engaged to solve the proposed problems and to model these requirements for analysis. Through the experiment the participants' perception of using UMLsec and including the new Security Owner role can be measured through the analysis of the responses to the post-experiment questionnaire. At the end of the training course, the candidate participants were introduced the requirements of the two scenarios problems. All participants in both groups of each experiment conducted know that the security issues which illustrated in the two scenarios should be considered. The researcher gave the group who has Security Owner role and UMLsec the opportunity to use these techniques that were introduced as an extra tools in the training course to use while the other group used to use the standard tools without introducing the new role or UMLsec. Both groups were given equal time to negotiate, consider, model and use their techniques to solve the proposed problems.

5.3.1 Scenarios:

- E-EXAM SECURE SYSTEM (E-ESS)

This scenario was designed to include consideration of secrecy (confidentiality), integrity and access control security requirements because in the experimentation, the researcher need to evaluate the effect of introducing Security Owner role and if the UMLsec is an approperate tool when it used by the Scrum team for considering confidentiality, integrity and access control security requirements in the e-exam system and if the process remains agile when introducing these additions.

In the e-exam secure system (E-ESS), the examiner prepares e-exam test, and it is audited by an external examiner without deleting or updating the original document. The external examiner returns the e-exam test after approval which then be accessed through authorisation by the students when the exam takes place. The primary security criterion of the system is that all the transitions between authorised parties should be secured. The interesting point of this scenario is that to see whether the inclusion of the Security Owner role will help the Scrum team when considering the security issues of these transitions in e-exam system and the effect of using the UMLsec to model the security requirements of the scenario which focus on messeges integrity and access control to the document.

- PATIENT HEALTH RECORD SYSTEM (PHRS)

This scenario was designed to include consideration of authorization, access control and confidentiality security requirements for the same purpose as above in e-exam secure system. This scenario gave the both groups the opportunity to negotiate, consider and manage security requirements in the field of the health care system, and of course the impact of the new additions in such fields. In the PHRS system, the clinician of the department updates patient record based on his informed consent and should notify the patient about any modification; this should be included in the record access control list.

- PHRS security requirements:

This scenario is suitable for participants to consider security requirements in the PHRS in their experiment based on the conditions of the scenario required. The PHRS has several security conditions. Firstly, all Electronic Paitent Records (EPR) should be marked using the accessible control list, which contains names of people who are authorised to read and make alterations to the information. This system is capable of preventing unauthorised people from accessing the control list and the information contained therein. Secondly, clinicians can access the EPR on their own or using patients that can be accessed on the control list. In the event that a patient is referred, he or she can access the record with the clinician who can be accessed on the control list. Thirdly, one clinician on the accessible control list should be marked as responsible. It is the responsibility of such a clinician to make changes on the ways of accessing the control list by increasing the number of healthcare specialists to the list. Furthermore, the clinician who has been marked to be responsible should inform the patient about the names that are available, those that have been added on the control list if there is transfer of responsibility. Additionally, the consent of the patient should be sought; however, in emergency and statutory exemptions. Moreover, without the expiry of the set time, clinical information cannot be deleted by anybody. In addition, any accessibility to the medical records should be marked using the name of the subject and date as well as time. An audit record containing all deleted data should be kept. Finally, effective mechanisms for preventing the acquisition of aggregate information on personal health will be included in PHRS security system. Particularly important would be the decision to inform patients about the proposed persons that are to be added to the list of controlled access. This scenario will validate the ability of the Scrum team, with and without the inclusion of Security Owner role and UMLsec, to consider the security issues of patient record system especially in access control list of the paitent.

5.3.2 Piloting for Validity

The aims of the experiment are to verify if the inclusion of a Security Owner role with UMLsec has a positive impact on the Scrum development process and Scrum team members in terms of considering, identifying and modelling security requirements and whether the Scrum development process remains agile with these inclusions. As mentioned in the above there are two different types of experiment, one with and one without these inclusions for comparative purposes. In order to ensure validity of the experimentation a pilot study for each of the two experiment types was carried out.

<u>Objectives of the Pilot study</u>

Before conducting the main experiment, the author coordinated and started with other participants who met the criteria in the pre-experiments' questionnaire, based on availability, to be as a small agile team to achieve the following objectives:

- Investigating the proposed training course
- Measuring the period of the experiment
- Getting and recording comments and the feedback

Preparing the pilot study:

The researcher finalised the training course presentation after amending its slides based on literature and supervision team comments. He prepared a few copies of the training course and activities descriptions, planning poker deck of cards, some index and coloured cards to use them besides some empty A4 papers. In addition, he booked a meeting room in the library for two hours assuring that it has the necessary facilities such as the presenting screen and the whiteboard.

Conducting the pilot study

The researcher began with the participants with the review of the training course presentation. Firstly, they started with the introduction of agile software development methods. Secondly, they reviewed the comparison between agile and traditional development methods (waterfall models). Thereafter, they introduced some techniques of requirement elicitation and analysis (user stories and use cases).

Then, the agile group test the time-boxed activity one, which list some of software projects examples written by the researcher to see if these examples can be applied using agile development methods or not. Scrum development framework has been introduced and the team organise themselves as a scum team. Due to the few members, depend on the availability, they can only determine the basic roles of the standard Scrum management framework. One of the candidate participants played the proposed Security Owner role. The researcher explained the idea to the audience behind the negotiation between the Product Owner and the Security Owner to produce the refined Product Backlog and the proposed integration framework.

Pilot study outcomes, supervision team comments and participants' feedback:

Based on the participants' discussion in the pilot study and the supervision team comments during meetings, the researcher ending with the following points:

- He should prepare the place of the experiment with all essential facilities (each group may need a flipchart stand with papers and pens).
- He needs to organise two circle groups to facilitate the discussion and communication among themselves in each experiment session.
- The researcher has to clarify agile (Scrum), UML, UMLsec and the integration approach to the audience within the reserved time, taking into account how to control and manage each section with its activity time-box.
- The researcher updated the presentation to be in logical order, the activity one examples to add clues to participants focus on some factors in their decision such as uncertainty, ambiguous and changed requirements, and critical high risk projects.
- Examples simplify the course for the participants in case of doing the scenarios.

5.4 Experiments Procedure

Now that the participants had attended the training course and the experiments were piloted, the next stage was the experimentation itself. The participants were requested to analyse the two aforementioned problem scenarios and then asked to carry out a number of tasks that form part of the Scrum agile software development method. These tasks included the following:

Organising into roles

Initially before the participants started the experiment there was a need for each participant to assume a Scrum team member role. The participants were left to decide who would assume what role by themselves. There are two main reasons for this approach; firstly, to avoid bias through the researcher selecting the participants, and secondly, to avoid ill feeling or disappointment among participants if they are assigned an inappropriate role according to their skills or experience.

Consider the security related problems / issues

The participants are presented with the two scenarios presented in the above and have to identify the security related issues for each scenario. This activity forms the initial part of the examination of whether or not the presence of the Security Owner role will have any effect, positive or otherwise, on this particular activity, specifically it answers the hypothesis that the presence of a Security Owner role helps the Scrum team to consider security issues for a given system.

Two simple scenarios were conducted by two different groups; these formed the experiment to evaluate the role of the Security Owner. One of the groups included a Security Owner role and the other did not, this was for comparison purposes. Each scenario experiment lasted for approximately twenty minutes, detail of the two scenarios can be found in the Appendix. All team members took one of these scenarios (e-exam system) and apply the proposed techniques for example playing the security poker game learned previously. During the scenario experimentation they presented some agile UMLsec diagrams on a flipchart. The end result of the teamwork in the scenario was a refined Product Backlog (all items are prioritised according to importance and return-on-investments, security items are included in this stack as well. The top of the stack is the most important one).

Identify security requirements

The second part in the experiment involves the participants establishing the security requirements after identification of the problems has been carried out. Specifically, there was identification and categorisation of security requirements into areas such as authentication, authorisation, confidentiality, access control and message transfer. This part of the experimentation helps to validate the second hypothesis of the study which states that the presence of a Security Owner role helps the Scrum team to identify security requirements.

Negotiate security requirements – through security poker game to Prioritise security requirements

Once the security requirements have been established they need to be prioritised. The reason for prioritisation is that in any Scrum development there are limited resources in terms of time, cost and technical ability and importantly the Scrum as an agile method has to remain agile, which it cannot do if it considers all security requirements equally. This part of the experimentation validates the hypothesis that the Security Owner role will help team members to prioritise security requirements.

Develop security requirements solution using UML / UMLsec

Now that the security requirements have been established and prioritised, the solutions can be developed using UML modelling. It is important to note that one experiment group will use UML for this modelling and the other group will use UMLsec facilitated by the Security Owner role. Therefore, this part of the experiment will validate the hypothesis that UMLsec is an appropriate tool for modelling security requirements when facilitated by a Security Owner.

Represent security requirements using UML / UMLsec

In order to facilitate discussion and communication about the identification and prioritisation of security requirements and proposed solutions, for each of the two experiment types a circle group was formed. Once the aforementioned tasks were complete and analysis and design artefacts are complete quantitative/qualitative analysis of the participants' experiences was conducted.

Bias

It is important to acknowledge any bias in the experiments. The participants are not told prior to the experimentation whether or not there will be the presence of UMLsec or the Security Owner. In reference to threats to validity, the experiments are conducted within the same experimental conditions, namely; in universities in Saudi Arabia and to reduce this threat to validity the experiments could be conducted in different conditions such as with professionals in the UK.

Post-experiment Questionnaire

To investigate the participants' evaluation of the experiment 'Modelling Security Requirements by UML in Agile Development Methods', a questionnaire was conducted. A half an hour pre-prepared questionnaire guided session was conducted with the participants and their experiences were recorded for detailed analysis. This step evaluates the usefulness and the complexity of our addition and we did it to see if the development is according to the user aspirations.

After discussion, some of them agreed that they became more knowledgeable and have awareness of addressing potential security requirements when engaging IT projects. They referred the reason to the debate occurring while playing security poker cards turns to determine the values of security requirements in addition to the stand-up meeting and informal discussion of these items obstacles. Full results of the participants' experiences are present in the results and analysis chapter (chapter six).

5.5 Chapter Summary

This experimentation is to evaluate if UMLsec is beneficial to use within and extend agile development methods, e.g. Scrum development framework including to what extent could be integrated, in accordance with objective one and whether the inclusion of UMLsec retains the agility of agile methods in accordance to research question four.

The experiments sought to see if the inclusion of a Security Owner role will improve security requirements consideration through identifying and managing security requirements, negotiation with Product Owner and supporting the Scrum team in order to improve the Scrum framework in accordance to objective one of this study. In addition, the experimentation investigated and evaluated the effect of the Security Owner role in terms of emphasizing security issues, coordination and negotiation with other Scrum team roles, this is related to research question two. We do this by adopting the current extension of UMLSec for the proposed Scrum agile method and by finding to what extent this exercise would be beneficial.

This chapter demonstrates the evaluation experiment of our research work, its aims, goals and steps. It has mentioned the sample groups of the needed participants and the environment. In addition, it has discussed the training course for those participants as well.

The next chapter will address and analyse the collected data of these experiments in order to assess/evaluate the new approach of modelling with UMLsec facilitated by the new Security

Owner role and its effectiveness of the security and setting interest derived by the agile team at the level of modeling security requirements.

CHAPTER SIX

DATA ANALYSIS AND DISCUSSION

Objectives

- To present analysis of results from experimentation
- To verify with analysis of results the hypotheses of the study
- To discuss the meaning and implications of the findings

6. Data Analysis and Discussion

6.1 Introduction

This chapter presents the findings of the post-experiment questionnaire that was conducted immediately after each experiment. Data analysis (chapter 3, section 3.8) of the participants' perceptions is to test the hypotheses of this research. Both groups, those with and those without the inclusion of a Security Owner role and UMLsec, were questioned about their experience of engaging in a Scrum team situation and considering security requirements. The questions were designed to reveal if there was an effect on the management of security requirements and team work, also in relation to security requirements. This was required in order to verify the hypotheses of this study to determine the effect of the presence of a Security Owner role and the inclusion of UMLsec on security requirements in Scrum. The results include a comparison between the two different groups and correlation analysis.

Statistical analysis is carried out to check the statistical significant difference in the responses to the statements between the group that includes the Security Owner role and UMLsec and the group that does not. This is carried out using the T-test in SPSS. The T-test shows if there is a statisitcal difference in the means of the two groups towards veryfying the hypothesis of this study. Specifically, it will show if there if there is a statistical significant difference in responses from the group that included a Security Owner and UMLsec and the group that didn't in order to veryfy the hypotheses which check if the inclusion of the Security Owner and UMLsec have an impact in terms of security requirements, teamwork and agility.

6.2 Identifying security requirements

The participants were given a statement to find out if they found it easy to identify security requirements. This statement was designed to test if the presence or absence of a Security Owner (SO) would have an effect on the ease of which security requirements are identified to verify that the SO has a positive effect on this ability.

For the group that included UMLsec and the Security Owner role the majority of the participants agreed and strongly agreed with the idea that they found it easy to identify security requirements, only 2 of the participants disagreed with this idea, however, 10 of the participants were neither agree nor disagree (see Table 6.1). These results support the hypothesis that the Security Owner will help to identify security requirements (H2).

Participants' response	**Frequency**	**Valid percent**	**Cumulative percent**
Strongly disagree	0	0.0	0.0
Disagree	2	3.4	3.4
Neutral	10	17.2	20.6
Agree	17	29.3	50.0
Strongly agree	29	50.0	100.0
Total	58	100.0	

Table 6.1: You found it easy to identify security requirements (with UMLsec and SO)

For those participants from the group without UMLsec and the Security Owner role there was a sharp contrast, there was a high level of disagreement with 33 of the participants disagreeing with the idea that they found it easy to identify security requirements, nine were unsure and only 10 participants agreed with the idea (see Table 6.2). The results here show that there was no agreement with the idea they found it easy to identify security requirements among those without a Security Owner, this further support H2.

Participants' response	**Frequency**	**Valid percent**	**Cumulative percent**
Strongly disagree	13	25	25
Disagree	20	38.5	63.5
Neutral	9	17.3	80.8
Agree	10	19.2	100.0
Strongly agree	0	0.0	100.0
Total	52	100.0	

Table 6.2: You found it easy to identify security requirements (without UMLsec and SO)

		Levene's Test for Equality of Variances		t-test for Equality of Means		
		F	Sig.	t	df	Sig. (2-tailed)
You find it easy to identify security requirements	Equal variances assumed	2.433	.122	10.606	108	.000
	Equal variances not assumed			10.494	99.010	.000

		t-test for Equality of Means			
		Mean Difference	Std. Error Difference	95% Confidence Interval of the Difference	
				Lower	Upper
You find it easy to identify security requirements	Equal variances assumed	1.951	.184	1.586	2.316
	Equal variances not assumed	1.951	.186	1.582	2.320

Table 6.3 Independent Samples Test - You find it easy to identify security requirements

Examining the statistical difference between the two groups in response to this statement, the analysis shows that the null hypothesis which would be that there would be no difference between the results of the two groups, meaning that the presence of a security owner role and UMLsec would have no effect on the ease of identifying security requirements. However, the results show that the null hypothesis is rejected because the independent samples T-test show less than 0.001 at 0.000 (Sig. 2-tailed) (Table 6.3) which shows there is no significant relationship between the two groups. Moreover, there is 95 percent confidence level that the mean difference in agreement with the statement that respondents found it easy to identify security requirements between the ‘with’ and ‘without’ groups was between 1.586 and 2.316 (Table 6.3). Therefore, attitudes to this statement do differ between the two different groups.

The results in the above clearly show that the inclusion of UMLsec and the presence of a Security Owner role have a significant positive impact on the participants in terms of their ability to identify security requirements. The participants clearly indicated that they find it difficult to identify security requirements in the absence of a Security Owner.

The statement ‘you were helped to identify security requirements’ was included to see if the SO as a role would help in identifying security requirements, this statement is different from the previous statement which designed verify the effectiveness of the advice. The inclusion of a SO is designed to improve security requirements consideration in Scrum, part of which is identifying requirements.

The fact that the group that included the Security Owner were more in agreement with the idea that they found it easy to identify security requirements is supported by the responses to the statement that they were helped to identify security requirements where there was a high level of agreement with this idea (see Figure 6.1).

It is important to note that there was a much higher level of neutrality about this idea among those without the Security Owner as well agreement and disagreement across the board, thus overall there was no conclusive result for this group, however, the important result to note is that there was not a strong level of agreement with the idea (see Figure 6.1). One explanation for the high number of neutral responses is that although they may have perceived that they received advice from the SO they may have had doubt about the effectiveness of the advice, this is an issue related to the wording of the question.

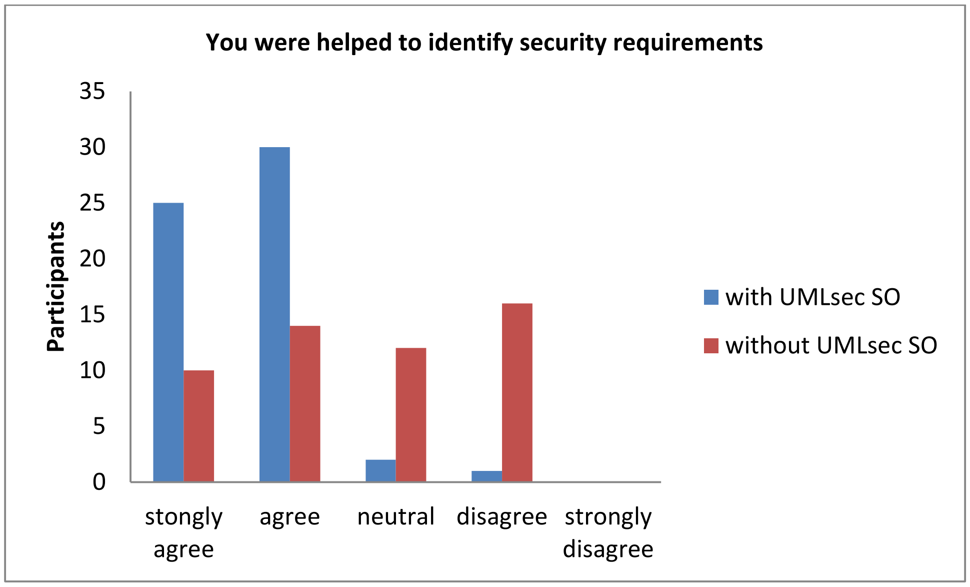


Figure 6.1: You were helped to identify security requirements

Looking at the statistical difference between the two groups in response to the statement ‘You were helped to identify security requirements’, the analysis shows that the null hypothesis which would be that there would be no difference between the results of the two groups, meaning that the presence of a security owner role and UMLsec would not make the respondents feel that they were helped with security requirements. However, the results show that the null hypothesis is rejected because the independent samples T-test show less than 0.001 at 0.000 (Sig. 2-tailed) (Table 6.4) which shows there is no significant relationship between the two groups. Moreover, there is 95 percent confidence level that the mean difference in agreement with the statement that respondents were helped to identify security requirements between the ‘with’ and ‘without’ groups was between 0.676 and 1.356 (Table 6.4). Therefore, attitudes to this statement do differ between the two different groups.

		Levene's Test for Equality of Variances		t-test for Equality of Means		
		F	Sig.	t	df	Sig. (2-tailed)
You were helped to identify security requirements	Equal variances assumed	30.171	.000	5.921	108	.000
	Equal variances not assumed			5.759	79.292	.000

		t-test for Equality of Means			
		Mean Difference	Std. Error Difference	95% Confidence Interval of the Difference	
				Lower	Upper
You were helped to identify security requirements	Equal variances assumed	1.016	.172	.676	1.356
	Equal variances not assumed	1.016	.176	.665	1.367

Table 6.4 Independent Samples Test - You were helped to identify security requirements

Therefore, both the above questions, finding it easy and being helped both in relation to security requirements supports the hypothesis that 'the Security Owner role helps to identify security requirements' (H2) and the hypothesis that 'the introduction of a Security Owner role helps the Scrum team in terms of effectiveness when considering and managing security requirements (when using UMLsec)' (H1).

In reference to H1 part of the effectiveness of a Scrum team when considering and managing security requirements is the identification of those security requirements which is the first part of the process (requirements elicitation). H1 hypothesises whether a Security Owner improves effectiveness of this security requirements consideration and the results show that identification of requirements is improve. Therefore, the results contribute towards supporting the hypothesis.

6.3 Managing security requirements

Another issue related to the effectiveness of a Scrum team is the ability to manage security requirements. The participants were asked if they were able to manage security requirements to see if the inclusion of a SO would have any effect on this ability. In reference to managing security requirements the results also showed a contrast between the two participant groups. For the group with the Security Owner half of the participants agreed with the idea that they were able to manage security requirements and a quarter of them strongly agreed (see Table 6.5)

Participants' response	Frequency	Valid percent	Cumulative percent
Strongly disagree	0	0.0	0.0
Disagree	4	6.9	6.9
Neutral	10	17.2	24.1
Agree	29	50.0	74.1
Strongly agree	15	25.9	100.0
Total	58	100.0	

Table 6.5: You were able to manage security requirements effectively (with UMLsec and SO)

This was in contrast to the group without the Security Owner and UMLsec, where there was a much level of disagreement where 47.2 percent of participants disagreed with this idea. This sharp contrast in the results is a clear indication that the presence of a SO had a strong positive effect on the management of security requirements. However, although this statement may show a strong influence from the presence of a SO, it does not show how this may occur. The following statements include about receiving effective advice about managing security requirements and finding it easy to prioritise security requirements, and they are designed to further elaborate of the general idea of effectively managing security requirements. In the following the results of these statements are presented.

6.3.1 Advised about managing security requirements

As discussed in the above, the statement here is about receiving advice about managing security requirements which is more direct and reflective of the actions of the SO. Within the agile development process team members are engaged in the practical implementation of security requirements as well as considering security requirements both in the sprints and the sprint meetings. Therefore, it is important for team members to receive advice from a member who has more ability and focus of security than themselves. The statement was presented to see if the participants would perceive that they receive more advise in the presence of the SO. For the group with the Security Owner and UMLsec there was a much stronger level of agreement with this idea than the group without (see Tables 6.6, 6.7). Therefore, with the prescence of a Security Owner team members felt that they received more advice about security requirements management. Although the statement does not specify who the advice comes from, the very presence of the SO increases agreement with the idea that advice is received, so whether the advice comes from the SO or another team member, the overall effect of the presence of the SO is positive in this sense.

Participants' response	**Frequency**	**Valid percent**	**Cumulative percent**
Strongly disagree	0	0.0	0.0
Disagree	6	10.3	10.3
Neutral	0	0.0	0.0
Agree	30	51.7	51.7
Strongly agree	22	38	100.0
Total	58	100.0	

Table 6.6: You received effective advice about security requirements to help with security requirements management (with SO and UMLsec)

Participants' response	**Frequency**	**Valid percent**	**Cumulative percent**
Strongly disagree	12	23.1	23.1
Disagree	14	26.9	50.0
Neutral	8	15.4	65.4
Agree	18	34.6	100.0
Strongly agree	0	0.0	100.0
Total	52	100.0	

Table 6.7: You received effective advice about security requirements to help with security requirements management (without SO and UMLsec)

6.3.2 Prioritising security requirements

Additionally, within the overall management of security requirements is the ability to prioritise security requirements , this is required within the Scrum framework as part of planning the sprints. Moreover, considering this prioritisation further elaborates on where the SO can be effective in terms of management of security requirements. If reference to the statement about whether or not the participants found it easy to prioritise security requirements there was a level of agreement with this idea for the group with the Security Owner and UMLsec (32.8 percent strongly agree and 46.6 percent agree), although some of the participants expressed neutrality about this idea (10.3 percent). As for the group without, there was more disagreement with this idea (see Tables 6.8, 6.9).

Participants' response	**Frequency**	**Valid percent**	**Cumulative percent**
Strongly disagree	0	0.0	0.0
Disagree	6	10.3	10.3
Neutral	6	10.3	20.6
Agree	27	46.6	67.2
Strongly agree	19	32.8	100.0
Total	58	100.0	

Table 6.8: You found it easy to prioritise security requirements (with UMLsec and SO)

Participants' response	**Frequency**	**Valid percent**	**Cumulative percent**
Strongly disagree	6	11.5	11.5
Disagree	26	50.0	61.5
Neutral	10	19.2	80.8
Agree	10	19.2	100.0
Strongly agree	0	0	100.0
Total	52	100.0	

Table 6.9: You found it easy to prioritise security requirements (without UMLsec and SO)

The higher level of agreement with this idea shows that the inclusion of a Security Owner role and UMLsec makes it is easier for the Scrum team members to prioritise security requirements. Therefore, the results show that the fourth hypothesis (H4) is supported, that the Security Owner will help the team members to prioritise security requirements. The fact that without the SO and UMLsec the participants did not find it easy to prioritise security requirements further supports the hypothesis. Again, although there is no clear evidence that the SO directly helped in prioritising security requirements, their presence has a positve effect.

		Levene's Test for Equality of Variances		t-test for Equality of Means		
		F	Sig.	t	df	Sig. (2-tailed)
You find it easy to prioritise security requirements	Equal variances assumed	1.938	.167	8.736	108	.000
	Equal variances not assumed			8.730	106.401	.000

		t-test for Equality of Means			
		Mean Difference	Std. Error Difference	95% Confidence Interval of the Difference	
				Lower	Upper
You find it easy to prioritise security requirements	Equal variances assumed	1.556	.178	1.203	1.909
	Equal variances not assumed	1.556	.178	1.202	1.909

Table 6.10 Independent Samples Test - You find it easy to prioritise security requirements

The statistical difference between the two groups in response to the statement ‘You find it easy to prioritise security requirements’, the analysis shows that the null hypothesis which would be that there would be no difference between the results of the two groups, meaning that the presence of a security owner role and UMLsec would not make the respondents feel that they found it easy to prioritise security requirements. However, the results show that the null hypothesis is rejected because the independent samples T-test show less than 0.001 at 0.000 (Sig. 2-tailed) (Table 6.10) which shows there is no significant relationship between the two groups. Moreover, there is 95 percent confidence level that the mean difference in agreement with the statement that respondents found it easy to prioritise security requirements between the ‘with’ and ‘without’ groups was between 1.203 and 1.909 (Table 6.10). Therefore, attitudes to this statement do differ between the two different groups.

The responses to the qualitative questions in the questionnaire for those with a Security Owner showed that particpants were content with the approach that they used. Specifically in response to question 19 which read as follows ‘*In consideration of the approach to security requirements that you engaged with in the experiment, what are your opinions about the security requirements consideration?*’ participants emphasised that the consideration of security requirements was something that they did collectively as the following statement shows:

> *We work together to consider security requirements as we need input from all the members (with UMLsec and SO)*

Although they did not mention the Security Owner directly, they did say that security requirements consideration was a process that went smoothly.

> *I find it easy to consider security requirements as we discussed them collectively and helped each other (UMLsec and SO)*

Ther was also mention of the use of UMLsec:

> *Modelling the security requirements using UMLsec was completely new to me but I was fine with it (UMLsec and SO)*

The group that did not include the Security Owner were also positive about security requirements consideration, however, they did not mention anything about teamwork or the ease of the process unlike the other group. Therefore, the results here show that the presence of a Security Owner had a postive effect on teamwork and the ease of the process of considering security requirements consideration.

6.4 Modelling security requirements

One of the hypothesis of the study is that the introduction of UMLsec will improve the modelling of security requirements with the facilitation of the Security Owner (H5). This section will present the results for the questions that were designed to verify this hypothesis. The first statement that is dealt with is ‘You required help with the technique you used for modelling security requirements’, this statement was designed to firstly, see if the participants needed help with UMLsec and secondly, if the SO had any influence in this.

For the group that included the Security Owner and UMLsec, this statement was designed to see if the Security Owner was instrumental in helping to use UMLsec in modelling requirements. The results show that most, 45 out of the 58 participants that used UMLsec agree with the idea that they need help with this modelling technique. For the group with the Security Owner and UMLsec this could suggest two main ideas, firstly, that UMLsec is new and that participants needed help, or secondly that the SO was not forthcoming with help in facilitating the use of UMLsec.

Participants' response	**Frequency**	**Valid percent**	**Cumulative percent**
Strongly disagree	0	0.0	0.0
Disagree	3	5.2	5.2
Neutral	10	17.2	22.4
Agree	31	53.4	75.9
Strongly agree	14	24.1	100.0
Total	58	100.0	

Table 6.11: You required help with the technique you used for modelling security requirements (with UMLsec and SO)

As for the group that did not use UMLsec (Group 2) most of them disagreed with the idea that they needed help, however, there was a significant number, 19, who said that they did need help (see Table 6.12). Again this further supports the idea that the introduction of UMLsec means that participants will need help with this new development tool, which the study proposes can be facilitated by a Security Owner.

Participants' response	**Frequency**	**Valid percent**	**Cumulative percent**
Strongly disagree	10	19.2	19.2
Disagree	14	26.9	46.2
Neutral	9	17.3	63.5
Agree	19	36.5	100.0
Strongly agree	0	0.0	100.0
Total	52	100.0	

Table 6.12: You required help with the technique you used for modelling security requirements (without UMLsec and SO)

		Levene's Test for Equality of Variances		t-test for Equality of Means		
		F	Sig.	t	df	Sig. (2-tailed)
You required help with the technique you used for modelling security requirements	Equal variances assumed	23.967	.000	6.672	108	.000
	Equal variances not assumed			6.540	88.799	.000

		t-test for Equality of Means			
				95% Confidence Interval of the Difference	
		Mean Difference	Std. Error Difference	Lower	Upper
You required help with the technique you used for modelling security requirements	Equal variances assumed	1.254	.188	.881	1.627
	Equal variances not assumed	1.254	.192	.873	1.635

Table 6.13 Independent Samples Test - You required help with the technique you used for modelling security requirements

The statistical analysis shows that the statistical difference between the two groups in response to the statement ‘You required help with the technique you used for modelling security requirements’, the analysis reveal that the null hypothesis which would be that there would be no difference between the results of the two groups, meaning that the presence of a security owner role and UMLsec would not make the respondents feel that they needed help with the technique they used for modelling security requirements. The results show that the null hypothesis is rejected because the independent samples T-test show less than 0.001 at 0.000 (Sig. 2-tailed) (Table 6.13) which shows there is no significant relationship between the two groups. Moreover, there is 95 percent confidence level that the mean difference in agreement with the statement that respondents found it easy to prioritise security requirements between the ‘with’ and ‘without’ groups was between 0.881 and 1.627 (Table 6.13). Thus, the attitudes to this statement do differ between the two different groups.

6.4.1 Model security requirements and methods used for modelling security requirements

The study proposes the introduction of UMLsec as part of extending and improving Scrum in terms of security requirements. One statement ‘you found it easy to model security requirements’ was designed to see if UMLsec was easy to use if faciltated by a Security Owner. The second statement ‘the methods you used for modelling security requirements were appropriate’ was designed to see if the participants approved of UMLsec. Here a correlation between these two ideas is established to see if there is a link between an appropriate method for modelling and ease of modelling. For the participants from the group that included UMLsec and the Security Owner role there was a close correlation between the ideas that they found it is easy to model security requirements and that the methods used for modelling these requirements were appropriate (see Table 6.14) indicating that the methods were appropriate and easy to use. This group included the use of UMLsec as a security requirements modelling tool.

		You found it easy to model security requirements	The methods you used for modelling security requirements were appropriate
You found it easy to model security requirements	Pearson Correlation	1	.885**
	Sig. (2-tailed)		.000
	N	58	58
The methods you used for modelling security requirements were appropriate	Pearson Correlation	.885**	1
	Sig. (2-tailed)	.000	
	N	58	58

**. Correlation is significant at the 0.01 level (2-tailed).

Table 6.14: Correlation between easy to model security requirements and methods used were appropriate

For the participants without UMLsec and a Security Owner role there was also a significant correlation between these two ideas, note that this correlation refers to a disagreement with these ideas. Therefore, with UMLsec and a Security Owner role where participants found it is easy to model security requirements, they also agreed that the tool for doing so, i.e. UMLsec was appropriate. For those who did not have UMLsec and the Security Owner role where they did not find it easy to model security requirements they also agreed that the tool they were using, which in this case did not include UMLsec, was inappropriate. Overall, therefore, the Security Owner role and UMLsec are very helpful in modelling security requirements.

These results clearly show that the hypothesis number 5 is supported, that 'UMLsec is an appropriate tool for modelling security requirements (when facilitated by a Security Owner)' (H5).

This idea is supported by the responses to these two statements by the group including UMLsec and Security Owner (see Table 6.15) and the group without (see Table 6.16). There was a stronger agreement with the idea that it was easy to model security requirements where UMLsec was used and a Security Owner was present, this was in contrast to those without UMLsec and a Security Owner, most of whom disagreed, however, there were 10 participants who agreed.

Participants' response	Frequency	Valid percent	Cumulative percent
Strongly disagree	0	0.0	0.0
Disagree	8	13.8	13.8
Neutral	8	13.8	27.6
Agree	28	48.3	75.9
Strongly agree	14	24.1	100.0
Total	58	100.0	

Table 6.15: You found it easy to model security requirements (with UMLsec and SO)

Participants' response	**Frequency**	**Valid percent**	**Cumulative percent**
Strongly disagree	12	23.1	23.1
Disagree	21	40.4	63.5
Neutral	9	17.3	80.8
Agree	10	19.2	100.0
Strongly agree	0	0.0	100.0
Total	52	100.0	

Table 6.16: You found it easy to model security requirements (without UMLsec and SO)

		Levene's Test for Equality of Variances		t-test for Equality of Means		
		F	Sig.	t	df	Sig. (2-tailed)
You find it easy to model security requirements	Equal variances assumed	1.676	.198	7.868	108	.000
	Equal variances not assumed			7.831	104.060	.000

		t-test for Equality of Means			
				95% Confidence Interval of the Difference	
		Mean Difference	Std. Error Difference	Lower	Upper
You find it easy to model security requirements	Equal variances assumed	1.501	.191	1.123	1.879
	Equal variances not assumed	1.501	.192	1.121	1.881

Table 6.17 Independent Samples Test - You find it easy to model security requirements

The statistical difference between the two groups in response to the statement ‘You find it easy to model security requirements’, shows that the null hypothesis which would be that there would be no difference between the results of the two groups, meaning that the presence of a security owner role and UMLsec would not make the respondents feel that they found it easy to model security requirements. The results show that the null hypothesis is rejected because the independent samples T-test show less than 0.001 at 0.000 (Sig. 2-tailed) (Table 6.17) which shows there is no significant relationship between the two groups. Moreover, there is 95 percent confidence level that the mean difference in agreement with the statement that respondents found it easy to prioritise security requirements between the ‘with’ and ‘without’ groups was between 1.123 and 1.879 (Table 6.17). Thus, the attitudes to this statement do differ between the two different groups.

In reference to the statement that the methods used for modelling security requirements were appropriate, there was general overall agreement with this idea for those who used UMLsec and had a Security Owner and general disagreement from those who did not (see Figure 6.2). Overall, these results suggest that UMLsec was easy to use and the participants felt that it was an appropriate tool for modelling security requirements.

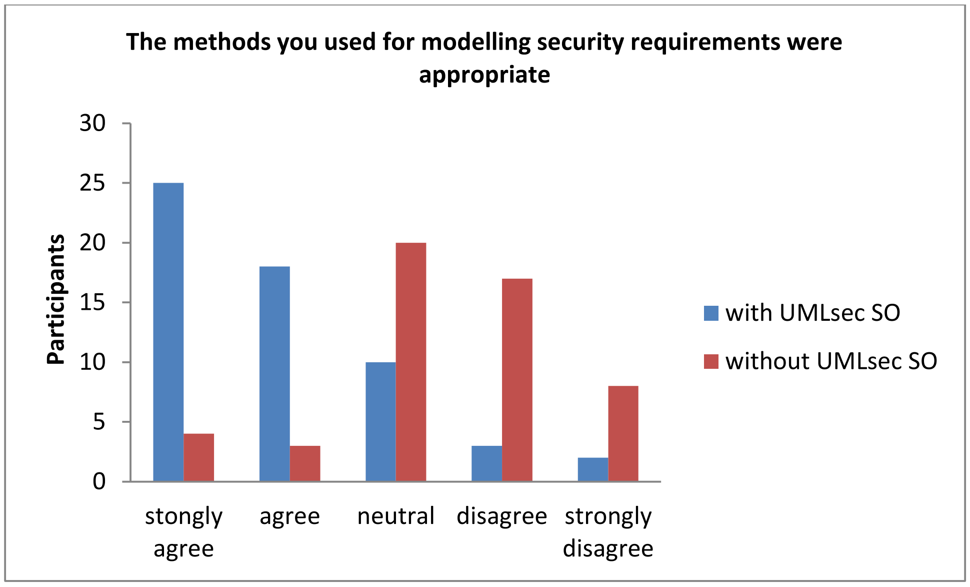


Figure 6.2: Method used for modelling were appropriate

6.4.2 You were helped with modelling security requirements

In this specific reference to the inclusion of a Security Owner role the participants were asked if they were helped with modelling security requirements. This statement was designed to determine the effect that the Security Owner played in modelling security requirements, this was towards verifying the idea that the introduction of a Security Owner role would extend and improve Scrum in relation to security requirements. Moreover, in case of the group that included the Security Owner and UMLsec, the statement would reveal if the SO helped in faciltating UMLsec. None of the participants from the experiment that included UMLsec and the Security Owner disagreed with the idea that they were helped to model security requirements and only six of the participants were neutral (see Table 6.18). not only does this support that the SO facilitates UMLsec in modelling security requirements, but also disproves the aforemention idea (section 6.4) that the SO was not forthcoming in faciltating UMLsec and the only reason the participants felt they needed help was because UMLsec was a unfamiliar technique.

Participants' response	**Frequency**	**Valid percent**	**Cumulative percent**
Strongly disagree	0	0.0	0.0
Disagree	0	0.0	0.0
Neutral	6	10.3	10.3
Agree	37	63.8	74.1
Strongly agree	15	25.9	100.0
Total	58	100.0	

Table 6.18: You were helped with modelling security requirements (with UMLsec and SO)

However, with the group that did not include the Security Owner and UMLsec there was a significant level of disagreement with this idea of help to model security requirements (see Table 6.19). Therefore this shows that the inclusion of a Security Owner contributes to participants feeling that they are being helped with modelling security requirements, and a significant number were unsure. This helps to support the hypothesis that a Security Owner will help the team to effectively manage security requirements (H1).

Participants' response	**Frequency**	**Valid percent**	**Cumulative percent**
Strongly disagree	0	0.0	0.0
Disagree	16	30.8	30.8
Neutral	14	26.9	57.7
Agree	20	38.5	96.2
Strongly agree	2	3.8	100.0
Total	52	100.0	

Table 6.19: You were helped with modelling security requirements (without UMLsec and SO)

6.5 Teamwork

Teamwork was an important consideration in the study. The structure and function of Scrum involves team members working together towards a goal. Moreover, the study proposes the introduction of a new team member, the Security Owner, that would improve Scrum and help team members improve security requirements consideration. Therefore, it was important to verify if team mebers felt that there was effective teamwork regarding security requirements which is addressed by the statement 'there was effective team work regarding security requirements in the Scrum meetings' was used to verify the hypothesis that a Security Owner will help team members in terms of team working when they model security requirements (H3).

In reference to the hypothesis (H3) the participants were asked if there was effective teamwork regarding security requirements in the Scrum meetings. For the team that included the Security Owner and UMLsec most of the participants agreed with the idea that there was effective teamwork regarding security requirements during the meetings, and the majority of those who agreed strongly agreed (see Table 6.20). There are two elements that should be considered in this statement, firstly, that there is effective teamwork, and secondly, this teamwork is in relation to security requirements. Where there is a high level of agreement with the idea it shows that the Security Owner is the person having a positive effect on teamworking because it is related to security requirements.

Participants' response	Frequency	Valid percent	Cumulative percent
Strongly disagree	0	0.0	0.0
Disagree	2	3.4	3.4
Neutral	3	5.2	8.6
Agree	23	39.7	48.3
Strongly agree	30	51.7	100.0
Total	58	100.0	

Table 6.20: There was effective team work regarding security requirements in the Scrum meetings (with UMLsec and SO)

For those that did not have any Security Owner and UMLsec there was a level of disagreement with this idea with only 15 participants agreeing, although 16 of the participants were neutral as opposed to none of the participants from the other group being neutral. This shows that with the absence of a Security Owner there is less enthusiam to agree with the idea that there is teamwork, despite the fact that 15 participants agreed.

Participants' response	Frequency	Valid percent	Cumulative percent
Strongly disagree	3	5.8	5.8
Disagree	18	34.6	40.4
Neutral	16	30.8	71.2
Agree	15	28.8	100.0
Strongly agree	0	0.0	100.0
Total	52	100.0	

Table 6.21: There was effective team work regarding security requirements in the Scrum meetings (without UMLsec and SO)

The contrast in the results shown here is a clear indication in that the presence of a Security Owner has a positive effect on both effective teamwork and the consideration of security requirements.

The statistical difference between the two groups in response to the statement ‘There is effective team work regarding security requirements in the Scrum meetings’, the analysis shows that the null hypothesis which would be that there would be no difference between the results of the two groups, meaning that the presence of a security owner role and UMLsec would not make the respondents feel that there was effective team work regarding security requirements in the Scrum meetings. The results show that the null hypothesis is rejected because the independent samples T-test show less than 0.001 at 0.000 (Sig. 2-tailed) (Table 6.22) which shows there is no significant relationship between the two groups. Moreover, there is 95 percent confidence level that the mean difference in agreement with the statement that respondents found it easy to prioritise security requirements between the ‘with’ and ‘without’ groups was between 1.153 and 1.886 (Table 6.22). Thus, the attitudes to this statement do differ between the two different groups.

		Levene's Test for Equality of Variances		t-test for Equality of Means		
		F	Sig.	t	df	Sig. (2-tailed)
There is effective team work regarding security requirements in the Scrum meetings	Equal variances assumed	3.598	.061	9.842	108	.000
	Equal variances not assumed			9.730	98.253	.000

		t-test for Equality of Means			
		Mean Difference	Std. Error Difference	95% Confidence Interval of the Difference	
				Lower	Upper
There is effective team work regarding security requirements in the Scrum meetings	Equal variances assumed	1.570	.159	1.253	1.886
	Equal variances not assumed	1.570	.161	1.250	1.890

Table 6.22 Independent Samples Test - There is effective team work regarding security requirements in the Scrum meetings

6.5.1 Team identifying

In support of the above ideas that the team was effective regarding security requirements in the Scrum with the presence of a Security Owner, there was also a high level of agreement with the idea that the team effectively identified security requirements where a Security Owner was present (Figure 6.3). The idea behind the statement ‘the team effectively identified security requirements’ was to see if the Security Owner had a positive effect on a specific aspect of security requirements. Without a Security Owner the agile team will identify security requirements as a team, however, that would be without a coordinating and overseeing function of a Security Owner and the statement here is designed to see if the prescence of a new security role would increase the teams effectiveness in identifying security requirements. For the group that included the Security Owner and UMLsec there was a strong level of agreement with this idea in contrast to a much weaker level of agreement with this idea for those without the Security Owner, for which there was a stronger level of disagreement that the team effectively identified security requirements (see Figure 6.3).

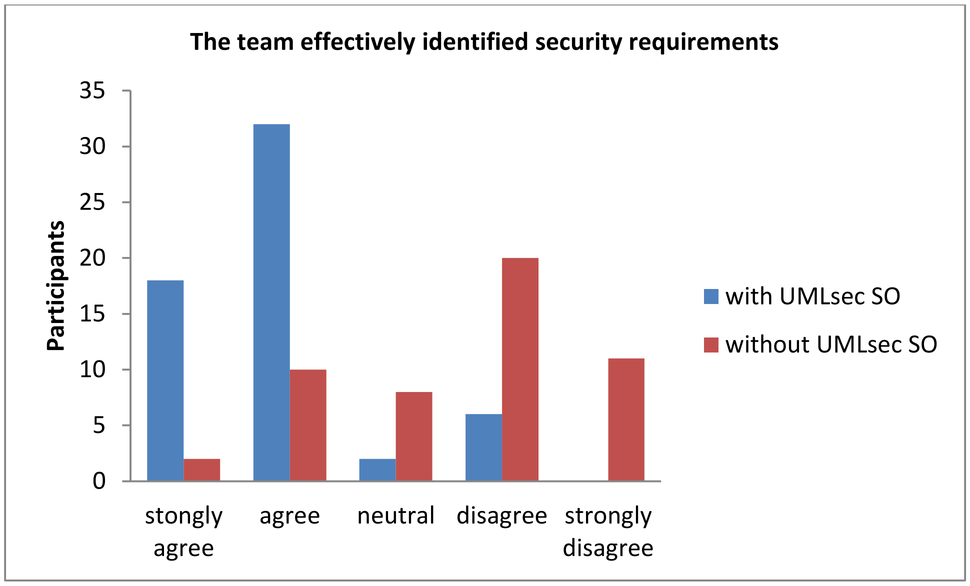


Figure 6.3: Effectively identifying security requirements

The results here show that a Security Owner has a positive effect on the ability of the team to identify security requirements.

6.5.2 Team and managing security requirements

Whether the inclusion of a Security Owner and UMLsec has an effect on the management of security requirements is also consideration of the study. Here the statement is designed to see if the team can manage security requirements as a team. In reference to the management of security requirements as a team there was a strong level of agreement with the idea that the team could manage security requirements without assistance if there was the presence of a Security Owner and the inclusion of UMLsec (see Table 6.23). This supports the hypotheses H1 and H4.

Participants' response	**Frequency**	**Valid percent**	**Cumulative percent**
Strongly disagree	0	0.0	0.0
Disagree	10	17.2	17.2
Neutral	6	10.3	27.6
Agree	24	41.4	69.0
Strongly agree	18	31.0	100.0
Total	58	100.0	

Table 6.23: As a team you could manage your own security requirements / considerations without assistance (with UMLsec and SO)

This was in contrast to the team without the Security Owner and UMLsec, the majority of which disagreed with the idea, however, there were a significant number, 8, did agree (see Table 6.24). The level of neutrality from both of the groups could be due to the fact that the question mentions without assistance and they have intepreted that as without any assistance including the Security Owner.

Participants' response	**Frequency**	**Valid percent**	**Cumulative percent**
Strongly disagree	2	3.8	3.8
Disagree	31	59.6	63.4
Neutral	11	21.2	84.6
Agree	8	15.4	100.0
Strongly agree	0	0.0	100.0
Total	52	100.0	

Table 6.24: As a team you could manage your own security requirements / considerations without assistance (without UMLsec and SO)

The results of the qualitative question also showed that the participants that included the Security Owner and UMLsec were generally positive about the idea that they work well as a team and that they were effective in modelling security requirements.

The approach I used made it easy to model the security requirements (with UMLsec and SO)

And

For the whole process which included considering and modelling the security the approach was suitable (with UMLsec and SO)

6.5.3 Implementing security into the product

In addition to considering, prioritising and modelling security requirements, within the Scrum development process it is also important that the modelled security is implemented into the product. Both of the groups were more in agreement with this idea than were in disagreement, there was also no significant difference in the uncertainty of these two groups (see Tables 6.25, 6.26).

Participants' response	Frequency	Valid percent	Cumulative percent
Strongly disagree	0	0.0	0.0
Disagree	0	0.0	0.0
Neutral	4	6.9	6.9
Agree	39	67.2	74.1
Strongly agree	15	25.9	100.0
Total	58	100.0	

Table 6.25: As a team you could effectively implement security into the product (with UMLsec and SO)

Participants' response	**Frequency**	**Valid percent**	**Cumulative percent**
Strongly disagree	0	0.0	0.0
Disagree	3	5.8	5.8
Neutral	3	5.8	11.5
Agree	30	57.7	69.2
Strongly agree	16	30.8	100.0
Total	52	100.0	

Table 6.26: As a team you could effectively implement security into the product (without UMLsec and SO)

The results here would suggest that either the Security Owner was not effective in helping the team in implmenting security requirements, or that the implemnetation of security requirements is something that is carried out by individuals and does not involve the Security Owner.

6.5.4 Team communication

The nature of a Scrum agile development process is that members regularly meet and discuss development, therefore, communication between team members is important. Moreover, in the consideration of security requirements there also needs to be effective communication between team members. The introduction of a Security Owner should help to highlight security issues and facilitate their discussion and consideration between team members. Therefore, the statement 'there was effective communication about security requirements in the Scrum team' was designed to verify this idea in the presence and absence of a Security Owner.

Evidence that the presence of a Security Owner and UMLsec have a positive impact on the consideration of security requirements is that the presence of a Security Owner is correlated with a high level of agreement with the idea that there is effective communication in the team about security requirements. Most of the participants that included the Security Owner agreed with this idea (see Table 6.27).

Participants' response	**Frequency**	**Valid percent**	**Cumulative percent**
Strongly disagree	0	0.0	0.0
Disagree	9	15.5	15.5
Neutral	9	15.5	31.0
Agree	25	43.1	74.1
Strongly agree	15	25.9	100.0
Total	58	100.0	

Table 6.27 There was effective communication about security requirements in the Scrum team (with UMLsec and SO)

On the other hand, those who did not have a Security Owner there was a higher level of disagreement with the idea that there was effective communication about security requirements within the team (see table 6.28), although there were a number of participants who were unsure about this idea. Therefore, the results here support the hypothesis that (H3) that a Security Owner role would improve teamwork. However, the hypothesis is not supported strongly because a significant number of the participants without UMLsec and the SO, 38.5 percent, also agreed with the statement (see Table 6.28).

Participants’ response	Frequency	Valid percent	Cumulative percent
Strongly disagree	6	11.5	11.5
Disagree	16	30.8	42.3
Neutral	10	19.2	61.5
Agree	20	38.5	100.0
Strongly agree	0	0.0	100.0
Total	52	100.0	

Table 6.28: There was effective communication about security requirements in the Scrum team (without UMLsec and SO)

The research was also interested in the idea that the presence of a Security Owner would improve consideration of security requirements as a team within the Scrum framework. The statement that the participants found that themselves as individuals and the team as a whole could effectively negotiate security requirements with the Product Owner was designed to determine if the inclusion of a Security Owner would improve the negotiation of security requirements.

For the participants who took part in the experiment without UMLsec and a Security Owner role there was a level of disagreement with this idea, however, the disagreement was not strong and a significant number of the participants were unsure about this idea, and the majority agreed (see Table 6.28).

Currently in a Scrum development situation the team members will negotiate with the Product Owner about security requirements as the Product Owner is the one who will bring the customer’s security requirements to the team. The Security Owner will help the team to negotiate security requirements with the product owner and help them both prioritise requirements, this is because the Security Owner can help with security requirements. The statement in the following (see Table

6.29) is designed to see if the Security Owner is effective in this way. For the group without the Security Owner and UMLsec 44.2 percent agreed with the statement (see Table 6.29) and for the group with the Security Owner 70.7 percent agreed (see Table 6.30). The results show that although there were a significant number of people from both groups that agreed with the idea, the group with the Security Owner agreed more and more strongly agreed suggesting that the Security Owner had some positive impact in this regard. These results support H3.

Participants' response	**Frequency**	**Valid percent**	**Cumulative percent**
Strongly disagree	0	0.0	0.0
Disagree	10	19.2	19.2
Neutral	19	36.5	55.8
Agree	19	36.5	92.3
Strongly agree	4	7.7	100.0
Total	52	100.0	

Table 6.29: You found that you and the team could effectively negotiate security requirements with the Product Owner (without UMLsec and SO)

Participants' response	**Frequency**	**Valid percent**	**Cumulative percent**
Strongly disagree	0	0.0	0.0
Disagree	9	15.5	15.5
Neutral	8	13.8	29.3
Agree	32	55.2	84.5
Strongly agree	9	15.5	100.0
Total	58	100.0	

Table 6.30: You found that you and the team could effectively negotiate security requirements with the Product Owner (with UMLsec and SO)

6.6 Prioritising security requirements

Not only was it important to determine the effect of the Security Owner and the inclusion of UMLsec for the team members, but because the Product Owner is the one responsible for the product it was important to see if the presence of a Security Owner helped the Product Owner effectively prioritised security requirements. The idea here is different from the idea in section 6.5.4 where the emphasis was on the team and communication, here the emphasis is on whether the Security Owner would have a positive impact in terms of the Product Owner prioritising security requirements. It was important that the level of agreement with this statement did not come from the product owner alone but from all team members in order to reduce bias. Asking a Product Owner about their effectiveness would certainly lead to a bias answer. For the participants in the experiment with the Security Owner and UMLsec there was a high level of agreement with the idea that the Product Owner effectively prioritised security requirements, (see Table 6.31) this takes place after negotiation with Security Owner. This was in strong contrast to those participants who took part in the experiment without the Security Owner whereby a significant number, 35, out of 52 disagreed with this idea, however a significant number, 7, were neutral about this idea and 10 agreed (see Table 6.32)

Participants' response	**Frequency**	**Valid percent**	**Cumulative percent**
Strongly disagree	0	0.0	0.0
Disagree	0	0.0	0.0
Neutral	8	13.8	13.8
Agree	36	62.1	75.9
Strongly agree	14	24.1	100.0
Total	58	100.0	

Table 6.31: The Product Owner effectively prioritised security requirements (with UMLsec and SO)

Participants' response	**Frequency**	**Valid percent**	**Cumulative percent**
Strongly disagree	10	19.2	19.2
Disagree	25	48.1	67.3
Neutral	7	13.5	80.8
Agree	10	19.2	100.0
Strongly agree	0	0.0	100.0
Total	52	100.0	

Table 6.32: The Product Owner effectively prioritised security requirements (without UMLsec and SO)

The higher level of agreement with this idea could be due to the fact that the Security Owner helps the Product Owner with security requirements because they have the ability to help, and therefore, where the Security Owner was present there was a higher perception that the Product Owner could effectively prioritise security requirements.

6.7 Agility

Because Scrum is an agile development process it was important that of the inclusion of the Security Owner and UMLsec did not have a detrimental impact on agility. The teams of participants were asked whether they thought that the agility was maintained throughout the agile process., this was achieved through the statement 'the agility of the team was maintained in the agile process'. The results showed that both teams agreed with this idea and there was a low level of disagreement. The difference between the level of agreement of the two groups was not significant (see Figure 6.4).

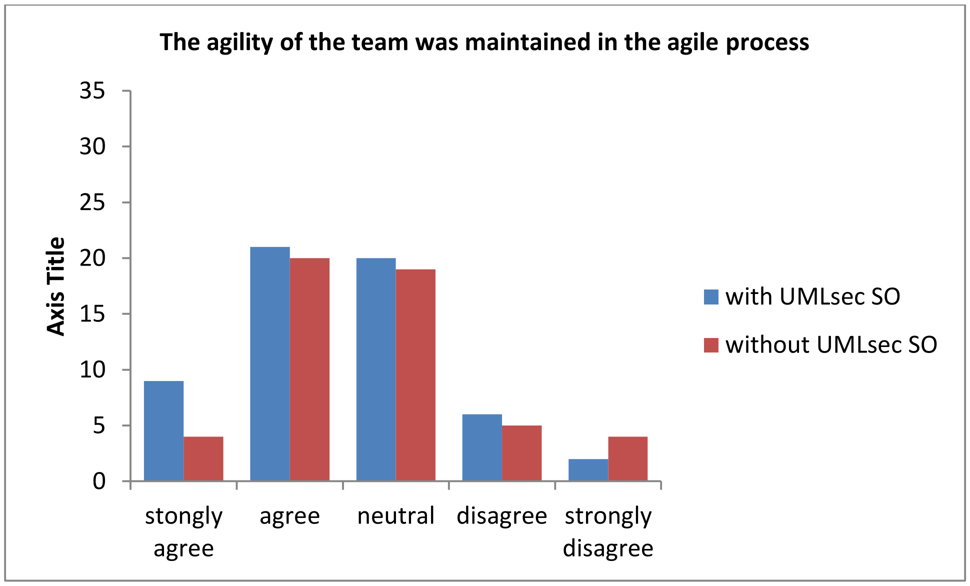


Figure 6.4: Agility of the team maintained in the agile process

A conclusion that can be drawn from these results is that the inclusion of a Security Owner role and UMLsec do not have a negative impact on the agility of the process. It was important to determine this because the benefits that UMLsec and the Security Owner bring can never outweigh the fundamental principles of Scrum that it is an agile development process.

Therefore, in reference the hypothesis 6 (H6), that ‘Scrum as an agile process remains agile when UMLsec is used when / if facilitated by a Security Owner’ the results here show that this hypothesis is not confirmed because participants from both groups agree with this idea. However, it is important to note that a significant number were unsure, but that was for both groups so it does not say anything about the inclusion or exclusion of the Security Owner with respect to having an impact on agility.

The statistical difference between the two groups in response to the statement ‘The agility of the team was maintained in the agile process’, the analysis shows that the null hypothesis which would be that there would be no difference between the results of the two groups, meaning that the presence of a security owner role and UMLsec would not make the respondents feel that the agility was not maintained due to the presence of UMLsec and a Security Owner. The results show that the null hypothesis is not rejected because the independent samples T-test show more than 0.001 at 0.314 (Sig. 2-tailed) (Table 6.33) which shows there is no significant relationship between the two groups. Moreover, there is 95 percent confidence level that the mean difference in agreement with the statement that respondents found it easy to prioritise security requirements between the ‘with’ and ‘without’ groups was between -0.186 and 0.575 (Table 6.33). Thus, the attitudes to this statement do not significantly differ between the two different groups.

		Levene's Test for Equality of Variances		t-test for Equality of Means		
		F	Sig.	t	df	Sig. (2-tailed)
The agility of the team was maintained in the agile process	Equal variances assumed	.024	.877	1.012	108	.314
	Equal variances not assumed			1.011	106.180	.314

		t-test for Equality of Means			
				95% Confidence Interval of the Difference	
		Mean Difference	Std. Error Difference	Lower	Upper
The agility of the team was maintained in the agile process	Equal variances assumed	.194	.192	-.186	.575
	Equal variances not assumed	.194	.192	-.187	.575

Table 6.33 Independent Samples Test - The agility of the team was maintained in the agile process

6.8 Chapter Summary

The aim of this chapter was to present the results that were derived from the post-experiment questionnaire that were designed to verify the hypotheses of the study. For each statement that was presented to the participants there was an explanation for its use, specifically, which hypothesis it was designed verify. The results for each statement was presented together with its support or other for the hypotheses and what it means to the study. Alternative explantions for results were also offered.

The chapter contributed to the completion of the aims and objectives of the study which were to improve Scrum as an agile process in relation to security requirements and to identify the best techniques and practices that improve the security considerations of an agile team. To this end, the results presented in this chapter have shown that the proposed method, presented as a framework for extending Scrum, has achieved the objectives of the study in that the results show that the inclusion of a Security Owner will improve security requirements consideration and has shown the suitability of UMLsec for improving the modelling of security requirements.

The results have clearly shown that the presence of a Security Owner role and the inclusion of UMLsec have had a significant effect on how the participant feel about their ability to manage security requirements and work as a team. In all areas of security requirements management which included the identification, prioritisation and modelling of security requirements, as well as the ability to work and communicate effectively as a team. Most of the participants agreed with the idea that these areas had improved with a Security Owner and UMLsec, contrast with a disagreement of these ideas from the group who did not have the Security Owner and UMLsec. The implication of these findings is that all of the hypotheses of the study have been verified to be confirmed. The following chapter will conclude the study and the findings that are presented here will contribute concluding this research and formulating ideas for future study.

CHAPTER SEVEN

CONCLUSION AND FUTURE WORK

Objectives

- Conclude the research work.
- Determining work limitations.
- Describing the future work.
- Specifying recommendations.

7. Conclusion and Future Work

7.1 Introduction

This research set out to see if it was possible to improve security requirements consideration and management in an agile development process through the introduction of a new role, the Security Owner, and UMLsec for modelling security requirements. This extension and improvement of agile development was achieve the a framework that detailed how these inclusions were to be achieved. The research sought to establish if the desired improvements in security requiremnts consideration and management were achieved through seeking the opinions of those who engage in Scrum development with the proposed inclusions.

The need to consider security requirements has stemmed from the fact an improvement in the software development process has not been met with an improvement in the consideration of security requirements. Moreover, this has been compounded by the fact that incremental and iterative modelling of security requirements is difficult. Although there have been attempts to consider security in incremental and iterative methods, there has been lack of clarity of how this is achieved. In response, this study proposed to extend and improve agile methods through the introduction of a Security Owner role and the use of the extended UML, UMLsec, towards improving the managing of security requirements.

7.2 Thesis Summary

A critical analysis of the literature was presented in Chapter Two and showed the efforts and different approaches to integrating security into agile development methods. This was conducted in order to reveal a gap in the research and the need to offer a better solution to managing security requirements. Much of the literature offered solutions for this purpose, however, there was a clear lack of how these ideas should be implemented and facilitated. This lead not only to the idea of an additional role but also the inclusion of a tool, UMLsec, that could be facilitated by this new role towards extending Scrum. This revealed a need to include a specific role for security requirements together with the use of UMLsec as an extension of UML. The study hypothesised that these two inclusions would improve the management of security requirements in the agile process without compromising its agility. The results showed that according the perceptions of the participants the agility of Scrum was maintained with these inclusions.

Chapter Three presented the methodology of the study beginning with an overview of possible methodological approaches that could be employed and a justification for the selected approach. Moreover, there was a presentation of the selected methods and a justification for their use in relation to the aims and objectives of the study. Chapter Three also included an overview of the overall research design for the study. Specifically, it was justified to conduct actual experimentation as opposed to researching the idea using online simulation which would not have included the interactions and relationships that take place in a Scrum setting, something that the proposed inclusions were hypothesised to improve. In reference to the methodological approach, the study is centered on the area of approaches to software development, while at the same time considering the social effects of an additional role on team members and their dynamics as well as improving security requirements consideration. Therefore, there was a need to methodologically address these two areas at the same. In response to this, the study adopted the ideas of Basili and Kitchenham in empirical research in software engineering. This was based on the idea that software engineering should follow the other physical sciences in reference to experimentation towards process improvement.

As the main contribution of the study was the proposed framework for the integration of the Security Owner role was presented in Chapter Four. Here the Security Owner was included to help and facilitate the agile team to improve security requirements consideration, both with the Product Owner negotiation and the Scrum team for modelling selected security features was demonstrated.

Chapter Five presented the experimentation of including a Security Owner role and UMLsec in an agile process. The experiment design was explained with a description of its structure as well as it aims to derive information about the impact of the Security Owner role and UMLsec. The extended Scrum development framework formed the basis of the survey that was undertaken as part of this research to identify any differences between the standard Scrum framework and the Scrum framework integrated with the new role. This information was derived from the use of questionnaires which enquired about the opinions of participants and were designed to verify the hypotheses of the study. The responses to the questionnaires were presented and analysed in Chapter Six.

7.3 Findings and Contributions

The hypotheses were designed to verify if the inclusion of UMLsec and the Security Owner role would have a positive effect on the Scrum as an agile process in terms of the management of security requirements. Through the use of experimentation and a post-experiment questionnaire these hypotheses were validated, and overall the results showed that the inclusion of a Security Owner role and UMLsec had a positive effect in this way. Specifically, the results showed that for all of the different aspects of managing security requirements, which included identifying, prioritising and modelling security requirements, there was a clear difference of opinion between the groups with and the groups without a Security Owner and UMLsec. This clearly indicated that such inclusions had a significant positive effect on the team members. For identifying security requirements a large majority of 79.3 percent of the participants agreed that the Security Owner help with this, this was against a strong level of disagreement that team members were helped where there was no Security Owner. The same was true for the prioritising of security requirements where 79.4 percent agreed the Security Owner helped in this area and only 19.2 percent of those that did not have a Security Owner agreed with this idea. These ideas were further supported by the fact that 75.9 percent of the participants felt that they could effectively manage security requirements. Similar levels of agreement were found for a number of different ideas related to the consideration and management of security requirements with presence of a Security Owner. Evidence that supported the use of UMLsec as a tool for modelling security requirements includeded that 77.5 percent of the respondents who used UMLsec agreed that they found it easy to model security requirements as opposed to only 36.5 percent of those who did not use UMLsec.

Therefore, the study as a proposal to improve the Scrum process as an agile development process in terms of security has been successful.

Importantly, Scrum as an agile development process includes different roles that come together periodically, during Scrum or sprint meetings, to discuss progress and the next tasks to be completed. Therefore, although each member has their own assigned role, the need for teamwork is very clear. In reference to security, this teamwork is especially a requirement because each team member is responsible for their own aspect of security which needs to be collectively considered. In relation to this idea, it was important that the inclusion of a Security Owner role would firstly, not interfere or be detrimental to teamwork, and secondly, improve teamwork in reference to security requirements. This was tested in the questionnaire using questions to determine the impact on teamwork of a Security Owner role. The results showed that such an inclusion significantly improved teamwork. Specifically, security requirements consideration in the Scrum meetings was found to be improved from the perspective of teamwork and the management of security as a team was also improved. Team communication and negotiation about security requirements was also found to be significantly improved.

This study integrated a new approach for the consideration and implementation of security requirements into an agile development method. The main contribution of the study was the introduction of a framework for extending and improving Scrum in terms of the consideration and management of security requirements through the introduction of a new security role and the use of UMLsec for modelling security requirements. The framework was based on a need that was identified in the literature that although there have been attempts to improve the integration of security requirements consideration, there was still a need for a coordinating function that would not only help to manage security requirements but also facilitate new tools for modelling security requirements.

The presented framework was verified as a valid method through the experimentation of the framework which showed that the presence of a Security Owner role improved the consideration and management of security requirements. Moreover, the results also showed that UMLsec improved modelling of security requirements.

The study showed that there was a need to reconsider the current approaches to security requirements in agile methods and thus the need to introduce a new method to address the current limitations. Specifically, the integration of a Security Owner role not only addressed the established limitations of existing agile frameworks which include a lack of awareness of security requirements, but also improved the agile process by facilitating Scrum team members when engaged in the security requirements of a product, this also allowed team members to be able to prioritise security requirements identified by the Security Owner. The study showed that the integration of UMLsec helped to emphasise or reveal security issues to team members that would normally have been neglected within an agile development framework. Importantly, the improved agile method will lead to a product with carefully considered and implemented security features making it more secure. The results of this research have demonstrated the importance of a Security

Owner role and integrating and modelling security features using UMLsec within an extended Scrum framework.

Scrum is an agile process and it was important that its agility was maintained when the Security Owner role and UMLsec were introduced. The results showed that agility was maintained, however, for the group without the Security Owner and UMLsec there was agreement with this idea and therefore, more investigation into the maintenance of agility is required. In reference to how this contributes to state of the art, of Baca and Carlson (2011) bring attention to the fact that it is the very agility of agile methods that means there is consideration of security requirements, and Baskerville (2004) says that a lack of use of documentation deters away from consideration of security requirements. However, the present study has shown that it is possible to improve the consideration and management of security requirements without affecting the agility of an agile process.

In this work we adopted the view of Baca and Carlson (2011) that there are numerous benefits to agile development which include that development takes place at a faster rate with more agility. However, there are concerns that this quicker development pace means that there is lack of documentation which can lead to less secure software (Baca and Carlson, 2011). Furthermore, research depicts that because agile methodologies primarily focus on less documentation than other processes they deter away from security considerations (Baskerville, 2004). Security specialists offer criticism of agile methods on the grounds of limited inherent security measures to establish secure software because they involve high-speed development that does not react to threats are continuously evolving and becoming increasingly sophisticated and dynamic (Baskerville, 2004). The current advocates of extending agile methodologies have, however, failed to establish why the current process cannot contain security practices as part of the recognised framework and the reasons behind necessity of an extension or amendment in order for this to occur; addressed in this study through the inclusion of a Security Owner role and UMLsec (see Chapter 4).

7.4 Limitations of the study

This section summarises the constraints and limitations in conducting research through conducting experiments in order to prove the validity of the proposed approach. These limitations are discussed here. The Scrum development framework is relatively a new area in the field of software development and therefore, there is a lack of experienced people who can use this method. This was particularly a problem because the researcher required that participants understand systems analysis, security requirements and agile methods as a requirement of using the Scrum framework in the experimentation of the study, without this a valid experiment could not take place because participants would not be able to participate and analyse the systems as required (see chapter five). Consequently, the researcher had to introduce and educate the participants to agile development methods, highlight the Scrum framework and conduct training for some agile activities before starting the evaluation experiments.

The experiments that were conducted were reflective of a real Scrum situation; however, where there may have been a limitation was their duration. A real Scrum development process can take a long time to complete, and therefore, to see the effect of a Security Owner role and UMLsec may produce more conclusive results over a longer period of time. Although this is a consideration of the present study, it was not practically possible to conduct experimentation using real Scrum development on this scale, this was due to resource limitations associated with time and cost as well as difficulties in gaining permission to conduct research within a real Scrum development process. Moreover, real Scrum development may take place across geographically distributed locations and there would be additional resources required for coordinating participants and the collection of data in different locations.

7.5 Future Work and Recommendations

When modelling critical and large systems the adoption of UMLsec is a major concern and in such projects there may be a need for the use of Scrum of Scrum teams where the Scrum Master of each team is involved in discussions in a cluster of teams. The recommendation here is to upscale the proposed approach of this study, i.e. the integration of UMLsec and the Security Owner role to the level of Scrum of Scrums. The aim of this study would be to see the effect of a Security Owner in this situation and what would be the best model of deployment of a Security Owner in a Scrum of Scrum structure. The methodology would have to include real Scrum development as experimentation on a geographically distributed scale would be difficult. The results of such a study would inform the future development of Scrum development processes in a globalised world.

Furthermore, software development using Scrum can be geographically distributed where there are a number of different teams working on the same product in different locations. A future study could examine the effect of a Security Owner role in this situation. Specifically, there would be a Security Owner in each Scrum team and the implications of the coordination that would be required between them could be investigated. Moreover, a central Security Owner role responsible for coordinating the team's Security Owners, or as a central role without the need for individual Security Owners, could be investigated. A research question in this study would address the best model of deployment, a central Security Owner or individual Security Owners and the associated aim would be to test these deployment approaches.

Moreover, there is a need for further research in developing a set of guidelines based on the main outcomes of this research. Specifically, these guidelines will show how the Security Owner role can be applied in agile methods in other areas such as government, industry and health care based on the foundation of the need for security modelling revealed in the present study. This extension to other contexts will require research into organisational policy and culture focusing on handling security requirements using an agile environment. The research should be concentrated on the Scrum teams' perception, attitudes and their daily practice with regard to agile security modelling in these different contexts.

Based on the results of this study, there is a need to train team members in considering and modelling security requirements within the new context of where a Security Owner role is present, this would include more training in the use of UMLsec. Lack of guidelines, agile principles and security knowledge can lead to projects failing. Thus, it is recommended that training employees in modelling agile security in the extended agile approach is considered. A future study could examine the benefits of this type of training, this is an extension of the present study whereby a Security Owner is introduced with little training of team members beforehand. An aim of this study would be to measure the level success of the inclusion of a Security Owner against training for the proposed extended framework of the present study.

In order to validate if the process remained agile during the implementation of the proposed framework, the responses of the participants was sought in this regard. A future study could examine further the agility of the Scrum process where it is extended using different measurement methods. Specifically, this measurement could examine agility from a quantitative scientific perspective by measuring delivery times and time taken to respond to changes in requirements or measured from a qualitiative perspective which could include adaptive planning, collaboration and continuous improvement.

BIBLIOGRAPHY

Alexander, I. 2003, "Misuse cases: use cases with hostile intent", IEEE Software, vol. 20, no. 1, pp. 58-66.

Alhir, S. (2002), Guide to applying the UML, page 350. Springer, 2002.

Almseidin, M. , Alrfou' , K. , Alnidami, N. and Tarawneh, A. (2015) 'A Comparative Study of Agile Methods: XP versus SCRUM', International Journal of Computer Science and Software Engineering (IJCSSE), 4(5), pp. 126 - 129.

Al Qurashi, S. and Qureshi, M.R.J. 2014, "Scrum of Scrums solution for large size teams using Scrum methodology", Life Science Journal, vol. 11, no. 8, pp. 443-449.

Al Saleem, S. Ullah, H. (2015). A Comparative Analysis and Evaluation of Different Agile Software Development Methodologies. International Journal of Computer Science and Network Security. 15 (7), 39 - 45.

Ambler, S. and Lines, M. (2009), Disciplined Agile Delivery (DAD): A Practitioner's Guide to Agile Software Delivery in the Enterprise, IBM Press, ISBN: 0132810131.

Ambler, S. and Lines, M. (2012), "Be Realistic About the UML: It's Simply Not Sufficient", web page "http://www.agilemodelling.com/essays/realisticUML.htm", accessed on May 23, 2012.

Ayalew, T. et al. (2011), Identification and Evaluation of Security Activities in Agile Projects, Springer-Verlag Berlin Heidelberg 2011.

Aydal, E. et al. (2006), Security Planning and Refactoring in Extreme Programming. Work, pp. 154 - 163.

Baca, D. Carlsson, B. (2011). Agile Development with Security Engineering Activities. ACM, 149 – 158.

Bannerman, P. Hossain, E. Jeffery, R. (2012). Scrum Practice Mitigation of Global Software Development Coordination Challenges: A Distinctive Advantage?. IEEE 45th Hawaii International Conference on System Sciences. 5309 - 5318.

Barbour, R. 2008, Doing Focus Groups, SAGE Publications Ltd.

Barn, B. (2007), Business Process Modelling, e-Framwork Workshop, JISC.

Basili, V. (1996). The Role of Experimentation in Software Engineering: Past, Current, and Future. IEEE, 442 – 449.

Basili, V. (2006a). The Role of Empirical Study in Software Engineering. Proceedings of the 30th Annual IEEE/NASA Software Engineering Workshop.

Basili, V. (2006b). Is There a Future for Empirical Software Engineering? (ISESE'06, September 21–22, 2006, Rio de Janeiro, Brazil). *ACM*.

Basili, V. (2008). Using Measures and Risk Indicators for Early Insight into Software Product Characteristics such as Software Safety (19th International Symposium on Software Reliability Engineering). *IEEE*. 4.

Basili, V. Elbaum, S. (2006). Empirically Driven SE Research: State of the Art and Required

Maturity (ICSE'06, May 20–28, 2006, Shanghai, China). *ACM*. 32.

Basili, V. Zelkowitz, M. (2007). Empirical Studies To Build A Science Of Computer Science. Communications Of The ACM. 50 (11), 33 – 37.

Baskerville, R. (2004), Agile security for information warfare: a call for research. ECIS.

Beck, K. (2004), Extreme Programming explained: Embrace change. *Reading, Mass., Addison-Wesley, Nov16, 2004.*

Bell, D. (2003) UML basics: An introduction to the Unified Modelling Language http://www.ibm.com/developerworks/rational/library/769.html, accessed on May 23, 2012.

Bennet, S. et al. (2010), Object-Oriented System Analysis and Design Using UML, McGraw Hill, 2nd Edition, ISBN: 9780077125363.

Beznosov, K. (2003), Extreme Security Engineering : On Employing XP Practices to Achieve " Good Enough Security " without Defining It. *Computer Engineering*, (2003).

Beznosov, K. and Kruchten, P. (2005), 'Towards agile security assurance'. *Proceedings of the 2004 workshop on New security paradigms - NSPW '04* pp.47.

Boehm, B. and Turner, R. (2004, May). Balancing agility and discipline: Evaluating and integrating agile and plan-driven methods. In *Proceedings of 26th International Conference on Software Engineering, 2004. ICSE 2004*, pp. 718-719, IEEE.

Bryman, A. (2001) Social Research Methods. Oxford, Oxford University Press.

Bryman, A. and Bell, A. (2007) 'Part Two' in Anonymous (eds.) Business Research Methods, Second Edition edn, New York United States: Oxford University Press. pp. 154-155.

Calder, A. and Watkins, S. (2012), IT Governance: An International Guide to Data Security and ISO27001/ISO27002, Kogan Page Limited, Great Britain and United States, 5th ed. (2012), ISBN: 9780749464851.

Clagett II, L. (2009), Security Requirements for the Prevention of Modern Software Vulnerabilities and a Process for Incorporation into Classic Software Development Life Cycles, Master thesis, 2009.

Cohen, L. et al. (2000), Research Methods in Education. London. Routledge Falmer.

Cohen, D. et al. (2004), "An Introduction to Agile Methods", Fraunhofer Center for Experimental Software Engineering, USA, Advances In computers, Vol. 62, 2004 Elsevier Inc. ISSN: 0065-2458/DOI 10.1016/S0065-2458 (03) 62001-2.

Cohn, M. (2010), Succeeding with agile: software development using Scrum, Upper Saddle River, NJ: Addison-Wesley.

Creswell, J. (2003), Research design, qualitative, quantitative, and mixed methods approaches. California, Sage. Thousand Oaks.

Crisp, Planning poker, http://www.crisp.se/bocker-och-produkter/planning-poker [Accessed on 10/06/2014].

Dandurand, F. Shultz, T. Onishi, K. (2008). Comparing online and lab methods in a problem solving experiment. *Behaviour Research Methods*. 40 (2), 428 - 434.

Daniel, P. (1995), Object Oriented Analysis and Design, Information Systems Management, Winter 95, Vol 12 Issue 1, p54.

Deemer, P. et al (2012), A Lightweight Guide to the Theory and Practice of Scrum, Ver. 2.0, 2012.

Denscombe, M. (2003) The Good Research Guide - For Small-Scale Social research Projects. Maidenhead. Philadelphia. Open University Press.

Derby, E. And Larsen, D. (2006), Agile Retrospectives (Derby, Larsen 2006). Washington, D.C.

Dijkman R., and Joosten S., (2002), Technical Report, Deriving Use Case Diagrams from Business Process Models.

Dyba, T. and Dingsoyr, T. (2009). What do we know about agile software development?. Software, IEEE, 26(5), 6-9.

Easterbrook, S. Singer, J. Storey, M-A. Damian, D. (2007). Selecting Empirical Methods for Software Engineering Research.

Eliza S. F. Cardozo, et al. (2010), SCRUM and Productivity in Software Projects: A Systematic Literature Review, ACM (2010).

El-Attar, M. (2012 A),"A framework for improving quality in misuse case models", Business Process Management Journal, Vol. 18 Iss 2 pp. 168 – 196.

El Attar, M. (2012 B). Towards developing consistent misuse case models.The Journal of Systems and Software. 85 (1), 323 - 339.

Firesmith, D.G. (2003). Security Use Cases. Journal of Object Technology. 2(3): 53-64.

Franqueira, V. Bakalova, Z. Tun, T. Daneva, M. (2011). Towards Agile Security Risk Management in RE and Beyond. IEEE, 33 – 36.

Galliers, R. (1992) Information Systems Research, Issues, Methods and Practical Guidelines. Oxford, Blackwell Scientific Publications.

Gilbert, N. (2004) Researching Social Life. London, SAGE Publications Ltd.

Gillham, B. 2005, Research interviewing: the range of techniques, Open University Press, GB.

Goertzel, K. et al. (2007), Software Security Assurance, State-of-the-Art Report (SOAR), Information Assurance Technology Analysis Centre (IATAC) and Data and Analysis Center for Software.

Gomaa, H. and E. Olimpiew (2005). The Role of Use Cases in Requirements and Analysis Modeling. Workshop on Use Cases in Model-Driven Software Engineering. Montego Bay, Jamaica.

Harleen, K. et al. (2014) ‘An Investigation into Mobile Application Development Process, Challenges and Best Practices’. International Journal of Modern Education and Computer Science (IJMECS).

Hartson, H. Rex (2012) The UX book: process and guidelines for ensuring a quality user experience, Published: Waltham, M.A.: Morgan aufmann2012.

Hazzan, O. et al. (2006), Qualitative Research in Computer Science Education, ACM SIG SE Bulletin, Vol. 38, issue 1, pp 408-412.

Hope, P. McGraw, G. Anton, A. (2004). Misuse and Abuse Cases: Getting Past the Positive. IEEE Security and Privacy, 90 - 92.

Houmb, S. H., Islam, S., Knauss, E., Jürjens, J., and Schneider, K. (2010). Eliciting security requirements and tracing them to design: an integration of Common Criteria, heuristics, and UMLsec. Requirements Engineering, 15(1):63– 93.

Hummel, M. Epp, A. (2015). Success Factors of Agile Information Systems Development: A Qualitative Study. 48th Hawaii International Conference on System Sciences. , 5045 - 5054.

Implementing Scrumban A short guide to implementing Scrumban at your organisation. October 2013, www.SwitchingToScrum.com.

Introduction to Data Analysis Handbook, Migrant and Seasonal Head Start, Technical Assistance Center.

Jain, S. and Ingle, M. 2011, "Software Security Requirements Gathering Instrument",International Journal of Advanced Computer Science and Applications, vol. 2, no. 7, pp. 116-121.

Jakobsen, C.R. and Sutherland, J. 2009, "Scrum and CMMI - Going from good to great: Are you ready-ready to be done-done?" in Agile conference, pp. 333-337.

John V. Seidel (1998), Qualitative Data Analysis, ftp://ftp.qualisresearch.com/pub/qda.pdf

Johnson, R. (1999), An Industry Analysis of Developer beliefs about Object-Oriented Systems Development, Database for Advances in Information Systems, Winter, 1999.

Johnstone, M.N. (2011), “Modelling misuse cases as a means of capturing, security requirements”, in Proceedings of the 9th Australian Information Security Management Conference, Perth, Western Australia, (2011) December 5-7.

Jurian van de Laar (2012), Risk Poker: Risk based testing in agile projects, Management and Leadership in Software Organisations, May 2012, www.agilerecord.com, free digital version made in Germany ISSN 2191-1320 issue 10.

Jurjens, J. (2002), UMLsec: Extending UML for Secure Systems Development, *In Proceedings of the 5th International Conference on The Unified Modelling Language* (2002), pp. 412-425 Key: citeulike: 9861823

Jurjens, J. (2005 A), Secure Systems Development with UML, Springer-Verlag Berlin Heidelberg, 1st edition-2005, Germany, ebook ISBN: 978-3-540-26494-1.

Jürjens, J (2005 B) Sound methods and effective tools for model-based security engineering with UML, in: Proceedings of the 27th International Conference on Software Engineering (ICSE), May 2005.

Kanyaru, J.M. and Phalp, K. (2009), "Validating software requirements with enactable use case descriptions", Requirements Engineering, vol. 14, no. 1, pp. 1-14.

Kampenes, V.B., Dybå, T., Hannay, J.E. and K. Sjøberg, D.I. (2009), "A systematic review of quasi-experiments in software engineering", Information and Software Technology, vol. 51, no. 1, pp. 71-82.

Keramati, H.; Mirian-Hosseinabadi, S.-H. , (2008) "Integrating software development, security activities with agile methodologies," Computer Systems and Applications, 2008. AICCSA 2008. IEEE/ACS International Conference on, vol., no., pp. 749-754, March 31 2008-April 4 2008.

Keyson, D. and Alonso, M. (2009), 'Empirical Research Through Design' http://www.iasdr2009.org/ap/Papers/Special%20Session/Assessing%20knowledge%20generated%20by%20research%20through%20design/Empirical%20Research%20Through%20Design.pdf

Kitchenham, B. Budgen, D. (2002). Editorial Special Issue: Empirical Software Engineering. *IEE Proceedings Software*. 149 (5), pp. 113 - 114.

Kitchenham, B. Pfleeger, S. Pickard, L, Jones, P. Hoaglin, D. El emam, K. Rosenberg, J. (2002). Preliminary Guidelines for Empirical Research in Software Engineering. *IEEE Transactions on Software Engineering*. 28 (8), pp. 721 - 734.

Kongsli, V. (2006), Towards agile security in web applications. *Companion to the 21st ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications* - OOPSLA '06, (2006), 805.

Kotenko, I. & Chechulin, A. 2013, "A Cyber Attack Modeling and Impact Assessment framework", IEEE, , pp. 1.

Kumar, M. Kumar, D. Rani, B. Rao, K. (2011). Research Methodology on Agile Modeled Layered Security Architectures. International Journal of Software Engineering Research and Practices. 1 (2), 21 - 26.

Larman, C. (2005),"Applying UML and patterns: an introduction to object-oriented analysis and design and iterative development", 3rd edition, Pearson Education, Inc. 2005, USA.

Leiwo, J. and Virtanen, V. (2005) "Understanding and Communicating IT Security Specifications with UML", presented at International Journal of Software Engineering and Knowledge Engineering, 2005, pp. 923-940.

Lilly S (1999) Use case pitfalls: top 10 problems from real projects using use cases. In: *Proceedings of TOOLS* USA 1999, IEEE Computer Society, Santa Barbara, California

Liza, A. et al. (2014), ‘Emergence of Agile Methods’: Perceptions from Software Practitioners in Malaysia, IEEE pp. 30-39.

Luj´an-Mora, S. et al. (UML 2002).: Extending UML for Multidimensional Modelling. In: Proc. Of the 5th International Conference on the Unified Modelling Language. Volume 2460 of LNCS., Dresden, Germany, SpringerVerlag.

Mannaro, K. Melis, M. Marchesi, M. (2004). Empirical Analysis on the Satisfaction of IT Employees Comparing XP Practices with Other Software Development Methodologies. *LNCS*. 3092, 166 - 174.

Martin, R. (2003),"Agile Software Development: principles, patterns, and practices", Pearson Education, Inc., Prentice Hall, 2003, USA.

Matharu, G. Mishra, A. (2015). Empirical Study of Agile Software Development Methodologies: A Comparative Analysis. ACM SIGSOFT Software Engineering Notes. 40, 1 – 6.

Matulevičius, R and Dumas, M. (2011), Towards Model Transformation between SecureUML and UMLsec for Role-based Access Control. *In Proceedings of the 2011 conference on Databases and Information Systems, Computer Science Information Systems*, 339-352.

McConnell, S. (1996), "Rapid Development". Redmond, WA.: Microsoft Press.

McGraw, G. (2006). Software security : building security in. Upper Saddle River, NJ: Addison-Wesley.

Mead, N. Viswanathan, V. Padmanabhan, D . (2008). Incorporating Security Requirements Engineering into the Dynamic Systems Development Method. Annual IEEE International

Computer Software and Applications Conference, 949 – 954.

Mellado, D., Blanco, C., Sánchez, L.E. and Fernández-Medina, E. (2010), "A systematic review of security requirements engineering", Computer Standards and Interfaces, vol. 32, no. 4, pp. 153-165.

Melnik. G. Maurer, F. "Perceptions of Agile Practices: A Student Survey," Extreme Programming/Agile Universe, Chicago, IL, 2002.

Misra, S. Kumar, V. Kumar, U. Fantazy, K. Akhter, M (2012),"Agile software development practices: evolution, principles, and criticisms", International Journal of Quality and Reliability Management, Vol. 29 Iss 9 pp. 972 – 980 Permanent link to this document: http://dx.doi.org/ 10.1108/02656711211272863

Moe, N. Dingsøyr, T. Dyba, T. (2010). A teamwork model for understanding an agile team: A case study of a Scrum project. Information and Software Technology. 52, 480 – 491.

Moniruzzaman ABM, Hossain DSA. (2013), Comparative study on agile software development methodologies. GJCST. 2013; 13(7):33–56.

Mougouei, D. et al (2013), "S-Scrum: a Secure Methodology for Agile Development of Web Services", World of Computer Science and Information Technology Journal (WCSIT).

Myers, M. (1997) Qualitative Research in Information Systems. MIS Quarterly, 21 (2), pp. 241-242.

Nachmias, C. and D. Nachmias (1981) Research Methods in the Social Sciences. London, St. Martin's Press.

Nagappan, N. Murphy, B. Basili, V . (2008). The Influence of Organizational Structure on Software Quality: An Empirical Case Study (ICSE'08, May 10–18, 2008, Leipzig, Germany). *ACM*. 521 - 530.

Nedic, Z. Machotka, J. Nafalski, A. (2003). Remote Laporatories Versus Virtual and Real Laporatories (33rd ASEE/IEEE Frontiers in Education Conference). *IEEE*. 1 - 6.

Neher, J. (2012), "Abuser Stories: Thinking Like the Bad Guy to Reduce Security Vulnerabilities", Presented, Emerging Applications of Agile, Workshop Conference**:** Agile http://conferences.agilealliance.org/sessions/13318#sthash.EWKMZECN.dpuf. [Accessed on 12/06/2014].

Opdahl, A.L. and Sindre, G. (2009), "Experimental comparison of attack trees and misuse cases for security threat identification",Information and Software Technology, vol. 51, no. 5, pp. 916-932.

Ouchani, S. and Debbabi, M. (2015),"Specification, verification, and quantifi- cation of security in model-based systems," Computing, vol. 97, no. 7, 2015.

Palvia, P. Mao, E. (2003) Management Information Systems Research: What's There in a Methodology? Communication of the Association for Information Systems, 11, 16.

Palys, T. (2008). Purposive sampling. In L. M. Given (Ed.) The Sage Encyclopedia of Qualitative Research Methods. (Vol.2). Sage: Los Angeles, pp. 697-8.

Pandey, S. and Mustafa, K. (2010), Recent advances in SRE research, (IJCSE) International Journal on Computer Science and Engineering, Vol. 02, No. 04, 2010, 1079-1085

Peeters, J. (2005), “Agile Security Requirements Engineering”, 2005.

Petersen, K. and Wohlin, C. (2009). ‘A comparison of issues and advantages in agile and incremental development between state of the art and an industrial case’: *Journal of Systems and Software*, 82(9), 1479-1490.

Peterson, G. Steven, J. (2006). Defining Misuse within the Development Process. IEEE Security and Privacy. 81 - 84.

Pikkarainen, M., Haikara, J., Salo, O., Abrahamsson, P. and Still, J. (2008), "The impact of agile practices on communication in software development", Empirical Software Engineering, vol. 13, no. 3, pp. 303-337.

Pinsonneault, A. and Kraemer, K. (1993), Survey Research Methodology in Management Information Systems: An Assessment. Journal of Management Information Systems, 10 (2), pp. 75-105.

Pollanen, T. (2014). The advantages and challenges of web-based experiments in behavioral sciences. Available: http://www.web-psychometrics.com/advantages_challenges.html. Last accessed 15th September, 2015.

Pooley, R. and Wilcox P. (2004), Applying UML: Advanced Applications, Elsvier Ltd, Great Britain (2004), ISBN 0750656832.

Punch, K. (2005), Introduction to Social Research: Quantitative and Qualitative Approaches. London, SAGE Publications Ltd.

Qualitative Data Analysis, investigating the social world, http://www.sagepub.com/upm-data/43454_10.pdf , chapter 10, pp. 320-357.

Ramesh, B., Cao, L. and Baskerville, R. (2010), "Agile requirements engineering practices and challenges: an empirical study", Information Systems Journal, vol. 20, no. 5, pp. 449-480.

Rao, K. Naidu, K. Chakka, P. (2011). A Study of the Agile Software Development Methods, Applicability and Implications in Industry.International Journal of Software Engineering and Its Applications. 5 (2), 35 - 46.

Raschke, W. Zilli, M. Baumgartner, P. Loinig, J. Steger, C. Kreiner, C. (2014). Supporting Evolving Security Models for an Agile Security Evaluation. IEEE. 31 - 36.

Rehman, T. et al. (2013), Analysis of Requirements Engineering Processes, Tools/Techniques and Methodologies, I.J. Information Technology and Computer Science, 2013, 03, 40-48.

Remenyi, D. et al. (1998), Doing research in business and management: an introduction to process and method. London, SAGE.

Robson, C. (2002) Real world research: a resource for social scientists and practitioner-researchers. Oxford, Blackwell.

Romano, B. da Silva, A. (2015). Project management using the Scrum agile method: A case study within a small enterprise. IEEE. 774 - 776.

Salo, O. and Abrahamsson, P. (2005) “Integrating Agile Software Development and Software Process Improvement: a Longitudinal Case Study”, pp. 193-202.

Sani, A. Ghani, I. Jeong, S (2013). Secure Dynamic System Development Method (SDSDM) Model for Secure Software Development. Science International, 1059 – 1064.

Santos, M. Bermejo, P. de Oliveira, M. Tonelli, A.O. (2011), "Agile Practices: An Assessment of Perception of Value of Professionals on the Quality Criteria in Performance of Projects", Journal of Software Engineering and Applications, vol. 4, no. 12, pp. 700-709.

Schieferdecker I, Grossmann J, Schneider M. Model-based security testing. Proceedings 7th Workshop on Model-Based Testing 2012.

Schwaber, K. (2004),"Agile Project Management with Scrum", Published by Microsoft Press 2004, USA.

Schwaber, K. and Beedle, M. (2002), "Agile Software Development with Scrum", Pearson International Edition, Prentice Hall, Inc. 2002, USA.

Sheetal, S. (2012), ”Agile Processes and Methodologies: A Conceptual Study”, International Journal on Computer Science and Engineering (IJCSE). Vol. 4 pp. 5

Shore, J. and Warden, S. (2008),"The Art of Agile Development", Published by O'Reilly Media, Inc. 2008, USA.

Shull, F. Seaman, C. Zelkowitz, M. (2006). Victor R. Basili's Contributions to Software Quality. *IEEE Computer Society*, 16 - 18.

Silverman, D. (2005), Doing Practical Research: A Practical Handbook, First edn, London UK: Sage.

Sindre, G. and Opdahl, A. (2000), Eliciting security requirements by misuse cases, In Proc. TOOLS Pacific 2000, pages 174-183, November 2000.

Silva da Silva, T. et al (2011)'User-Centered Design and Agile Methods': A Systematic Review, Agile Conference (AGILE), , pp.77-86.

Siponen, M. et al. (2005), Integrating Security into agile Development Methods. *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, pp.185.

Sobh, R. and Perry, C. (2006), "Research design and data analysis in realism research",European Journal of Marketing,vol. 40, no. 11/12, pp. 1194-1209.

Sohaib, O. and Khan, K. (2010), 'Integrating Usability Engineering and Agile Software Development': A Literature Review, Proceedings of the ICCDA 2010 conference, pp.32-38.

Spears, J.L. & Parrish, J.L., Jr 2013, "Teaching Case: IS Security Requirements Identification from Conceptual Models in Systems Analysis and Design: The Fun & Fitness, Inc. Case", Journal of Information Systems Education, vol. 24, no. 1, pp. 17.

Stephens, M. Rosenburg, D (2003). Extreme Programming Refactored: The Case Against XP. New York: Apress.

Tongco, D . (2007). Purposive Sampling as a Tool for Informant Selection. Ethnobotany Research and Applications. 5, 147 - 158.

Trujillo, J. et al. (2009). A UML 2.0 profile to define security requirements for Data Warehouses. Comput. Stand. Interfaces 31, 5 (September 2009), 969-983.

Turk, D. France, R. Rumpe, B. (2002). Limitations of Agile Software Processes. Processes in Software Engineering, 43 – 46.

Vaus, D. (2004) Surveys in Social Research. London, Routledge.

Wagener, J. et al (2011-2012), Project Management Using Kanban, University of Luxembourg, Master in Computer Science, Advanced Project Management, 2011 – 2012.

Wendy, O. (2004), 'Triangulation in Social Research: Qualitative and Quantitative Methods Can Really Be Mixed', Developments in Sociology.

Williams, L. and Cockburn, A. (2003), "Agile Software Development: It's about Feedback and Change," IEEE Computer, June 2003, pp. 39-43.

Williams, L. et al. (2008), *Protection Poker: Structuring Software Security Risk Assessment and Knowledge Transfer*, National Science Foundation.

Williams, L. et al. (2009), Engineering Secure Software and Systems-Lecture Notes in Computer Science (2009), Protection Poker: Structuring Software Security Risk Assessment and Knowledge Transfer, Volume 5429, pp 122-134.

Williams, L. and Meneely, A. (2010), Assessing Risk Protection Poker: The New Software Security "Game", North Carolina State University-Grant Shipley Red Hat, 2010 IEEE.

Wilson, T. (2001) Structure in Research Methods. Available from: [http://informationr.net/ tdw/publ/ppt/ResMethods/] [Accessed 22/01/2014].

Wingo, R. Tanik, M. (2015). Using an Agile Software Development Methodology for a Complex Problem Domain. Proceedings of the IEEE SoutheastCon 2015.

Woods, P. (2006), 'Qualitative Research', http://www.edu.plymouth.ac.uk/resined/ qualitative %20methods%202/qualrshm.htm

Zaigham, M. (2012), "Novel Hybrid Model: Integrating Scrum and XP", International Journal of Information Technology and Computer Science (IJITCS). MECS. vol. 4, pp.6.

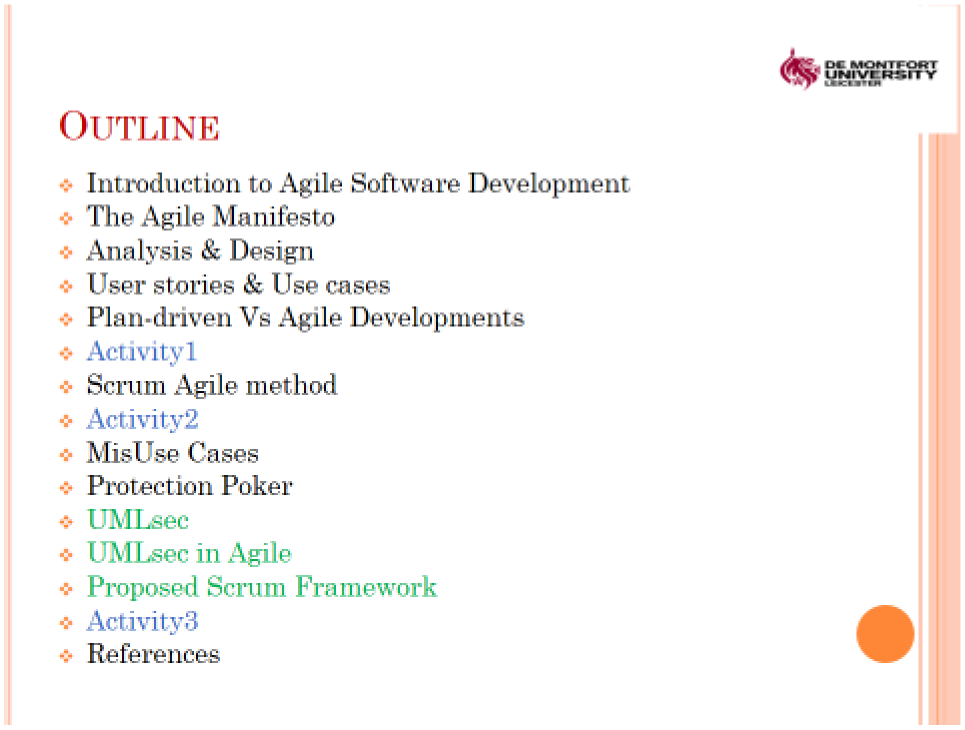
APPENDIXES

A. Training course

DE MONTFORT UNIVERSITY LEICESTER

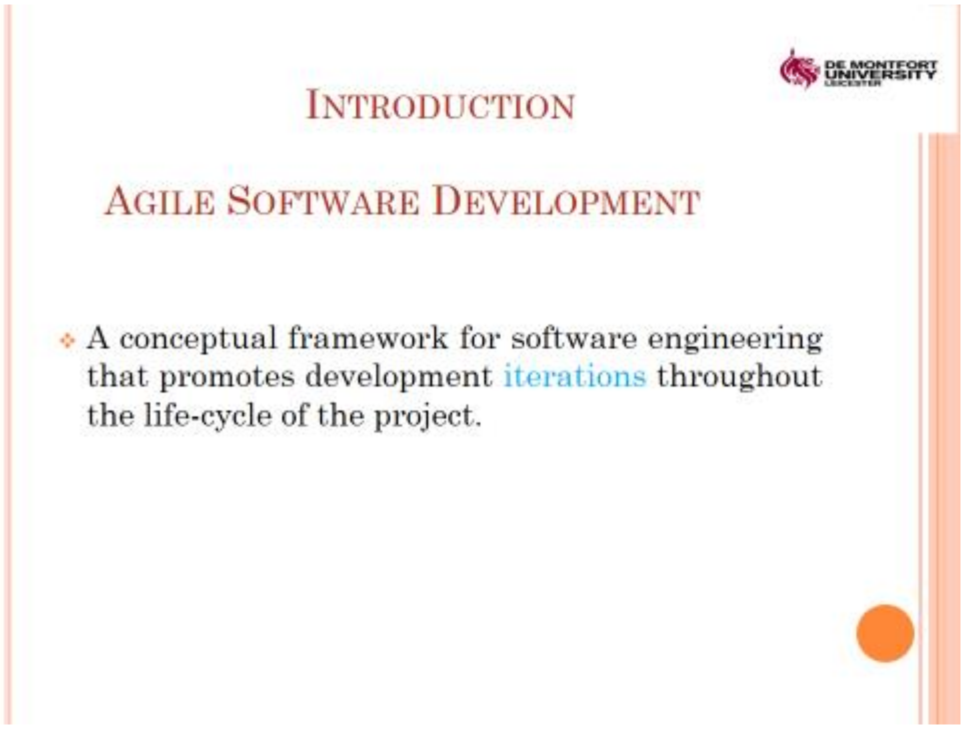
TRAINING COURSE

©Minahi 2014


DE MONTFORT UNIVERSITY LEICESTER

OUTLINE

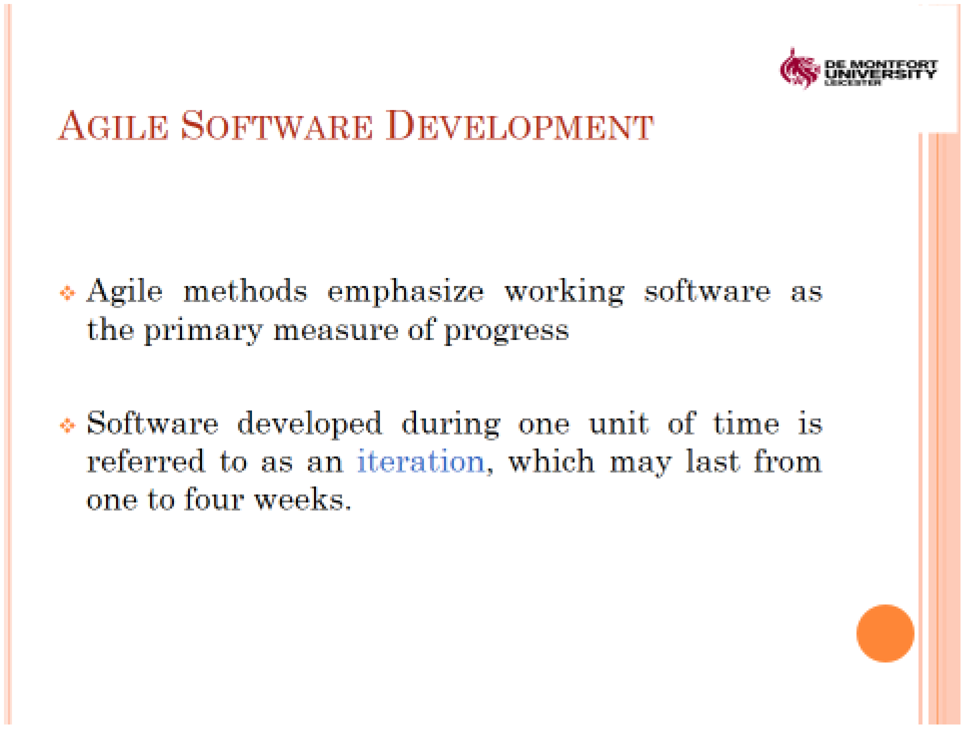
- Introduction to Agile Software Development
- The Agile Manifesto
- Analysis & Design
- User stories & Use cases
- Plan-driven Vs Agile Developments
- Activity1
- Scrum Agile method
- Activity2
- MisUse Cases
- Protection Poker
- UMLsec
- UMLsec in Agile
- Proposed Scrum Framework
- Activity3
- References



INTRODUCTION

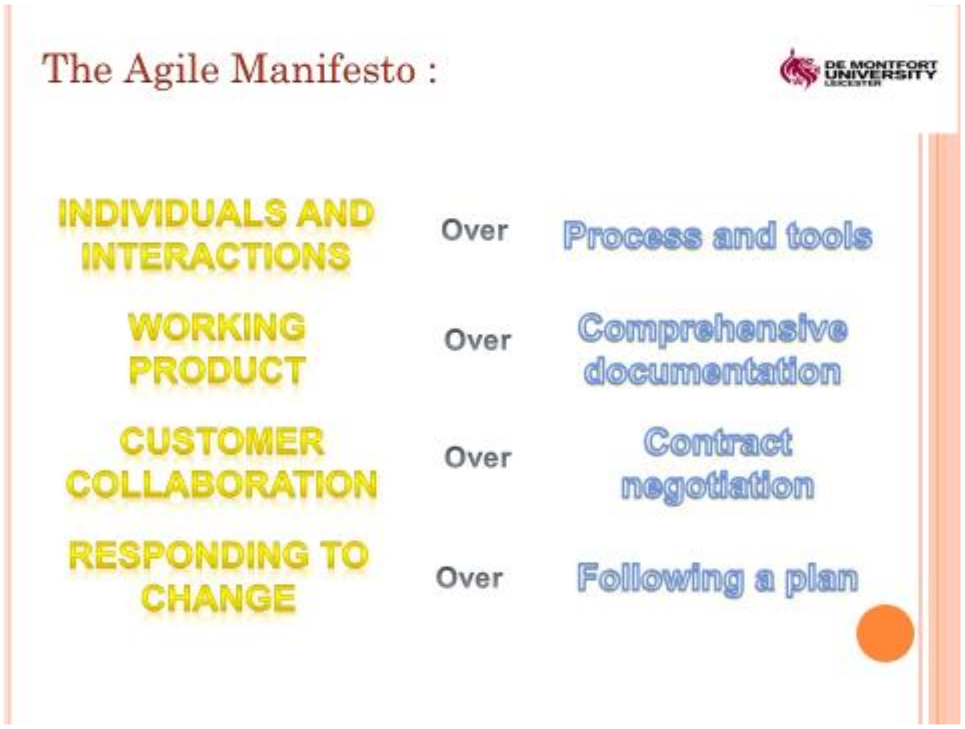
AGILE SOFTWARE DEVELOPMENT

- A conceptual framework for software engineering that promotes development iterations throughout the life-cycle of the project.



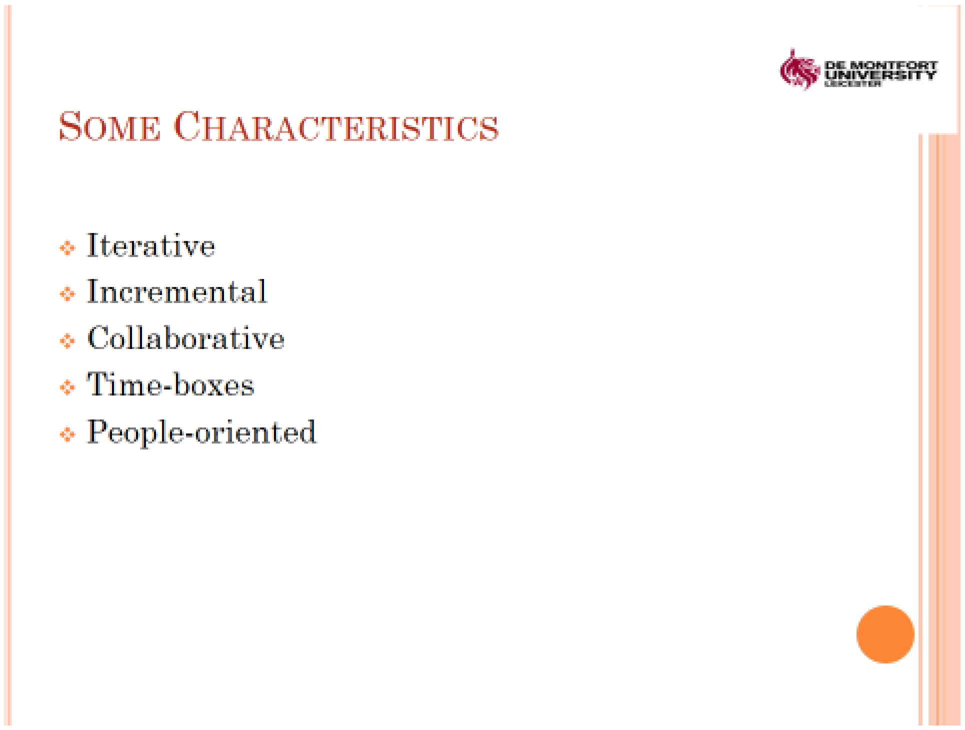
AGILE SOFTWARE DEVELOPMENT

- Agile methods emphasize working software as the primary measure of progress
- Software developed during one unit of time is referred to as an iteration, which may last from one to four weeks.



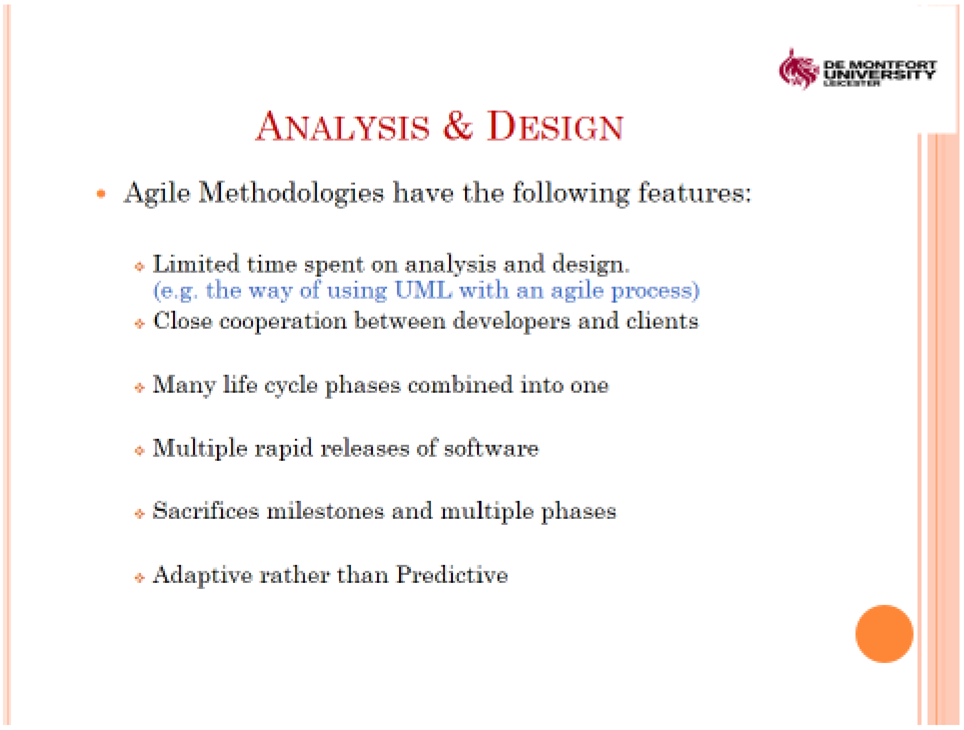
The Agile Manifesto :

INDIVIDUALS AND INTERACTIONS	Over	Process and tools
WORKING PRODUCT	Over	Comprehensive documentation
CUSTOMER COLLABORATION	Over	Contract negotiation
RESPONDING TO CHANGE	Over	Following a plan



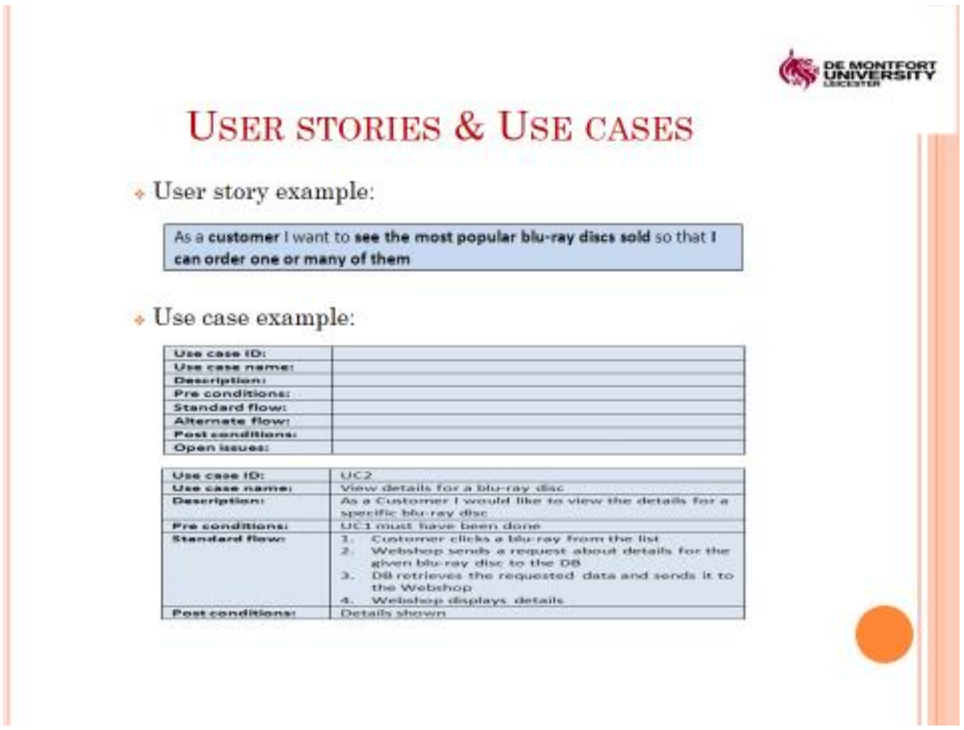
Some Characteristics

- Iterative
- Incremental
- Collaborative
- Time-boxes
- People-oriented



ANALYSIS & DESIGN

- Agile Methodologies have the following features:
  - Limited time spent on analysis and design.
    (e.g. the way of using UML with an agile process)
  - Close cooperation between developers and clients
  - Many life cycle phases combined into one
  - Multiple rapid releases of software
  - Sacrifices milestones and multiple phases
  - Adaptive rather than Predictive



USER STORIES & USE CASES

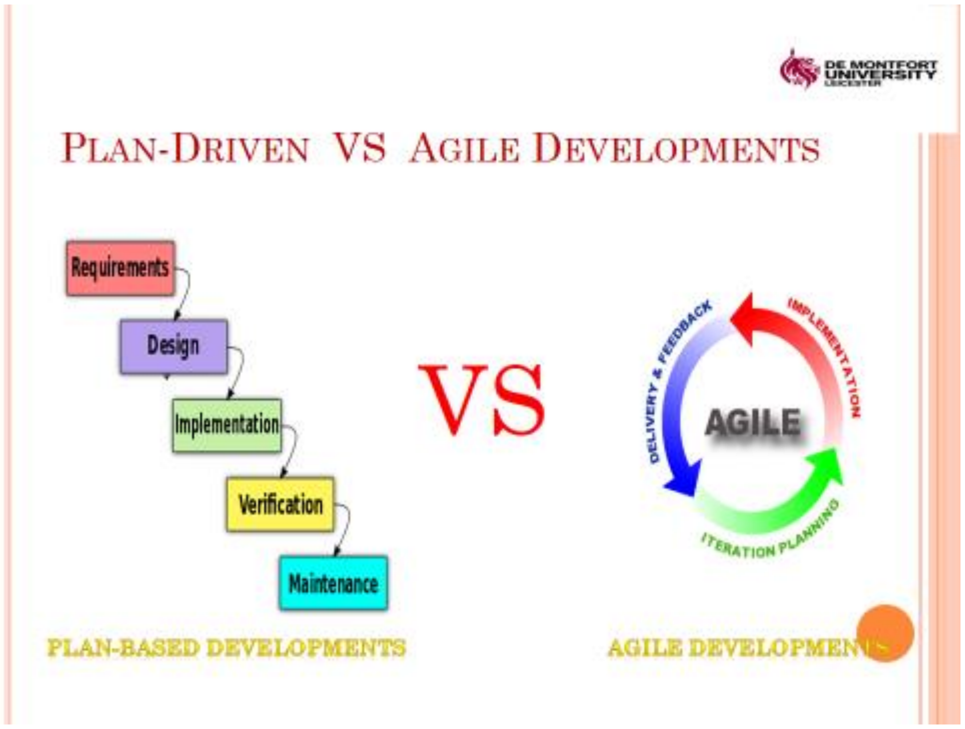
- User story example:

As a **customer** I want to **see the most popular blu-ray discs sold** so that **I can order one or many of them**

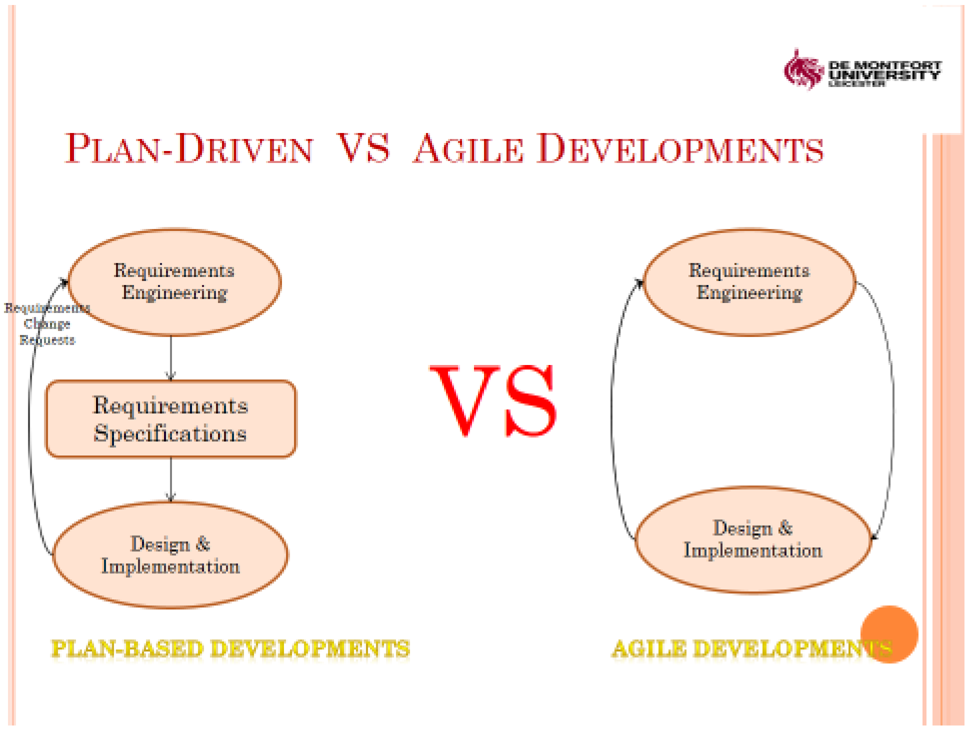
- Use case example:

Use case ID:	
Use case name:	
Description:	
Pre conditions:	
Standard flow:	
Alternate flow:	
Post conditions:	
Open issues:	

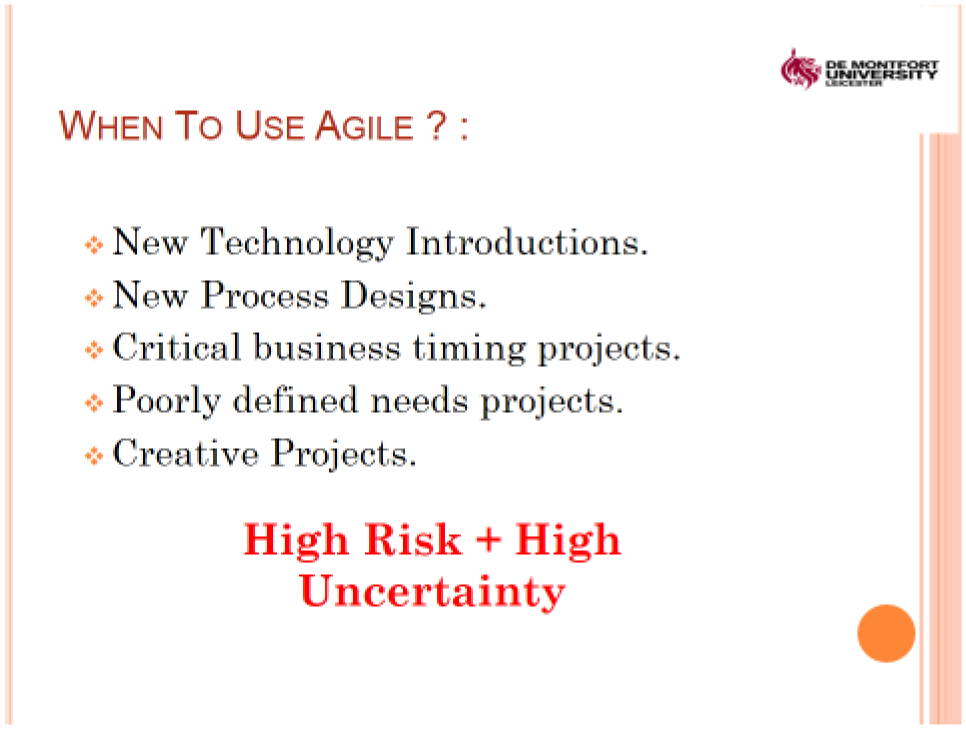
Use case ID:	UC2
Use case name:	View details for a blu-ray disc
Description:	As a Customer I would like to view the details for a specific blu-ray disc
Pre conditions:	UC1 must have been done
Standard flow:	1. Customer clicks a blu-ray from the list 2. Webshop sends a request about details for the given blu-ray disc to the DB 3. DB retrieves the requested data and sends it to the Webshop 4. Webshop displays details
Post conditions:	Details shown



PLAN-DRIVEN VS AGILE DEVELOPMENTS



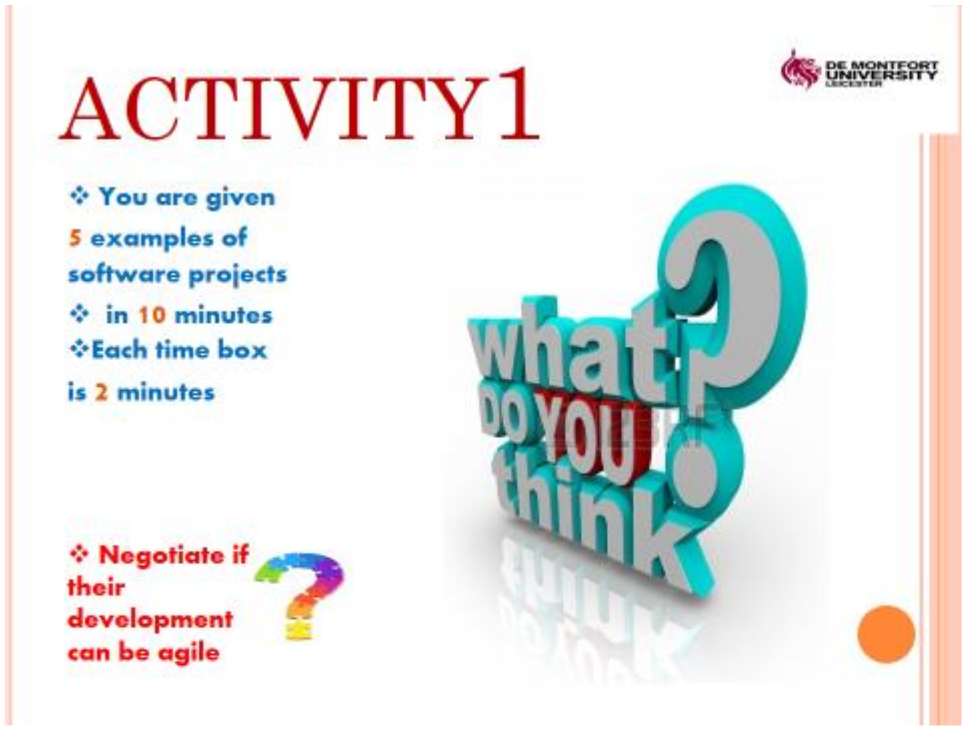
PLAN-DRIVEN VS AGILE DEVELOPMENTS



WHEN TO USE AGILE ? :

- New Technology Introductions.
- New Process Designs.
- Critical business timing projects.
- Poorly defined needs projects.
- Creative Projects.

High Risk + High Uncertainty

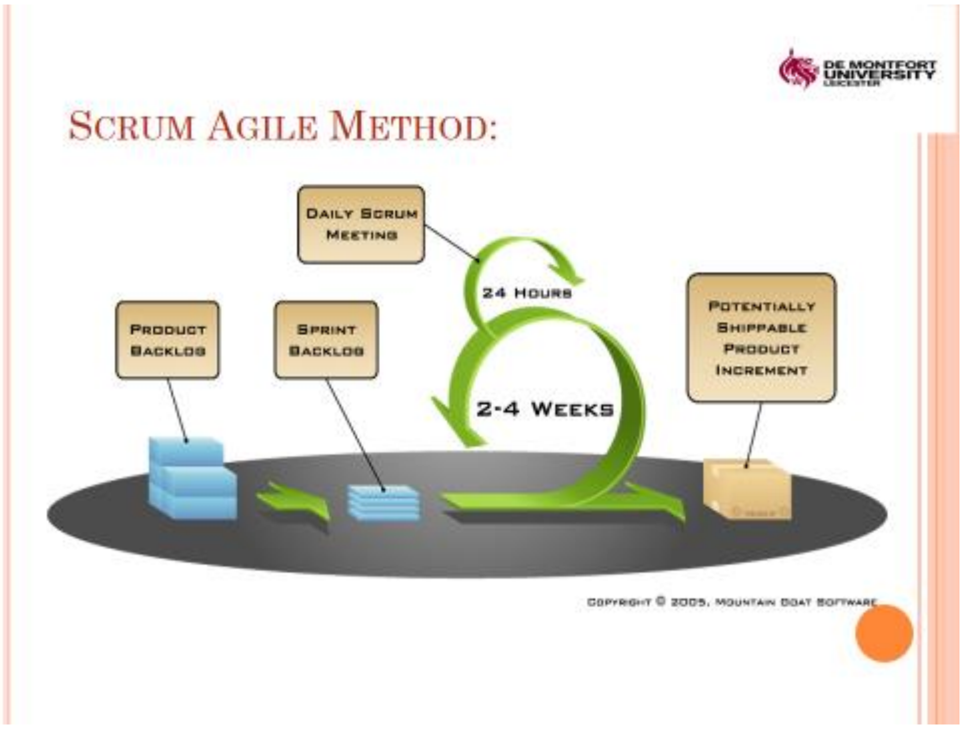


ACTIVITY1

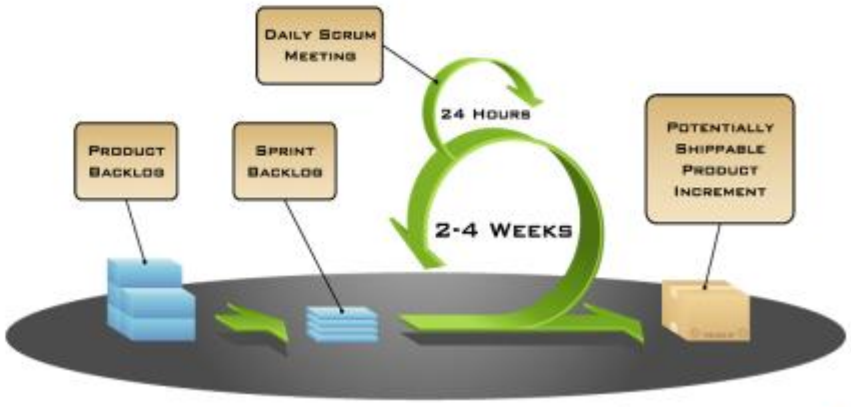
- You are given 5 examples of software projects
- in 10 minutes
- Each time box is 2 minutes

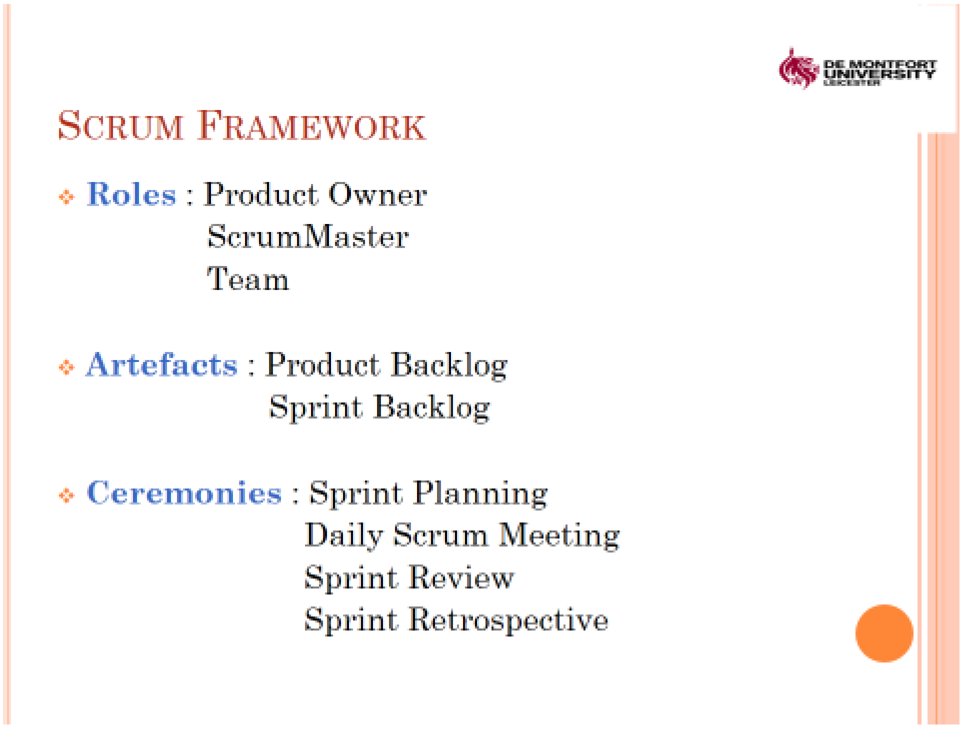
- Negotiate if their development can be agile

what DO YOU think?



SCRUM AGILE METHOD:


DAILY SCRUM MEETING
24 HOURS
PRODUCT BACKLOG
SPRINT BACKLOG
2-4 WEEKS
POTENTIALLY SHIPPABLE PRODUCT INCREMENT
COPYRIGHT © 2005, MOUNTAIN GOAT SOFTWARE



SCRUM FRAMEWORK

- **Roles** : Product Owner
  ScrumMaster
  Team
- **Artefacts** : Product Backlog
  Sprint Backlog
- **Ceremonies** : Sprint Planning
  Daily Scrum Meeting
  Sprint Review
  Sprint Retrospective

ACTIVITY2

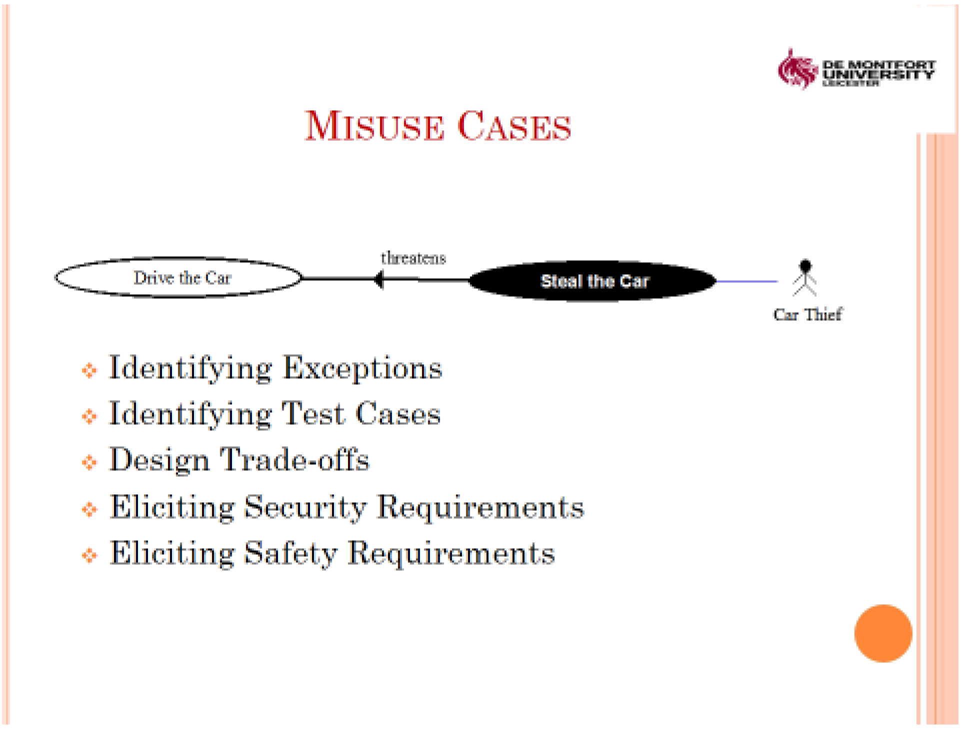
5 minutes.

Try to Organize the Group as a SCRUM team!

WHAT YOU HAVE TO DO THEN ?

HERE WE ARE

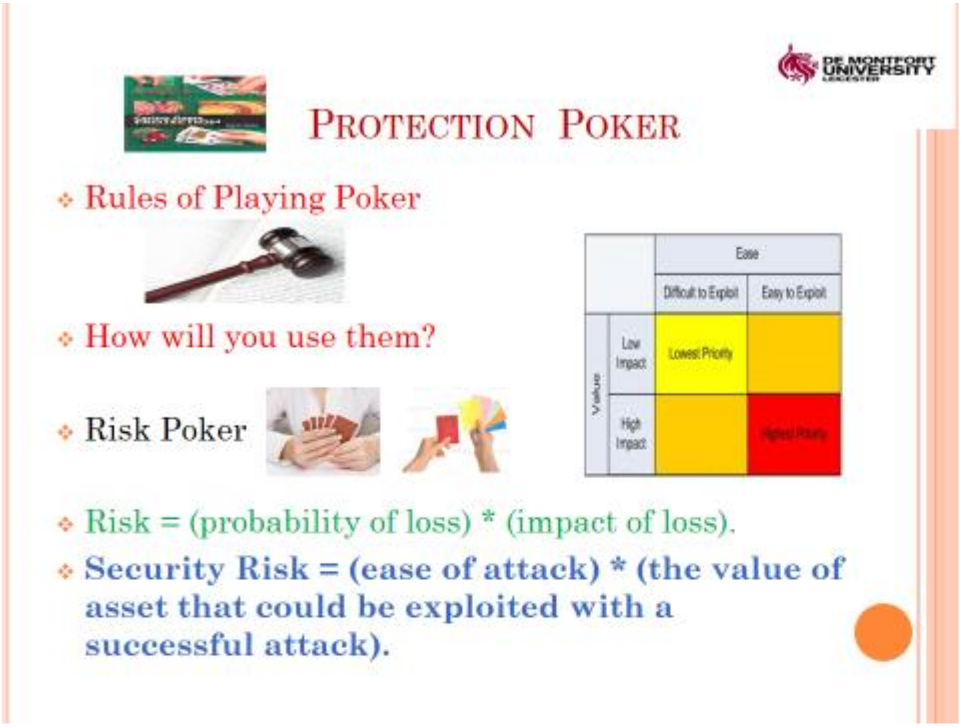
- **Now, we introduced & understood *Agile Development Methods***



Misuse Cases

threatens
Drive the Car
Steal the Car
Car Thief

- Identifying Exceptions
- Identifying Test Cases
- Design Trade-offs
- Eliciting Security Requirements
- Eliciting Safety Requirements

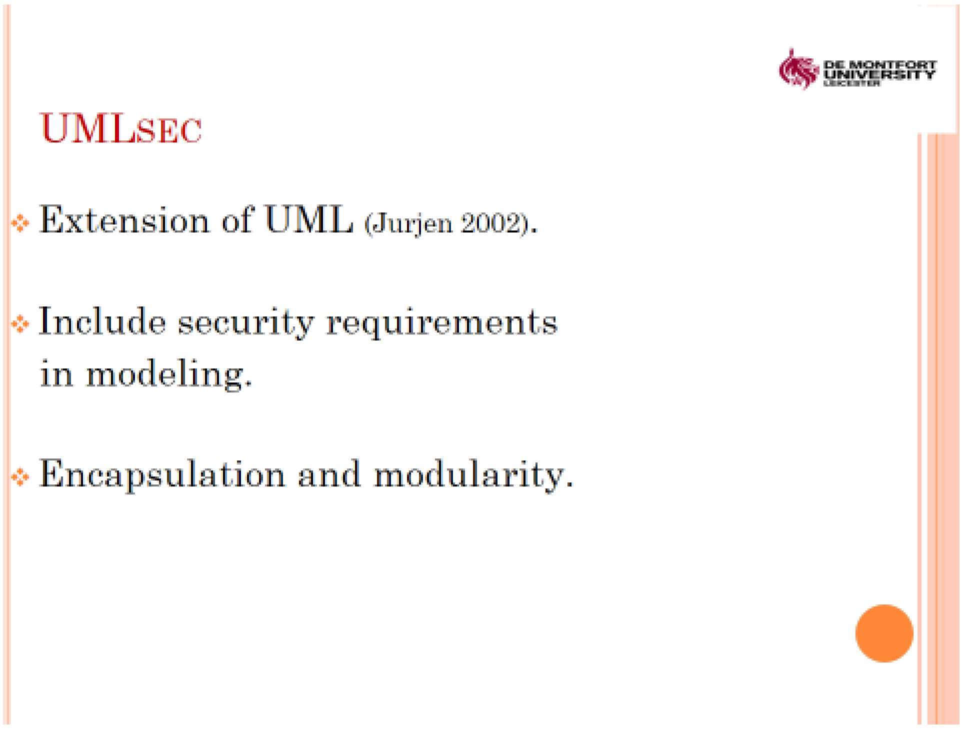


Protection Poker

- Rules of Playing Poker
- How will you use them?
- Risk Poker

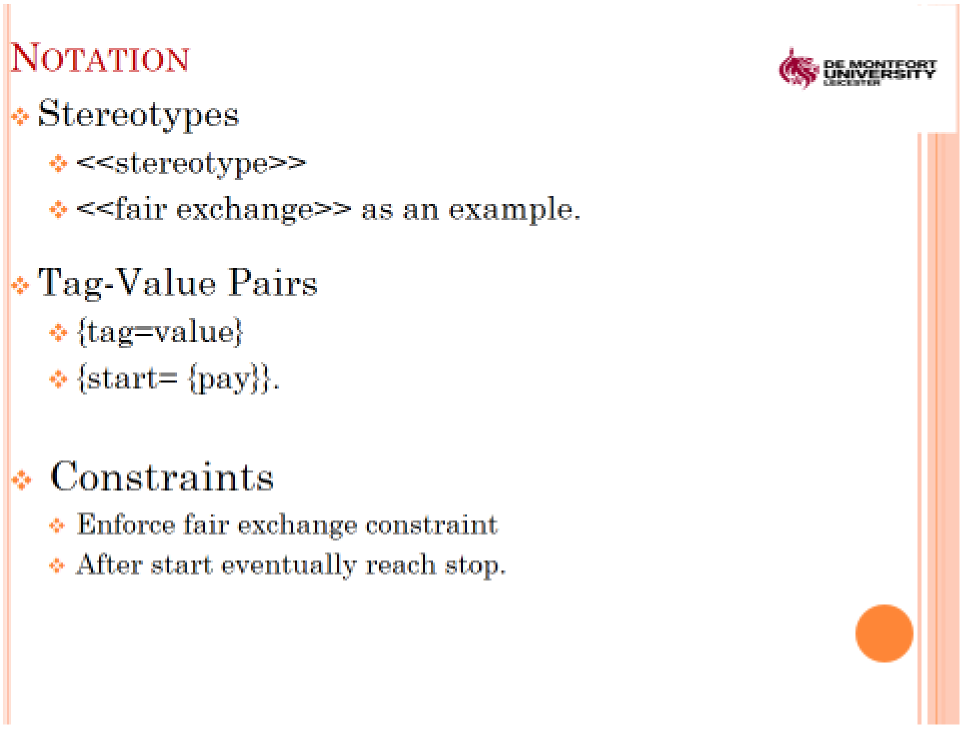
		Ease	
		Difficult to Exploit	Easy to Exploit
Value	Low Impact	Lowest Priority	
	High Impact		Highest Priority

- Risk = (probability of loss) * (impact of loss).
- Security Risk = (ease of attack) * (the value of asset that could be exploited with a successful attack).



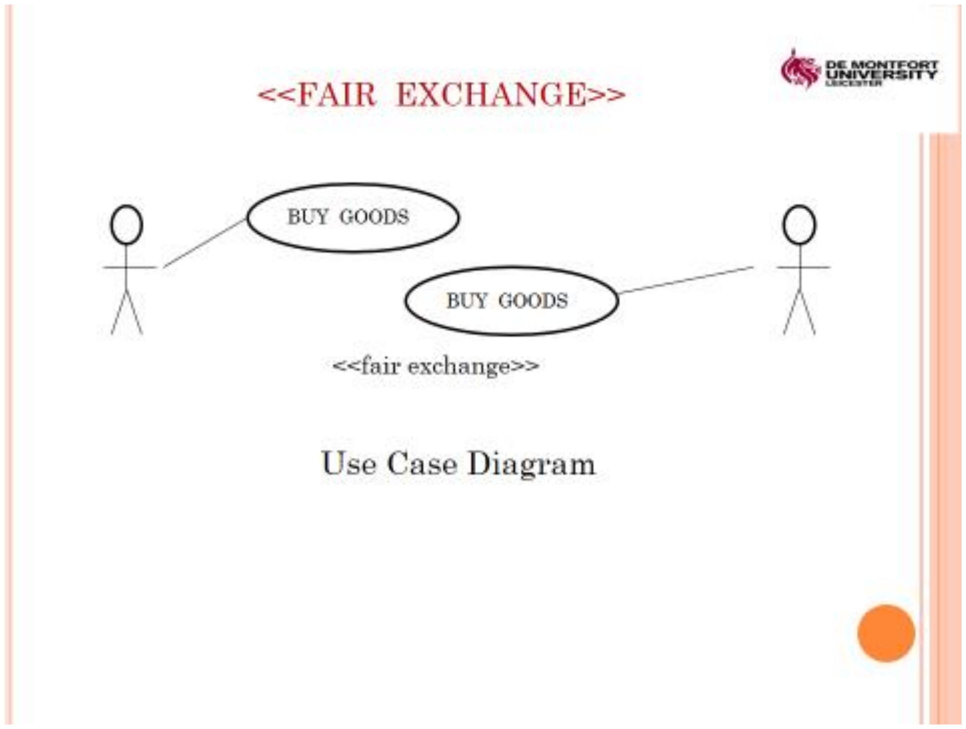
UMLsec

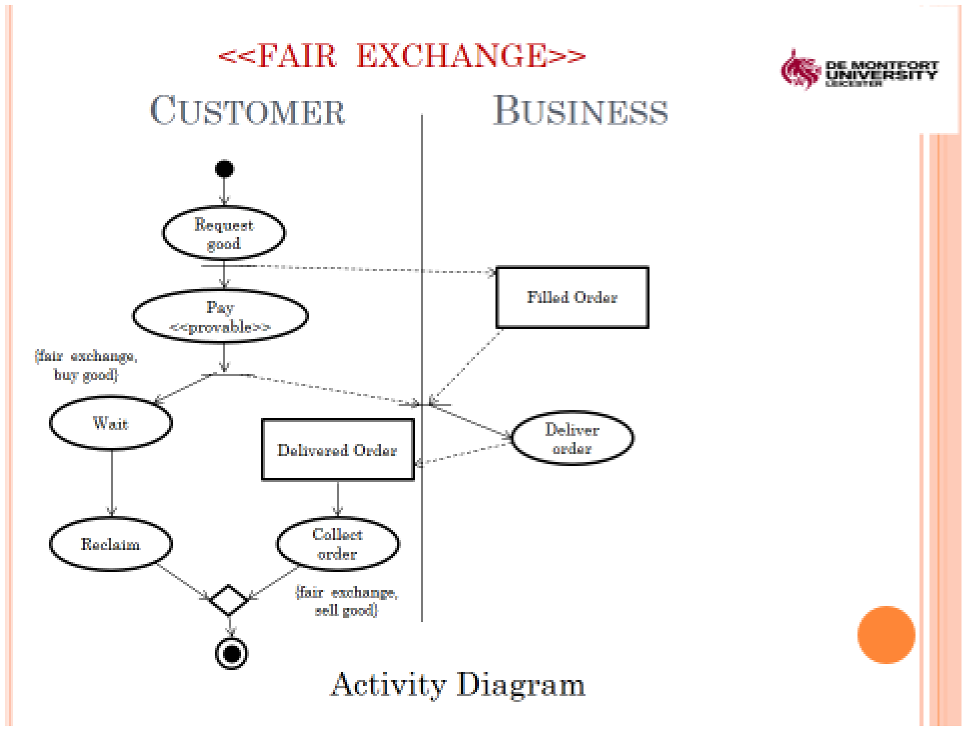
- Extension of UML (Jurjen 2002).
- Include security requirements in modeling.
- Encapsulation and modularity.

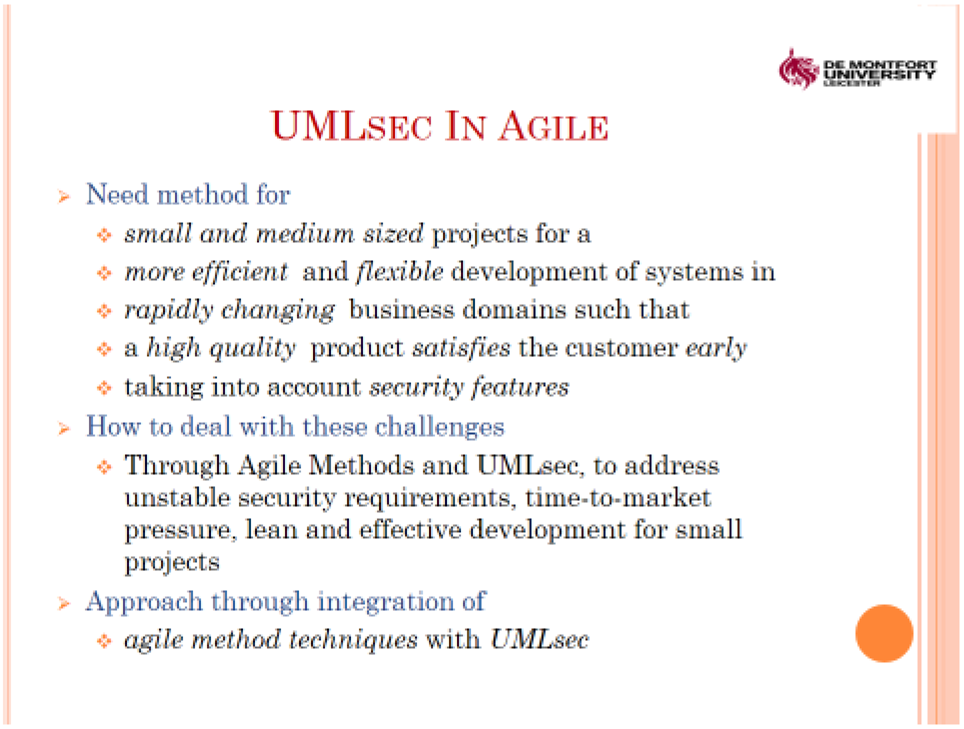


Notation

- Stereotypes
  - <<stereotype>>
  - <<fair exchange>> as an example.
- Tag-Value Pairs
  - {tag=value}
  - {start= {pay}}.
- Constraints
  - Enforce fair exchange constraint
  - After start eventually reach stop.

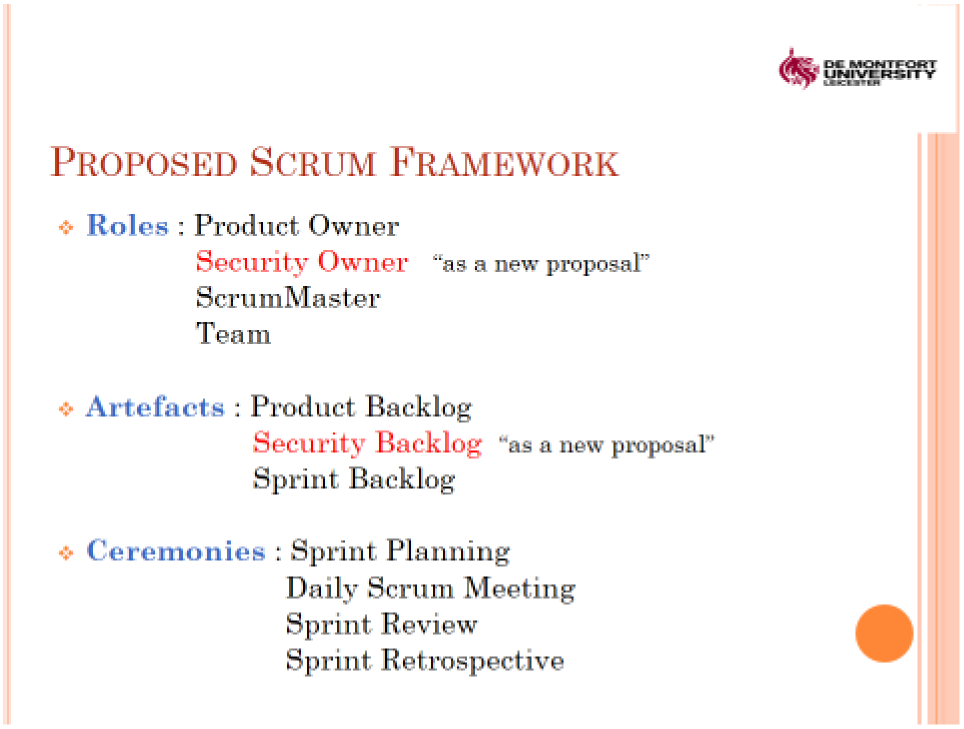

<<FAIR EXCHANGE>>
BUY GOODS
BUY GOODS
<<fair exchange>>
Use Case Diagram


<<FAIR EXCHANGE>>
CUSTOMER
BUSINESS
Request good
Filled Order
Pay <<provable>>
{fair exchange, buy good}
Wait
Delivered Order
Deliver order
Reclaim
Collect order
{fair exchange, sell good}
Activity Diagram



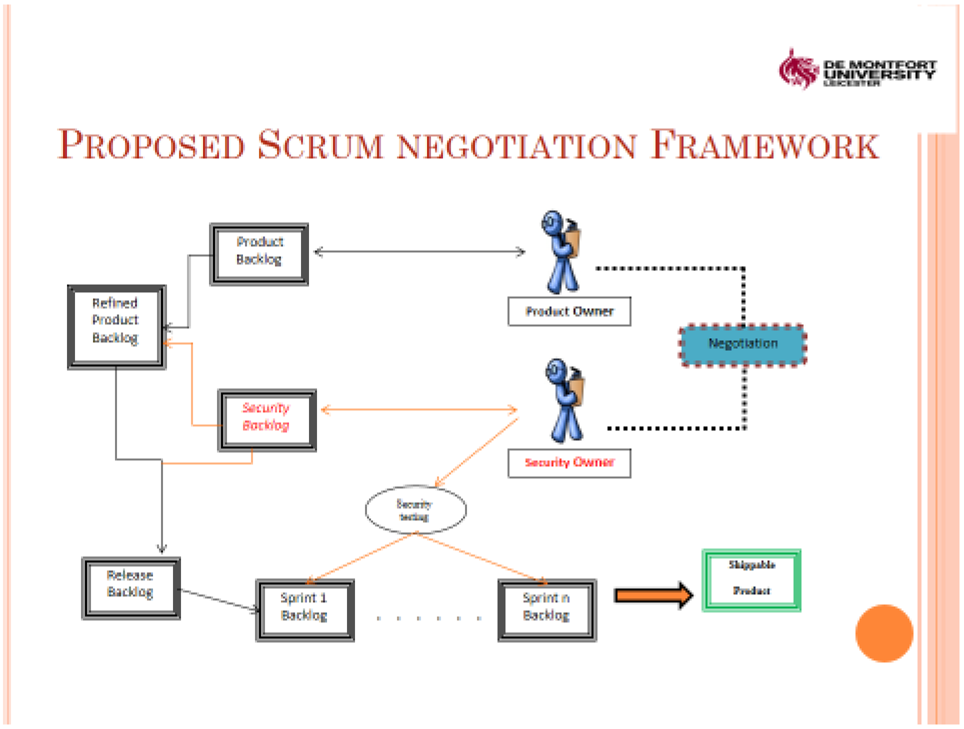
UMLsec In Agile

- Need method for
  - *small and medium sized* projects for a
  - *more efficient* and *flexible* development of systems in
  - *rapidly changing* business domains such that
  - a *high quality* product *satisfies* the customer *early*
  - taking into account *security features*
- How to deal with these challenges
  - Through Agile Methods and UMLsec, to address unstable security requirements, time-to-market pressure, lean and effective development for small projects
- Approach through integration of
  - *agile method techniques* with *UMLsec*

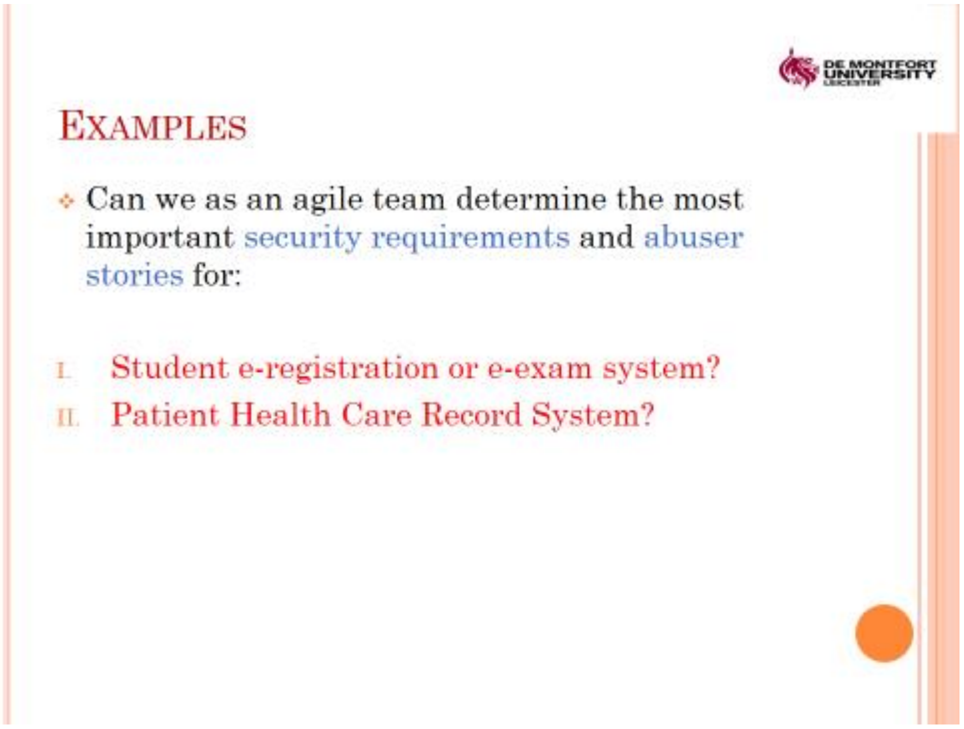


Proposed Scrum Framework

- **Roles** : Product Owner
  Security Owner “as a new proposal”
  ScrumMaster
  Team
- **Artefacts** : Product Backlog
  Security Backlog “as a new proposal”
  Sprint Backlog
- **Ceremonies** : Sprint Planning
  Daily Scrum Meeting
  Sprint Review
  Sprint Retrospective



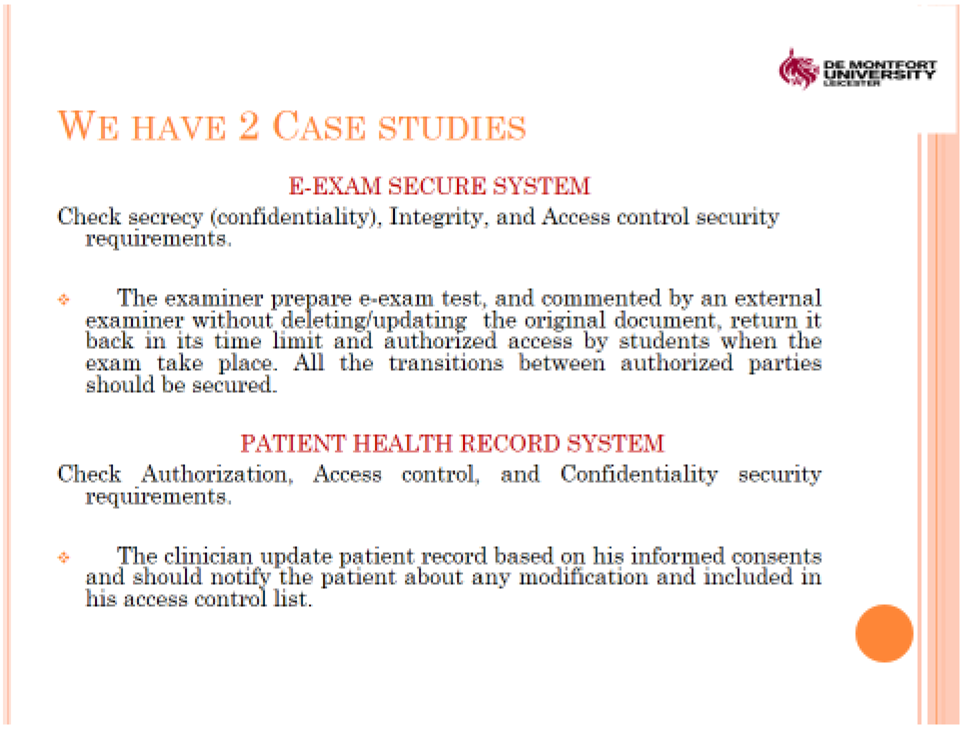
PROPOSED SCRUM NEGOTIATION FRAMEWORK



EXAMPLES

- Can we as an agile team determine the most important security requirements and abuser stories for:

I. Student e-registration or e-exam system?
II. Patient Health Care Record System?



DE MONTFORT UNIVERSITY LEICESTER

WE HAVE 2 CASE STUDIES

E-EXAM SECURE SYSTEM

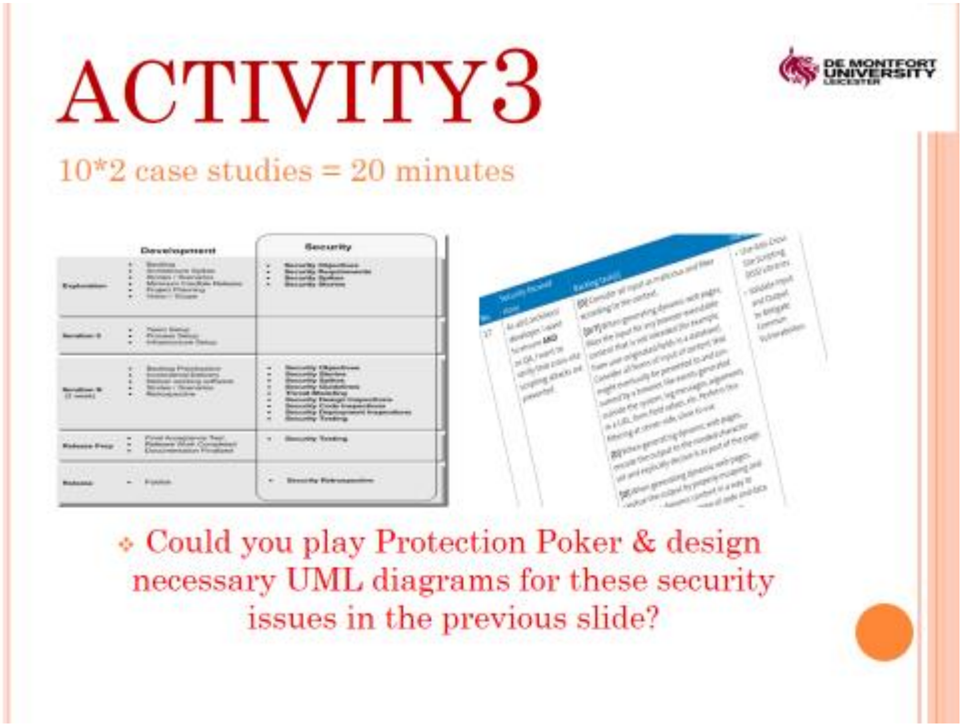
Check secrecy (confidentiality), Integrity, and Access control security requirements.

- The examiner prepare e-exam test, and commented by an external examiner without deleting/updating the original document, return it back in its time limit and authorized access by students when the exam take place. All the transitions between authorized parties should be secured.

PATIENT HEALTH RECORD SYSTEM

Check Authorization, Access control, and Confidentiality security requirements.

- The clinician update patient record based on his informed consents and should notify the patient about any modification and included in his access control list.



DE MONTFORT UNIVERSITY LEICESTER

ACTIVITY3

10*2 case studies = 20 minutes

- Could you play Protection Poker & design necessary UML diagrams for these security issues in the previous slide?

REFERENCES

1. Agile Security: Successful Application Security Testing for Agile Development, white paper (2010), VeraCode Inc, www.veracode.com.
2. COHEN, D. et al. (2004), *An Introduction to Agile Methods*. Advances in Computers, vol. 62. Elsevier Inc. 2004, USA.
3. Jurian Van de LaaR, (2012), Risk Poker: Risk based testing in agile projects, published in Agile Record magazine for Agile Developers and Agile Testers, www.agilerecord.com, issue 10, May 2012, Germany, ISSN 2191-1320.
4. Jurjens J., (2002), UMLSec: Extending UML for Secure Systems Development, In Proceedings of the 5th International Conference on The Unified Modeling Language (2002), pp. 412-425.
5. WILLIAMS, L. et al. (2008), *Protection Poker: Structuring Software Security Risk Assessment and Knowledge Transfer*, National Science Foundation, Grant No. 0716176.

Activity 1 examples:

1. Shopping Cart System:

As an e-commerce software in online marketing, shopping cart system is an example that is working on a web server to allow visitors select items for eventual purchase "online shopping". They can add preferred items to the shopping cart while shopping online or remove some items back to the shelves. At checkout, the software calculates the order total including service charges and taxes. It is business-to-consumer B2C e-shop business model which can capture client's payment with the secure gateway provider in conjunction with the secure payment gateway in order to conduct secure credit card transactions online. It must be installed on a secure server that accepts sensitive ordering information. After selecting different items from the cart, later at finalizing the transaction, the information is accessed and order is generated against the selected item, thus clearing the shopping cart. This software deals with some essential security requirements. While building this kind of software, the requirements are changing based on merchant's offers and levels of items discount prices.

2. Course e-registration System:

The online course registration system is the central part of the educational administration system, which consists of registration guidance, registration controlling, undergraduate course registration, graduate course registration, retaking and retesting, dropping the course in the middle phase and information exchange, etc. By registering the course voluntarily, the system improved registration mechanism, implemented course registration of common course for undergraduate and graduate students, and also supported the teaching activities across spring, summer and fall semesters. This software should work in a secure environment which preserves security and privacy requirements. These requirements are changeable from time to time based on the university or institute regulations and policies (ambiguous according to Product Owner needs).

3. Cash Machine System:

Automated Teller Machine (ATM) or cash machine system is a good example to handle various types of security requirements and transactions between card-holder (client) and his bank server. How can the customer deal with this machine in a secure way? Some of processes, functions, and security requirements below are crucial for both client and bank administering accounts:

- Authentication/ processing client data.
- Access controls to authorized accounts.
- Secure channel transmission between two entities.
- Calculation and management of accounts.
- Cash deposit/ withdraw and minimum/maximum applicable amounts.

4. A customer-house Specification System:

This software project is a dynamic example of a system that built incrementally based on changeable customer requirements to design his/her residence. As a prototype, the final specifications and steps will be produced at the end of this project. Safety and security requirements depend on a customer need. Product Owner can observe and modify what should be done each stage, negotiate with developers who develop the system. At the end, the customer can see the final design and a demo of building his/her house phases, the architecture, required materials, descriptions, and their estimated costs. Designing this kind of software is different from one customer to another, from big and high quality specifications required or just a little effort in designing.

5. A contracted Inventory Management System for a company X:

A company X made a contract with the software provider to design and implement an inventory management system with obvious prepared requirements which registered in the contract and the order specifications. Inventory management system is required for the company X to manage and deal with its inventory issues. The system is documentation-oriented that following clear analysis, design, coding, and testing phases. Both parties sign agreements for start and product due dates. The company X has not a representative to follow and observe developers work at the software organisation. Any update or modification of the software product will be formal through a contract between them.

Activity 3: The Two Scenarios Exercise

Development Techniques of Integrating UML Models in Agile

This section will conduct an evaluation of developed techniques that can be used to integrate UML models with agile techniques within secure systems using two cases that have been under study. In the first studied case, an illustration of developing an agile method from a specified level of a secured system downwards to a conclusive execution as a secured e-exam system is provided. Additionally, the case study gives an illustration of the analytical design that can be used to bring dynamic change to security requirements and enforcement mechanism within the paradigm that constitutes the agile as well as secured e-exam system. This case study applies a simplified example that characterizes a secure examination system. The other case study gives an illustration on the ways of using the proposed methods in specifying complicated security requirements in healthcare structures. This may entail conducting an analysis of the security needs of an approach, which incorporates the aspect of confidentiality as well as integrity policies that ensures that the privacy of patients coupled with the record integrity is protected. All agile teams may embrace teamwork when developing and modelling reliable UMLsec diagrams as well as notations for the subsequent case studies as a way of building secure systems.

E-Exam Secure System (E-ESS)

In this case study, an examination system that involves assigning each examination to three people namely an examiner, a moderating specialist as well as the external examiner. The role of the examiner is to prepare questions that will be used in the examination. Reviewing of the examination questions is conducted by a moderator as well as an external examiner. Proper security mechanisms should be put in place to ensure that the examination is not compromised in terms of integrity as well as secrecy. Particularly important, would be the care taken not to disclose the details of the examination to students.

In this case study, the initial draft containing examination questions is written by the examiner and then submitted to the moderating specialist. The moderating specialist will go through the examination questions and make comments. This should take at least ten days from the date they were submitted. It should be noted that the examination content cannot be altered when the moderator has not finished reviewing it. Upon receiving the comments from the moderator, the examiner embarks on the review of the examination questions with the aim of including them. However, it should be noted that the comments made by the moderator are not subject to change by the examiner or even an external examiner. The version that has been revised is then verified by an external examiner. An external examiner will then revise questions that constitute the examination and make comments in seven days. From all the comments made, the examiner embarks on producing the finalised script of the examination that is ideal for submission. Afterwards, questions that will be tackled in the examination coupled with the comments from the moderator as well as the comments of the external examiner are stored in a secure place where they cannot be accessed until the day of commencing the exam. For simple reasons, it is assumed that it would take four weeks after submission of the finalised version for students to sit for the exam. When the exam is going on, students who are sitting for it are permitted to read it. Other people who are eligible to read the exam include the examiner, the moderating specialist as well as an external examiner. However, it should be noted that during this period, writing accessibility is not allowed.

Patient Health Record System (PHRS)

Sensitive data concerning the health of patients is stored by healthcare systems. In the past, such information was mainly stored in form of papers and accessing them required one to place a request with the manager, who oversees security issues at the health facility. However, the introduction of technology, particularly the EPRs (electronic patient records) has proved to be of much significance. Firstly, it has increased the possibility of improving healthcare because information as well as decision supporting aids can be accessed on time. Secondly, sharing of healthcare data amongst workers in organisations is enabled. Thirdly, EPRs play an important role in supporting clinical research. However, it should be noted the EPRs can be applied practically when there is enough evidence that there is guarantee to the security pertaining to personal health data. Indeed, mishandling of patients' personal health data could compel some to shun from seeking the required healthcare.

Information Security in Health Care Systems

Medical records have massive information concerning patient's health; they include the HIV status, blood pressure, psychiatric treatment among others. The information should be kept confidential because disclosing the information illegally could cause negative repercussions on the health of the patient, social status as well as deny them a chance to be employed. It is imperative to note that when sensitive information concerning an individual reaches the public domain, it cannot be reversed thus the image is ruined. However, the information may be beneficial only when patients share it with Medicare providers as well as healthcare institutions that treat them. Indeed, physicians ought to have unrestricted access to conclusive Medicare records because of several reasons. Firstly, it enables them to make accurate diagnosis of diseases. Secondly, it is convenient since it saves time since risky and expensive tests cannot be repeated. Thirdly, it enables physicians in designing of efficient treatment mechanisms, which encompass several complex factors. Appropriate information sharing transcends personal care by including a holistic social relationship through supported Medicare research, management of public healthcare as well as law enforcement. Guaranteeing secrecy of an individual's healthcare information, while sharing information with several healthcare organisations, complicates the security of the healthcare information. Although such information could be accessed by doctors, other stakeholders, for instance, insurers, payment services and law enforcement agencies have their access restricted. It is worth to note that patients are eligible to access their personal EPRs and with the assistance of the management identify and correct any anomalies. Medical and nursing Practioners are required to seek the consent of the patient in order to use or share personal healthcare information. Although some doctors do not adhere to the rule during emergency situations, it would be imperative to keep the patient informed of any usage as well as the information that has been disclosed. Additionally, usage and disclosure of a patient's personal healthcare information can be restricted if a request is made.

Conditions for PHRS security

The PHRS has several security conditions. Firstly, all EPR should be marked using the accessible control list, which contains names of people who are authorised to read and make alterations to the information. This system is capable of preventing unauthorised people from accessing the control list and the information contained therein. Secondly, clinicians can access the EPR on their own or using patients that can be accessed on the control list. In the event that a patient is referred, he or she can access the record with the clinician who can be accessed on the control list. Thirdly, one clinician on the accessible control list should be marked as responsible. It is the responsibility of such a clinician to make changes on the ways of accessing the control list by increasing the number of healthcare specialists to the list. Furthermore, the clinician who has been marked to be responsible should inform the patient about the names that are available, those that have been added on the control list if there is transfer of responsibility. Additionally, the consent of the patient should be sought; however, in emergency and statutory exemptions.

Moreover, without the expiry of the set time, clinical information cannot be deleted by anybody. In addition, any accessibility to the medical records should be marked using the name of the subject and date as well as time. An audit record containing all deleted data should be kept. Finally, effective mechanisms for preventing the acquisition of aggregate information on personal health will be included in PHRS security system. Particularly important would be the decision to inform patients about the proposed persons that are to be added to the list of controlled access.

Reference:

Siewe, F. (2005), *A Compositional Framework for the Development of Secure Access Control Systems*, PhD Thesis, Software Technology Research Laboratory, Faculty of Computing Sciences and Engineering, De Montfort University, England.

B. Questionnaire

B1: Pre-Experiments Questionnaire

Pre-experiment Questionnaire

In order to determine you eligibility for the study please answer the following questions. Please circle as appropriate.

Please answer the following questions, circle as appropriate.

1. Do you have experience or knowledge of agile software development methods? YES / NO
2. Do you have experience or knowledge of Scrum? YES / NO
3. Do you have experience or knowledge of security in development of software? YES / NO
4. Do you understand the significance of security in agile development? YES / NO
5. Do you have experience in modelling UMLsec? YES / NO
6. Do you have experience in modelling UML? YES / NO

B2: Post-Experiments Questionnaire

Post-experiment questionnaire

Please answer the following questions about the experiment you have just completed. Please circle answers as appropriate.

Section 1 - Effectiveness of UMLsec

1. You found it easy to identify security requirements

strongly agree	*agree*	*neutral*	*disagree*	*strongly disagree*

2. You found it easy to prioritise security requirements

strongly agree	*agree*	*neutral*	*disagree*	*strongly disagree*

3. You were able to manage security requirements effectively

strongly agree	*agree*	*neutral*	*disagree*	*strongly disagree*

4. You found it easy to model security requirements

strongly agree	*agree*	*neutral*	*disagree*	*strongly disagree*

5. The methods you used for modelling security requirements were appropriate

strongly agree	*agree*	*neutral*	*disagree*	*strongly disagree*

Section 2 – Effectiveness of Security Owner

6. **You were helped to identify security requirements**

strongly agree *agree* *neutral* *disagree* *strongly disagree*

7. **You were helped with modelling security requirements**

strongly agree *agree* *neutral* *disagree* *strongly disagree*

8. **As a team you could effectively implement security into the product**

strongly agree *agree* *neutral* *disagree* *strongly disagree*

9. **You received effective advice about security requirements to help with security requirements management**

strongly agree *agree* *neutral* *disagree* *strongly disagree*

10. **You found that you and the team could effectively negotiate security requirements with the Product Owner**

strongly agree *agree* *neutral* *disagree* *strongly disagree*

11. **There was effective team work regarding security requirements in the Scrum meetings**

strongly agree *agree* *neutral* *disagree* *strongly disagree*

12. **There was effective communication about security requirements in the Scrum team**

strongly agree *agree* *neutral* *disagree* *strongly disagree*

13. **The team effectively identified security requirements**

strongly agree *agree* *neutral* *disagree* *strongly disagree*

14. The Product Owner effectively prioritised security requirements

strongly agree *agree* *neutral* *disagree* *strongly disagree*

15. Security requirements were effectively managed by the Scrum team

strongly agree *agree* *neutral* *disagree* *strongly disagree*

16. As a team you could manage your own security requirements / considerations without assistance

strongly agree *agree* *neutral* *disagree* *strongly disagree*

17. The agility of the team was maintained in the agile process

strongly agree *agree* *neutral* *disagree* *strongly disagree*

18. You required help with the technique you used for modelling security requirements

strongly agree *agree* *neutral* *disagree* *strongly disagree*

19. In consideration of the approach to security requirements that you engaged with in the experiment, what are your opinions about the security requirements consideration?

____

____

____

____

20. Please give you overall opinion of the team and how well you work with them and whether you think they were effective in modelling security.

____

____

____

C. Ethics and Consent Forms

C1. Survey Questionnaire

Dear Participant,

We would like to ask you to participate in the data collection for the UMLsec in an agile Project. UMLsec in an agile project, which consider security requirements in modelling will investigate and evaluate adopting UMLsec for the proposed extended Scrum agile methods in order to analyse and evaluate ethical issues arising from these. By including a variety of stakeholders and disciplinary perspectives in the evaluation experiment, it aims to evaluate and address the most ethical issues identified and see the applicability of using UMLsec in agile methods.

The purpose of this survey is to validate these expectations and ask people involved in AgileUMLsec development, whether these views are correct.

Your participation in this study is entirely voluntary. The survey includes several questions. Filling in the survey will take no more than 10 - 15 minutes.

Your name or any other personal identifying information will not appear in any publications resulting from this study; neither will there be anything to identify your place of work or the project you are involved in.

Even though the study findings will be published in international conferences and journals, only relevant researchers will have access to the survey data itself. These researchers will be bound by the principles outlined above. There are no known or anticipated risks to you as a participant in this study.

If you have any questions regarding this study or would like additional information, please contact AgileUMLsec researcher (p10459491@myemail.dmu.ac.uk).

By filling in this survey you indicate that you understand its purpose and consent to the use of the data as indicated above. Should you decide not to complete the survey, the data you have entered up to that point will be used, unless you indicate otherwise in questions 1 and 2.

Many Thanks for your time, support and cooperation.

Minahi Alotaibi, PhD researcher at STRL; Technology, DMU.

1. I agree with the use of my responses for research purpose of the AgileUMLsec project as outlined above.

○ Yes

○ No

2. I agree to the use of anonymised quotes from my response for research and publication purposes

○ Yes

○ No

C2. Consent Form

Title of Project:

Modelling Security Requirements In Extended Scrum Agile Development Framework

.

By: Minahi Alotaibi **– Research student @** De Montfort University (STRL),

I confirm that I understand the research study and have had the opportunity to ask questions. Also, I understand that my participation is voluntary and that I am free to withdraw of any point, without giving reason. I agree to take part in the above study

.

Name of Participant	**Date**	**Signature**
Minahi Alotaibi		Minahi
Name of Researcher	**Date**	**Signature**