ACHIEVING APPROPRIATE SOFTWARE SECURITY LEVELS WITH AGILE SOFTWARE DEVELOPMENT

GEOFREY GATINO KAGOMBE

MASTER OF SCIENCE

(Software Engineering)

JOMO KENYATTA UNIVERSITY

OF

AGRICULTURE AND TECHNOLOGY

2023

Achieving Appropriate Software Security Levels with Agile Software Development

Geofrey Gatino Kagombe

A Thesis Submitted in Partial Fulfilment of the Requirements for the Degree of Master of Science in Software Engineering of the Jomo Kenyatta University of Agriculture and Technology

2023

DECLARATION

This thesis is my original work and has not been presented for a degree in any other university.

Signature………………………………… Date ……………………………….

Geofrey Gatino Kagombe

This thesis has been submitted for examination with our approval as the university supervisors

Signature………………………………… Date ……………………………….

Prof. Ronald Waweru Mwangi, PhD

JKUAT, Kenya

Signature………………………………… Date ……………………………….

Prof. Joseph Wafula Muliaro, PhD

JKUAT, Kenya

DEDICATION

To Jesus Christ my Lord and saviour. For His mercy and grace up to this point. To him be the glory forever! Amen

ACKNOWLEDGEMENT

First and foremost, I give my deepest and most sincere gratitude to the almighty God for granting me this opportunity, sustaining me, and bringing me to this end. If it was not for His grace, it would not have been possible.

I would also like to express my deepest appreciation to my supervisors, Prof. Waweru Mwangi and Prof. Wafula Muliaro. They have patiently guided me, giving feedback and direction to the very end. In the same vein, I thank the JKUAT Department of Computing for various forms of support given to this research.

I also thank my family for giving me the time to concentrate on this research. I express my greatest gratitude to my wife Naomi for taking care of the family while I dedicated my time to this research. I thank my lovely daughters for their prayers and love while I concentrated on this work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

LIST OF ABBREVIATIONS/ ACRONYMS

Agile SE	Agile Software Engineering
CMM	Capability Maturity Model
SSAM	SSE-CMM Appraisal Method
SSE	System Security Engineering
SSE-CMM	System Security Engineering Capability Maturity Model

ABSTRACT

Software security for agile methods is still a major concern. Security has become an integral component of software quality in today's world. This is influenced by the criticality and amount of data the software handles and the volatility of the environment of deployment, e.g. the cloud. In addressing this problem, this research proposes a secure agile software development framework that conforms to standard industry best practice in software security engineering. Agile methods have taken over the software development industry, mainly due to their ability to deliver timely and quality software. Research has also shown that most agile methods are not equipped to handle security and assurance in the developed software. A review of literature conducted in this thesis confirmed the lack of security practices in agile development methods. This research uses Design Science Research (DSR) to build, test and evaluate an agile Security engineering framework. It involved a rigorous process to design an agile security framework to solve the observed problems by ensuring that security is part of the development process from the beginning of the project to the end. It was modelled after standard security engineering models targeting the intended security goals. The security framework is agile, meaning it adheres to agile principles. A multiple-case study in an academic and industry setting is conducted to demonstrate and evaluate the utility of the methodology. The evaluation criterion for security capability was Systems Security Engineering Capability Maturity Model (SSE-CMM) Appraisal Method (SSAM). The agility of the resulting process was evaluated using the four-dimensional analytical tool (4-DAT) and it showed satisfactory compliance of the methodology with agile principles. The main contributions in this thesis are: the secure framework, which entails description of the concepts, a pre-game risk analysis, security engineering stages, tasks, tools and techniques; generation of a quality theory on practices that promote quality in a software development environment. This research would be of value to researchers as it introduces standard security components of software quality into an agile software development environment, probing more research in the area. To software developers, the research has provided a secure agile framework that builds security and assurance into the product. This would be a first step towards standardisation of the developer's process model as a secure process.

CHAPTER ONE

INTRODUCTION

1.1 Background

Software security is the idea of engineering software so that it continues to function correctly under malicious attack (McGraw et al., 2011). This discipline encapsulates fundamental principles that when properly employed would give users an acceptable level of confidence that the system is fit for purpose (Irvine & Nguyen, 2010). Its best practices leverage good software engineering practice and involve thinking about security early in the software life cycle. This translates to knowing and understanding of common threats (including language-based flaws and pitfalls), leading to designing for security, and subjecting all software artefacts to thorough objective risk analyses and testing.

The need for secure, trustworthy systems cannot be overemphasised, especially in today's world. Today's information systems have a variety of characteristics that raise security concerns. These range from the sensitivity and cost of data and transactions handled to the ever-evolving geographical size and distribution complexities for example the number and types of components and technologies involved, among others.

The common practice in the software development world was fixing security into systems after production, but experience has proved this to be costly. It has therefore become generally accepted that Software Assurance should be built into a system as it is being developed (Irvine & Nguyen, 2010) and thus software security engineering valuably comes to play. This means that as software is being built, the developers should consider or determine the pool of attacks that pose as threats to the particular software and the data or the property it handles and think of ways of building a system that has high probability of repelling the attack.

The increased prevalence of agile methodology in emerging software development and deployment environments has brought new concerns in building secure systems.

This is mainly because the traditional literature and methods of Software security engineering provide a very complete and exhaustive guide to security objectives. These objectives include; to understanding the threats in the environment, determining relevant controls, and implementation of these controls to minimise the expected risk. Unfortunately, these artefacts are found to take security teams so much time to produce, that by the time they are delivered, the software build in question was already deployed and the next had begun (Taati & Modiri, 2015). The nature of agile development with a framework like Scrum requires a time-boxed "sprint," whose end date does not change. The build is delivered on that date, and any portions that are not complete get removed and put back into the backlog. The pace of such a project means that IT security personnel are forever catching up to the latest release. These security measures in their traditional form are just not agile and hence present a general conflict of cultures.

In seeking to prioritise customer satisfaction, agile processes aim towards producing customer value within iteration. Security requirements end up being denied the same citizenship status or importance as reliability and performance requirements, since most users are not aware of software security issues.

To develop secure software, developers are supposed to execute a set of security engineering activities within the software development process (Merkow, 2019; McGraw 2012; Ransome & Schoenfield, 2021). Security engineering standards and processes are therefore designed to be coordinated within the SDL. When introduced in their traditional state in agile environments, security engineering activities tend to create production bottlenecks (Rindell et al., 2021). These bottlenecks might include the production of additional documentations, performing security related reviews, scans and tests. In addition, most traditional security engineering (e.g. SSE-CMM, CoC) advocates for strict process adherence and heavy upfront requirements. This implies that there is a need for research in this area in order to have efficient coordination of security activities in the agile space. Research in this area will promote security-enhanced Agile processes and practices and the skilled people to manage them and perform them.

System Security Engineering (SSE) provides the activities that can be carried out during software development to ensure the security of the resultant product and the processes to be followed. There are no generally acceptable security activities, but there has been credible and sufficient work by different bodies in this area. SSE-CMM is a standardised example of such work.

The System Security Engineering – Capability Maturity Model (SSE-CMM) original work and project infrastructure was sponsored by NSA (National Security Agency) with collaboration from several industrial players with an aim to develop a CMM for security engineering (Ferraiolo, 2000). The International Systems Security Engineering Association (ISSEA) was later selected by the SSE-CMM project to continue support. It oversees further development and use of the model (Association, 2002). SSE-CMM aims to improve and assess security engineering capability. It examines the maturity of the Security processes implemented by an organisation or in a project. It was built exclusively for security purposes, and it has also been adopted as an ISO standard. The major problem is that most existing security engineering standards, including SSE-CMM, were created with the traditional software development life cycle (water fall) in perspective, and therefore they are incompatible to agile techniques due to the difference in culture.

In this study, an agile security engineering framework is introduced, and tested on a sample group of developers. The framework adopts the SSE-CMM security engineering framework as a high-level model framework. Note that, as mentioned earlier, SSE-CMM with its heavy and complete documentations and perceived weight to the process presents a conflict in culture with agile methods. Nevertheless, the idea is not to fully adopt it but to acquire proven guidance on principles and goals on how to get secure software, based on the fact that these standards embed knowledge of the necessary activities required for a system with security assurance.

Although SSE-CMM provides the base framework, this work has not been limited to it, it has built upon contributions from several other reputable works (e.g. Correctness by Construction (Kourie & Watson, 2012), Kanban (Terlecka, 2012), etc) to improve and come up with the agile security framework. The final work has influence from many relevant literatures (including CMM’s, related security engineering documents as well as other researchers’ works) discussed and compared in the literature review (chapter 2).

It is important to insist that not all security activities given by SSE-CMM can be agile or even implemented in a lean way. IT security engineering activities are generally costly in terms of time and money (Maqsood & Bondavalli, 2020), they call for detailed documentation that means they are heavy, directly contrasting the lean approach of agile methodologies. As already mentioned, the introduction of these security activities introduces bottlenecks in the agile process (Rindell et al., 2021). On the other hand, agile methods are characterised by strict timelines e.g. time boxed sprints in scrum and these bottlenecks would therefore threaten the agility of the process. Another example is, agile methodologies e.g. scrum do not harbour an avenue to gather security requirements through risk analysis. If this is introduced in its traditional form it would mean static, heavy requirements upfront and when introduced later it would mean additional cost in terms of time and even sometimes extra manpower. At the same time, discarding some important activities and processes might render the whole framework ineffective as far as security is concerned. A trade-off between agility and security has to be struck for success to be achieved. The choice of the word “appropriate security levels" was to avoid a total loss of agility in seeking to achieve maximum security. This work seeks to achieve basic maturity levels for a security engineering development process. It investigates whether an agile process can consider security requirements and implement them during development without necessarily compromising on agility to the extent of instilling customer confidence in using the software. In increasing security there is a giveaway in that some level of agility will be lost (Keramati, & Mirian-Hosseinabadi, 2008). This work focuses on the process and will measure the security levels in terms of a display of capability maturity of the process. Also, the measuring gauge of how much security activities will be loaded on the process depends on the agility of the overall process.

This research builds upon known practices and methods to achieve security, making the necessary changes to maintain agility on the overall process.

1.2 Problem Statement

There has been a significant adoption of agile software engineering methodologies in the software industry, indicating a cultural shift from traditional development methodologies. In the software world, the main reason for this culture shift is mainly because of;

- Agile methods involve the customer or a representation of the customer throughout the process, and thereby promote a sense of project ownership to the customers.
- Agile technique's ability to adapt quickly to customer's changing requirements significantly reduces chances of failure to meet user requirements.
- There is also a significant increase in the general productivity of teams. Teams can deliver software products on time and within budget, which has been a major problem, especially when software development contracts are involved. (Pries-Heje & Pries-Heje, 2011; Bellenzier et al., 2015)

Security engineering in the agile environment has generated a lot of discourse within industry, with most literature in this area coming from practitioners and consultants. The lack of security practices in agile methods, is corroborated by a number of researchers (Baca et al., 2015; Aguda 2016; Ghani, Maqsood & Bondavalli, 2020b; Singh et al., 2021). Insecure software development methodologies build insecure software products (Homaei & Shahriari 2019). The vulnerabilities that emerge from modern software deployment environments and systems need to be addressed with more caution to avoid catastrophic ends.

The weaknesses of agile methodologies in engineering secure software can be attributed mainly to the fact that security engineering activities have not been given residence in agile techniques (Mirza & Datta, 2019). In short, two key issues stick out:

1. Security requirements have not been given citizenship in these techniques, as functionality and user requirements.
2. The traditional security engineering e.g., from SSE-CMM activities cannot fit into the agile lifecycle without affecting agility i.e., there is a conflict of cultures.

Because of the mentioned reasons, agile techniques as per the status quo do not possess the ability to give software assurance and hence, from the process itself, promote customer confidence in the final product as secure.

Most of the existing research in the area addresses phases in the process but not the whole development circle (Alotaibi, 2015; Koc & Aydos, 2017). Fewer address the issue of building software assurance (Rindell, 2021). Another gap in research for this area is a lack of concrete guidance for application (Hollar 2006; Bernabé, Navia & García-Peñalvo, 2015; Agarwal & Umphress, 2008; Dzhurov, Krasteva & Ilieva, 2009).

To solve this problem, this work has designed a secure agile security engineering framework after the SSE-CMM, which is a standardised model for SSE. It approaches SSE by breaking it into three phases: risk, engineering and assurance, hence covering the entire process from inception to the end. The main contribution in this framework is the agile initial risk analysis, the tools, tasks and techniques. The generation of knowledge on how to enforce security as a quality in the developed software comes in as a second contribution of this work. These contributions are discussed in more detail later in subsequent chapters.

1.3 Research questions

To deal with the above problems, the research came up with the following research questions (RQ) to be tackled in the course of the work :-

RQ. How can an agile development framework that conforms to recognized software security standards be designed to ensure the development of secure software products without losing agility of the overall process?

To answer this question (RQ), the following sub-questions were tackled:-

SQ1. What Software security engineering standards and best practices exists?

SQ2. What standard software engineering activities and techniques are essential for developing secure software?

SQ3. How can the goals achieved by the activities and guidelines provided by the standards be achieved in an agile manner?

SQ4. How can these activities be implemented in an agile environment (e.g. scrum) without losing agility in the resultant process?

SQ5. How can the resultant process be evaluated?

1.4 Objectives

The main objective of this study is to improve the process of software development using agile techniques by coming up with a framework that will incorporate standard security artefacts into the life cycle without losing agility. This research seeks to achieve the basic level of security engineering maturity as “appropriate” this being a pilot attempt.

To achieve this, the following specific objectives will be involved:

1. *Explore existing security engineering frameworks, standards and literature in order to fully expose the gap between existing security engineering and agile methodologies, and also review existing solutions from previous research. The intention was that, the understanding gained, will influence the high-quality design of the framework.*

2. *To design a framework that ensures standard security engineering goals are met within the agile environment. The framework should cover the entire engineering process from inception to completion without losing agility.*
3. *Implement the framework in an agile environment (scrum) to produce a secure agile development process.*
4. *Evaluate the resultant process in an industrial setup to demonstrate utility as well as assess security capability of the process as well as agility.*

1.5 Justification

This research is warranted by the need for secure and trustworthy software. The increased popularity of agile techniques in software development coupled with the increased sensitivity of software and the data and function has warranted this research. Agile Scrum is the most popular agile method at the moment. The 15th state of agile Survey (2021) showed that about 94% of the 4,182 respondents applied agile techniques. This indicated acceleration from 37% increment in 2020 to 87% in 2021. Some of the reasons for this increment are; better management of priorities, accelerating software deliveries, increased team productivity, among others. The adoption of scrum has also increased, with 66% of the respondents claiming to use scrum and a further 15% using some variants of it. It also reports that organisations adopting agile methodologies meet the goals that they had set out to achieve in going agile, however they also noted ample room for improvement where organisational agile maturity is concerned. As most developers are moving towards agile there is a trending decline of developers who are mastering the traditional waterfall SLC, in short Agile is the future. Therefore, it is vital for it to be made fit for all types of software, including security critical systems.

It is important to note that the traditional approaches had some level of success especially where discipline was required for example the requirements, designs and plans were made available early and constant monitoring was done to force the projects to conform to plans, they also provided for easy certification against external assurance standards. Although this is true, some literature, e.g. Beznosov and Kruchten (2004), agree that there are still major problems associated with them because the success rate of software projects subjected to these traditional approaches have proved to be low, actually less than 50%. This is mainly due to changing user requirements, not meeting agreed timelines and escalated costs. This is where agile methodologies come in but the question that arises is; can we improve agile techniques to a level where their projects can still enjoy the benefits of the traditional approaches without losing agility if it is to some acceptable extent.

There is adequate literature on the challenges of agile techniques to develop secure software for instance, Boehm & Turner (2005) listed among others developing safety critical software as a challenge to agile processes. Beznosov and Kruchten (2004) evaluated the mismatches between security assurance methods/techniques and agile practices. The general issue is basically a conflict of culture between the agile methodologies and security engineering activities provided by standards e.g. SSE-CMM. This work attempts to bridge the gap between these two by coming up with a framework for agile security.

Though several researches have attempted to address the problem of security in agile, none has pegged their approach with a standardised approach as SSE-CMM or Common Criteria for IT security (CC). The proposed framework aims to meet standardised goals by trying to incorporate standard activities or hybrid versions of the same. Some expected benefits of the framework would be; Heightened user trust in the process as well as the final product, and also it is an initial phase towards attaining security engineering capability maturity through agile.

1.6 Scope

The focus of this research is on agile methods, traditional waterfall methodologies are out of the scope of this research and will only be mentioned when there is need for comparison or learning. The investigation is further narrowed to SCRUM because the researcher's experience with the methodology as well as its popularity makes the work more relevant within the software industry. All the other agile methodologies are out of scope for this work.

Software security engineering also involves organisational and process administrative roles which involve things like setting policies, training, and quality controls among others. In this work the focus is on the actual security engineering process involving risk analysis, security engineering process and assurance development. This excludes the organisation's roles in security engineering from this work.

Capturing the right requirements and ensuring implementation is critical in software development (Singh, 2011; Martin, 2013), and a key pillar of agile software methods. Therefore, this work will cover the whole development circle from capturing security requirements, to ensuring that they are implemented amidst a cacophony of voices poised as customer needs (Daneva & Wang, 2018b), and ensuring assurance is given through evidence of implementation and tests. Suggestions from other researchers on the best practices that will work well with the proposed framework might be mentioned.

1.7 Limitations of the Study

The prior knowledge on which the secure agile framework is premised was obtained through documents searched electronically. This implies that unpublished documents and those not indexed by the databases, or the databases only allowed access to paid subscribers were not included. The security engineering framework will only mirror those studies that were included in the systematic literature survey.

Another limitation is that it is not possible to separate the experience or capability of the developer(s) who were involved in the case study from the quality of the methodology. The experience of the team, methodology in use and the project environment will determine the quality of software. The experience of the team can neutralise the weakness of the methodology and deliver a high-quality product. In attempting to mitigate this, a triangulation of sources was carried out by interviewing other developers in seeking out their opinions, but this might not be enough. Another limitation is that this research did not define any quantitative metrics for evaluating the impact of quality and security practices on the application programmes designed using this methodology. This research used practices that have been proved to be effective by other researchers, therefore proving each practice’s effect on the quality of the software of the product is outside the scope of this research.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

The purpose of this chapter is to give a detailed understanding of the problem area and to present arguments in favour of the introduced approach through relevant literature on the specific area. The chapter is divided into four major parts; the first is a general discussion on agile processes, then it narrows down to Scrum. The second part is a discussion about the software security process, building an understanding from the original standard heavyweight software security frameworks that exist. It includes the SSE-CMM as a major element, since as already mentioned it is a significant framework in this work. Also discussed, is Capability Maturity Model Integration (CMMI) and how it can be used to instil discipline to agile processes in the development of security critical systems. A survey of literature contributing to security in agile software is also added. The knowledge built by the literature influences the design and logic behind this framework. Lastly, an overview of the framework is discussed, presenting its three major areas.

2.2 Agile Software Development Methods

Lightweight software development methods (later came to be known as agile software development methods) have been in existence since the early 90s. They evolved in the mid-1990 as a reaction against the heavyweight, process driven methods (Poppendieck & Poppendieck, 2003). Though the traditional methodologies provided a disciplined mechanism of software engineering, they were still blamed for late delivery of software, plus they could not accommodate change in requirements. The major arguments against these Classical methods of software development are; huge efforts during the planning phase, poor requirements' conversion in rapidly changing environments and treatment of staff as a factor of production.

The fact that requirements change during the process of development posed concerns because traditional software development models were too rigid to accommodate these changes, and when they were forced upon them, it resulted in increased budgets and missed timescales. The high cost of change and rigidity to changing requirements drew a possibility of building software that does not meet user requirements. This is a major reason Agile methodologies were conceived in the first place and also why most software development companies are shifting to Agile today. Barry Boehm and Turner (2004) stated that as software proceeds through its lifecycle the cost of making a change becomes larger with ratios of 100:1 for making a change after delivery versus project starting are common. These ratios are usually dependent on individual projects, but it is true for most projects that the cost of change will be higher as the process advances (Barry Boehm & Turner, 2004). Agile methodologies facilitate change accommodation through iterations and at the same time promote high involvement of customers, hence they have low cost of change. Other characteristics of heavyweight methods that received criticism were heavy regulations, regimented and micromanaged among others. Agile software development methodologies were introduced to specifically address these challenges.

In recent times, there has been a lot of excitement about agile and agile software development methods. The software world is witnessing a shift from the traditional to agile, with major players in the industry adopting these methodologies or hybrids of agile and the traditional waterfall SDLC development. Many researchers have actually tried to define agile and agile software development, but still by the time of publication no common definition had been agreed upon.

Mushtaq and Qureshi (2012) defined agility as the capacity to have intellectual response to business forecasts with the intent of remaining aggressive and inventive for any unstable and swiftly shifting business environments. Conboy and Fitzgerald (2004) through study of relevant materials came up with a similar but slightly expounded definition that captured some keywords that express the spirit of agile software development. His definition can be applied across the board regardless of the method. According to them, agility is the continual readiness of an Information system development method to rapidly or inherently create change and learn from change while contributing to perceived customer value (economy, quality and simplicity) through its collective components and relationships with its environments.

Qumer and Henderson-Seller, after carrying out a survey and assessment of various literatures including the agile manifesto (Beck et al., 2001) offered the following definition for the agility of any entity:

"*Agility is a persistent behaviour or ability of a sensitive entity that exhibits flexibility to accommodate expected or unexpected changes rapidly, follows the shortest time span, uses economical, simple and quality instruments in a dynamic environment and applies updated prior knowledge and experience to learn from the internal and external environment*." (Asif Qumer & Henderson-Sellers, 2006b).

From the aforementioned definitions, it is easy to note that they all agree an agile software development methodology must bear some key elements that channel it towards flexibility, speed, leanness, learning and responsiveness.

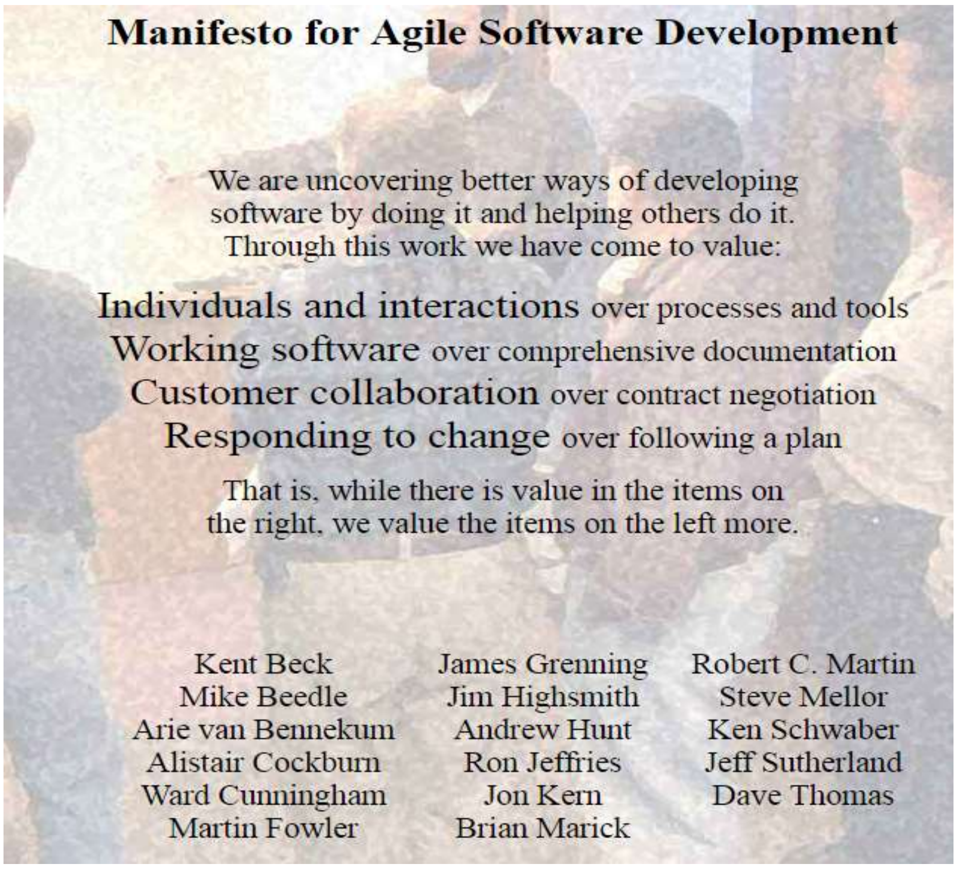
Some of these key elements are iterations, incremental development, and special team structures. The Agile Alliance describes Agile software Development as a group of frameworks and practices of software development methodologies based on the values and principles expressed in the agile manifesto (Beck et al., 2001). A major characteristic in such frameworks and practices is the focus on the people doing the work and how they work together, in that requirements and solutions evolve through collaboration between self-organising, cross-functional teams.

Note that agile development is only a subset of Incremental and Iterative Development (IID) which is the idea of revising phases over and over but not the same thing. This arrangement dramatically improves project efficiency. The iterative nature of the software lifecycle is increased further by tightening the normal traditional software development cycle (design-code-test loop) to at least once a day (if not much more frequently) as opposed to once per iteration.

Some of the early agile frameworks were Scrum in 1995, Crystal Clear, Extreme Programming both in 1996, Adaptive Software Development, Feature Driven Development, and Dynamic Systems Development Method (DSDM) also in 1995, others came later (Dingsøyr & Lassenius, 2016).

In the year 2001 the agile manifesto, a concise summary of agile values, was written and signed. With these came a wide adoption of the name “Agile processes” (Rao et al., 2011).

In 2001 a group of seventeen software developers took some existing management and development concepts and codified them into an umbrella philosophy which is the origin of what is today defined as Agile software development. This group produced a set of values that outlines this definition and sets apart agile development from the traditional processes (see figure 2.1).



Manifesto for Agile Software Development

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

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

That is, while there is value in the items on
the right, we value the items on the left more.

Kent Beck	James Grenning	Robert C. Martin
Mike Beedle	Jim Highsmith	Steve Mellor
Arie van Bennekum	Andrew Hunt	Ken Schwaber
Alistair Cockburn	Ron Jeffries	Jeff Sutherland
Ward Cunningham	Jon Kern	Dave Thomas
Martin Fowler	Brian Marick	

Figure 2.1: The Agile Manifesto (Beck et al., 2001)

In addition to the values, the agile association in February 2010 declared 12 agile principles. These were aimed at promoting common practice in every agile software development and better software development.

These principles, though informally defined, are intended to help developers produce software in an agile manner. By following these principles, developers will be integrating the four values of the manifesto. They promote agility within the software process.

Table 2.1: Agile Principles

	Principle
P1.	Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
P2.	Allow changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
P3.	Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
P4.	Business people and developers must work together daily throughout the project.
P5.	Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
P6.	The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
P7.	Working software is the primary measure of progress.
P8.	Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
P9.	Continuous attention to technical excellence and good design enhances agility.
P10.	Simplicity--the art of maximising the amount of work not done--is essential.
P11.	The best architectures, requirements, and designs emerge from self-organising teams.
P12.	At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.

Concerning the Agile manifesto values (figure 2.1), it is important not to misinterpret that the items on the right should be completely discarded, but just that more emphasis is given to the items on the left. For example, agile methods stress productivity and value over heavy-weight process overhead and artefacts (Hneif & Ow, 2009). This might sound catastrophic from a security engineer's point of view;

consider for example the statement “working software over comprehensive documentation”. First, this does not really mean that you completely do away with documentation but rather, we can redefine the scope and nature of the documentation to capture only the essential in order to give the developers more focus on the software (Taati & Modiri, 2015). Secondly to a security practitioner, security risk assessments, controls and mitigations, monitoring infrastructure, and IDS/IPS hardware are the products that he/she is supposed to produce, so to them the working software can be replaced by these.

Figure 2.2 is a diagrammatic overview of the agile Software Development Lifecycle at a high level.

To a software development team, agile methods have diverse benefits. Among these is an increased delivery rate and customer satisfaction due to customer involvement throughout the process, these will result in increased productivity among others (Rao et al., 2011; Rover et al., 2014).

On the flip side, some of the criticisms agile software development has received are; lack of structure and necessary documentation, only working with experienced developers, insufficient software design, and requiring too much cultural change. Uncertainties in the beginning of the process may result in difficult contract negotiations, and also, they provide limited support to the quality control in safety critical-software (Anderson, 2010; Cho, 2008).

A key point to note from the above list is that most of the above-mentioned are also the mismatch between agile methodologies and conventional methods for security engineering e.g., the use of heavy documentation in building the security case, prioritisation of security requirements versus functional requirements, lack of trained Agile security experts among other issues. In subsequent sections, a discussion of scrum in perspective to security is provided.

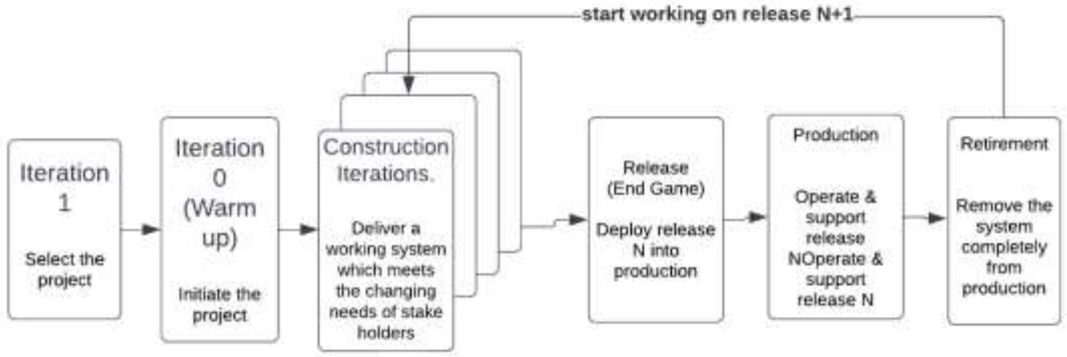

start working on release N+1
Iteration 1
Select the project
Iteration 0 (Warm up)
Initiate the project
Construction Iterations.
Deliver a working system which meets the changing needs of stake holders
Release (End Game)
Deploy release N into production
Production
Operate & support release NOperate & support release N
Retirement
Remove the system completely from production

Figure 2.2: High Level Agile Software Development Lifecycle.

The Agile Manifesto doesn’t provide concrete steps. Organisations usually seek more specific methods. Rao et al. (2011) suggested that methodologies are nothing but chosen, team specific conventions that are followed in order for a team to achieve agility. As long as they are generalised to be used by any team out there, they remain frameworks, of which the actual implementation may differ across teams. Several of these Agile Frameworks exist today. Some of these include; Extreme Programming, Agile Modelling, Scrum, Crystal methodologies family, Feature-Driven Development, Adaptive Software Development, etc.

In this research, the framework of choice is Scrum; the others are out of the scope of this work. Below, a brief overview of Scrum is given and also an explanation of “why scrum?” is offered.

2.3 Agility Assessment

Measuring agility can be done both qualitatively and quantitatively. Most studies have aimed at assessing agility at organisation level (Gandomani, 2014; Telemaco et al., 2020). In this work, e assessment models are discussed in the following section and a rational is drawn towards the selected model:

2.3.1 Agility and Discipline Assessment (Boehm and Turner):

Boehm and Turner (2004) proposed a framework to find a balance between agility and discipline. Discipline contributes to well-organised history and experience (Lucassen et al., 2015). Agility uses this history to adapt to new environments. The experienced teams are better placed to react and adapt to the changes, and even leverage on unexpected opportunities. This has been emphasised further by Jakobsen and Johnson (2008) where automatic tests, nightly build and integration are desired, such as in security engineering. Barry Boehm and Turner (2004), highlighted five critical factors involved in determining the relative suitability of agile or plan-driven methods in a particular project situation:

- The number of persons working in the project.
- The balance between chaos and order.
- An estimate of how much the team or organisation likes to work on the edge of chaos or with more planning and defined practices and procedures.
- The type of personnel required to achieve success in a project.
- The nature of damage from undetected defects.

In summary, there are projects which suit a more disciplined approach of development and security is one of the factors that call for this. Jakobsen and Johnson (2008) concur with this, but also went further to give guidelines on how you can achieve agility with discipline from CMMI. This has been discussed further in this chapter, since it has played a role in the framework design. Boehm and Turner’s (2004) does not cover the management level of the process.

2.3.2 Team agility assessment by Dean Leffingwell:

Leffingwell (2007), applies two metrics, agile project metrics and agile process metrics, to assess team agility. Analysing several categories of team’s performance aspects and ranking each category with a set of specific measures rated on a scale of 0 to 5 (5, describing better agility). The model also uses a radar chart, where each axis represents a metrics category.

Though the approach of data collection

2.3.3 Dimensional Analytical Tool by A. Qumer and B. Henderson:

This model evaluates agility and the adoptability of agile methods in software development organisations (Asif Qumer & Henderson-Sellers, 2006b). It provides a quantitative mechanism to measure the degree of agility of an agile methodology at specific levels within a process, using specific practices. The four dimensions are as follows:

- *Dimension 1* - Method Scope Characterization: It describes the scopes for the application of the agile methodologies. It is used to compare the methods at a high level. The scopes assessed in this dimension are: a) Project Size b) Team Size c) Development Style d) Code Style e) Technology Environment f) Physical Environment g) Business Culture h) Abstraction Mechanism
- *Dimension 2* - Agility Characterization: This dimension checks the existence of agility in agile methods for process level and method practices level. This dimension is the only one of the four proposed that is quantitative.
- *Dimension 3* - Agile Value Characterization: This dimension examines the support of six agile values in different practices of agile methods. Most of the values (4) are extracted from the agile manifesto; the rest are provided by the author based on the study of several agile methods.
- *Dimension 4* - Software Process Characterization: The fourth dimension examines the practices that support the four components of the software process proposed by (Asif Qumer & Henderson-Sellers, 2006b).

In this method agility is measured in terms of five different variables described in dimension 2: Flexibility (FY), Speed (SD), Leanness (LS), Learning (LG) and responsiveness (RS), where each variable may acquire a value of either 0 or 1 at any given time. A software development method may encompass agility in the design phase, planning phase or in the requirement engineering phase, but not necessarily all three. Thus, the degree of agility (DA) for each of these phases is the fraction of the five agility variables that is encompassed and supported. The below formula summarises this.

$$DA(Object) = \left(\frac{1}{m}\right)\sum_{i=0}^{m} \quad m\ DA(Object, Phase\ or\ Practices)$$

Qumer et al. (2006) used this approach to measure the degree of agility (at both phase and practice level) using Extreme Programming (XP) (Asif Qumer & Henderson-Sellers, 2006a)

The 4-D model is the most complete of the models discussed; it tries to include the major quantity of aspects related to the entire company, such as general software process characterization, agile values execution and agile practices inside the company, it provides a clear measure of agility that has been pegged on the agile principles. This work will only carry out assessment at the process level.

2.4 Scrum

It is an iterative and incremental agile software development framework for managing product development. Within it, people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value. Larman et al. (2012) gave the following description of scrum in 100 words;

> *Scrum is an agile process that allows us to focus on delivering the highest business value in the shortest time.*
>
> *It allows us to rapidly and repeatedly inspect actual working software (every two weeks to one month).*
>
> *The business sets the priorities. Our teams self-manage to determine the best way to deliver the highest priority features.*

Every two weeks to a month, anyone can see real working software and decide to release it as is or continue to enhance for iteration.

It is important to note that Scrum is not a process or a technique for building products; rather, it is a framework within which you can employ various processes and techniques. Scrum makes clear the relative efficacy of your product management and development practices so that you can improve.

The Scrum framework consists of Scrum Teams and their associated roles, events, artefacts, and rules. Each component within the framework serves a specific purpose and is essential to Scrum's success and usage.

The rules of Scrum bind together the events, roles, and artefacts, governing the relationships and interaction between them.

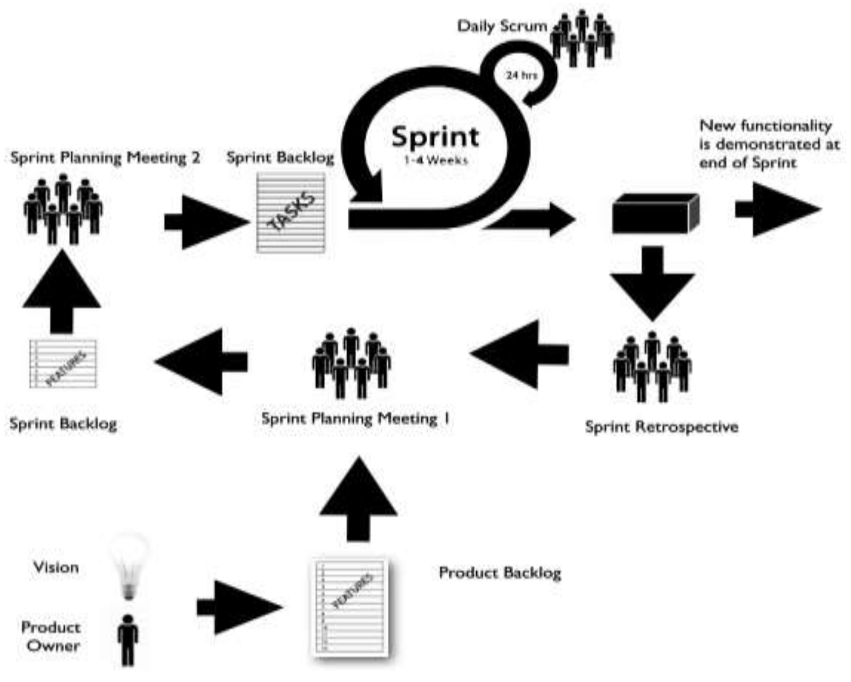


Figure 2.3: Scrum Framework

The major characteristics of scrum include:

i) Work is organised in short cycles.

ii) Management does not interrupt the team during a work cycle.

iii) The team reports to the customer, not the manager.

iv) The team estimates how much time work will take.

v) The team decides how much work it can do in an iteration.

vi) The team decides how to do the work in the iteration.

vii) The team measures its own performance.

viii) Work goals are defined before each cycle starts.

ix) Work goals are defined through user stories.

x) Impediments to getting the work done are systematically removed. (Ken Schwaber & Sutherland, 2017)

The scrum framework involves roles, ceremonies and artefacts where the roles are Product Owner, Scrum Master and Team. The ceremonies are Sprint Planning, Sprint Review, Sprint Retrospective, and Daily Scrum Meeting. Finally, the artefacts involved are Product Backlog, Sprint Backlog, and Burn down chart. In summary there are three roles, two shared tools and four meetings all within a regular period of time called a sprint. It boasts of transparency and enhanced communication as its strengths. Its creators advance it as a framework that promotes transparency and communication saturation, which together enable the team to regularly inspect and adapt to change (Schwaber & Sutherland, 2011).

Two key shared tools in traditional scrum are:

1) The product backlog (PB), which is basically a list of everything that should be done during the development life cycle divided into items that can be delivered independently. The PB is ordered according to item priority depending on the value the user places on the item. Scrum work is basically dictated by a product backlog.
2) The sprint backlog, which is a subset of the PB that is picked by the team to be completed in a sprint. It reflects the priority given in PB. Once picked, it cannot be changed.

The product owner is responsible for the commercial success of the product. He/she must know what the customer wants and the relative business value of those wants. He is responsible for controlling the order of the product backlog according to the business value of items (most valuable at the top) and thus he can be seen as the customer's advocate in the development. Since the team is responsible for the sprint backlog, the product owner spends about half his or her time working with the team to make sure the work in the sprint backlog reflects the vision of the product backlog. The other half of the product owner’s time should be spent consulting with customers and stakeholders to make sure the product backlog reflects their needs.

A sprint is a series of iterations that span a period of between 2 weeks to a month (30 days) that begins with a daily 15 minutes meeting called a scrum. The scrum meeting handles three questions; 1). What did you do yesterday? 2). What will you do today? 3). What obstacles are in your way?

Sprint reviews offer process improvement mechanisms, where the process is reviewed and improvements are suggested and agreed upon.

Security requirements are crucial in ensuring that security has been factored in the end product. In fact, research shows that 80-85% of project failures are due to incorrect requirements). A big failure of the waterfall was it was not flexible enough to handle dynamic requirements. Agile has won this war, but there emerges another challenge in that the voice of the customer is not one (Sven & Poller, 2017), for example from a single client we can have technical decision makers, end users and even system operators, all of whom have different priorities. Amidst this cacophony of voices, there is a danger of overlooking the security argument. This work makes sure this has been captured through a security sprint zero and within sprints by slightly changing the team roles.

A sprint zero is a pseudo sprint that does not necessarily deliver customer value or working software. Our sprint zero from a security perspective should be able to involve some research spike to carry out vulnerability, threat, and risk analysis. It should produce a security understanding for example in terms of abuse stories with a few critical stories developed to accomplish a minimal flexible upfront design in tandem with agile principles.

It is necessary to avoid a situation where you have a sprint zero that produces a heavy project plan and a large requirement document, taking us back to waterfall (Carlson & Soukop, 2017). To ensure this, the scrum alliance states the following as what it is not meant to do; 1). Assemble the team. 2) Organisation and logistics. 3) PB setup and design (Prakash, 2022).

More on the security sprint zero has been discussed in later chapters.

The popularity of SCRUM as a methodology for software development has greatly accelerated over the past few years. It has become the dominant agile methodology in industry today. This popularity is one factor that has advised the choice of framework. The idea is to have research that adds significant relevance to the industry.

Another factor is scrum boasts of good collaboration mechanisms which according to this research could be taken advantage of in terms of security engineering collaboration and also to build customer trust. In fact, scrum can be seen as just a framework of collaboration and project management, meaning it has very little emphasis on process. The next factor is flexibility of process i.e., as long as the scrum rules on strict timelines, roles and ceremonies are adhered to the process activities are left to be determined by the organisation or even the developers. Lastly, the researchers' preference and familiarity with the methodology has also played a role in selecting scrum as the agile methodology to use.

2.5 Security Issues and Scrum

It has already been established that agile methodologies suffer critical shortcomings when it comes to Security issues. In fact, it is reported that in the software development industry, some existing cultures have resisted the agile revolution especially where large or critical projects are involved (Bajta et al., 2015; Jamissen, 2012; Leffingwell, 2007). In these cases, the rigorous nature of the traditional methodologies has been trusted. In the case of security, it is quite easy to see why this is so, for example, looking at the agile principles previously mentioned “working software over documentation” at face value would spell disaster for a security expert. With all its strength, Scrum itself has several limitations related to security. These are stated below:

- As the Scrum release cycle is too short, there is not enough time for the development team to address security requirements for each release.
- Although the inclusion of security elements in the existing Scrum affects the process characteristics or agility of Scrum.
- Absence of documentation during the requirement planning. This happens due to the team players not having enough skills in building security. This work attempts to create security knowledge within the team as well as stakeholders.
- Another factor is the inadequacy of the process to instil the discipline required to undertake a security process (Abbas, 2016). The lack of guidelines in the collection of security requirements is also a factor, as there are no frozen requirements in agile as there are in the other conventional SDLCs (Karim et al., 2016).
- Lack of security awareness in scrums project managers coupled with the pressure to complete the project within a minimal amount of time affects the entire process (Rindell et al., 2015). For example, security activities such as threat analysis necessary during requirements capturing/analysis, or security testing within sprint phases are not included at all, or they cannot be completed due to a lack of time (Ghani et al., 2014).
- Security testing has its own limitations, for example indirect verification by demonstrating test or review results hinges on testing capabilities. Automated security tests, which fit into agile practices, do not work well on some types of vulnerabilities. Some tests are costly and hard to repeat within sprint, for example Penetration testing and thorough reviews. Alternatively, if carried out asynchronously of the iteration, their findings are likely to be handled through bug fixing (Erdogan & Per, 2010).

While viewing security requirements as non-functional requirements raises a challenge in that, these requirements are usually treated as soft goals and thereby there is no obvious way of defining whether they are met or not. Security is not just a quality aspect, but it also comprises functional and architectural aspects (Daneva & Wang, 2018). A wholesome approach is necessary

To overcome these issues, this study attempts to suggest ways in which the Scrum process can be suitably combined with security practices. The strengths of SCRUM can be taken advantage of to enhance security. Previous research (Azham, Zulkarnain, 2011; Ghani et al., 2014; Pohl & Hof, 2015), proposed solutions for this situation, however, most only address some portion of the process neglecting other parts, secondly there is no benchmark acceptable security process to compare against and lastly there was no evidence regarding how the proposed solutions affected Scrum agility. Later in this thesis, findings will be presented regarding these issues. Before this let us first look at traditional security engineering, the intention is to lay foundation knowledge and an understanding of the subject.

2.6 System Security Engineering

With the increasing reliance of the society on information, the protection of that information is becoming increasingly important. Many products, systems, and services are needed to maintain and protect information. The focus of security engineering has expanded from one primarily concerned with safeguarding classified government data to broader applications including financial transactions, contractual agreements, personal information, and the Internet (McGraw, 2004). Today’s systems are too complex, software-controlled, and highly networked. They are built by integrating hundreds of suppliers and commercial off-the-shelf (COTS) components, whose origin and level of integrity are difficult to ascertain (Karim et al., 2016).

Security engineering can be thought of as the discipline of building secure software. In 1989 the System Security Engineering was defined by the US DoD Military Standard1785 as follows;

Systems Security Engineering: An element of system engineering that applies scientific and engineering principles to identify security vulnerabilities and minimise or contain risks associated with these vulnerabilities. It uses mathematical, physical, and related scientific disciplines, and the principles and methods of engineering design and analysis to specify, predict, and evaluate the vulnerability of the system to security threats (Mailloux et al., 2013; Ross & Oren, 2016; Mailloux et al., 2016).

From the definition, we note that SSE must identify or predict threats and later use engineering methods to reduce or minimise risk.

2.6.1 Security and Vulnerability in Today's Environments

Security is defined as the probability (can be estimated or defined from empirical evidence) that an attack of a specified type will be repelled (Mailloux et al., 2016). A threat on the other hand is defined as the probability that an attack of a specified type will occur within a given time (Shostack, 2014).

The Internet continues to grow exponentially both in terms of technological advancement, the number of people using it and also in level of dependency. Individual, government, and business applications continue to multiply on the Internet, with radical benefits to end users. However, these network-based applications and services can pose security risks to individuals and to the information resources of companies and governments. Information is an asset that must be protected. The increase of sensitive information being handled by computer systems means a parallel increase in the need for their protection.

There are many aspects of information security ranging from network security, environmental security and one of the most important and vulnerable areas is software application security. Most of the applications have different platforms, frameworks, and various types of vulnerabilities.

Studies show that the greatest threat to computer systems in today's world comes from humans, either through malicious actions or ignorance. When the action is malicious, there is some motivation or goal that the attackers seek to achieve. An example is, disrupting normal business operations in order to deny service. Other objectives might be to steal, alter, damage or delete information or even make a joke or simply to show off.

For a malicious attack to happen successfully, apart from the attacker having a motive, he/she must also have a method and the system must be vulnerable to the particular method of attack (Mailloux et al., 2013).

Attack= Motive + Method + Vulnerability

The method exploits the system's vulnerability in order to infiltrate the system and cause an unwanted occurrence or launch the attack.

Software vulnerabilities are weaknesses in a software system design, implementation and configuration that could be accidental or intentional. They allow an attacker to reduce the system's information assurance. It is suggested that behind every malicious attack and computer security problem, there is insecure software (McGraw, 2006). Attackers do not create security holes, they simply exploit them (McGraw et al., 2011). Most security related vulnerabilities arise from defects that are unintentionally inserted in the software during the design and implementation phase of the development. Common software vulnerabilities include buffer overflow, heap overflow, race condition, format string bugs, poor random number generator, SQL injection, denial of service (DoS) and misplaced trust (Sánchez et al., 2020).

Emerging technologies such as cloud and IoT bring about new and diverse challenges. For example, in an IoT environment issues such as privacy, authorization, verification, access control, system configuration, information storage, and management, are the main challenges (Jing et al., 2014). Such technologies apart from collecting personal information data, they also monitor behaviour such as the user's eating patterns, schedules, etc. The security of such data has to be ensured to give users confidence in using the applications. Cloud computing also brings great deals of new challenges for data security, access control, etc (Paudel et al., 2013).

The discipline of systems security engineering provides an important mechanism for the engineering team to assess and mitigate the vulnerabilities of the system and subsystems.

Landoll (2011), observed that most cyber security breaches that occur don’t even require expertise, timing, motivation or even time. They are enacted by adolescents, disgruntled employees, or even novice computer users. A simple act such as opening an email, running a hacker program or placing a phone call could mean your entire system has been breached (Landoll, 2011). This means that security requirements should never be overlooked and should be tracked at every point during development, including in the training of employees. Actually, security development life cycle activities involve security professionals in all phases of development (Mailloux et al., 2013).

In general, security refers to an *absence of objective dangers,* i.e. of security ‘threats’, ‘challenges’, ‘vulnerabilities’ and ‘risks’, and of *subjective fears or concerns,* and to the *perception thereof*. From a realist perspective, *objective* security is achieved when the dangers posed by manifold threats, challenges, vulnerabilities and risks are avoided, prevented, managed, coped with, mitigated and adapted to by individuals, societal groups, the state or regional or global international organisations (Brauch, 2011). From a software engineering perspective it is impossible to achieve 100% security through engineering mainly because of the evolving nature of the software world itself, i.e. the threats of tomorrow would be expected to be more sophisticated than the security employed today. Also, many defects are not security-related, and some security vulnerabilities are not caused by software defects. For example, intentionally added malicious code is a security vulnerability not caused by common software defects. Another heavy factor is that sometimes security would come at the expense of other requirements, e.g. performance. These trade-offs will sometimes hinder the implementation of security features. These partially explain why security engineering will have to continue throughout the lifetime of the software and also include user training at different levels.

This research attempted to meet this objective in an agile development environment.

2.6.2 Security Engineering Process Overview

SSE process can be defined as the set of activities performed to develop, maintain and deliver a secure software product; it can either take a sequential or even an iterative approach (ISO, 2008).

A software product developed by the SSE process is supposed to be secure, meaning it is able to; Operate correctly in the existence of most attacks, either by resisting the exploitation of weaknesses in the software or tolerating the failures. Secondly, it should limit the damage from attack-triggered fault failures and recover quickly from those failures that the software was unable to resist or tolerate (Irvine & Nguyen, 2010).

The SSE-CMM model divides the process into three separate processes:

- risk process
- security engineering process
- assurance process

Risk can be defined as the expected value associated with an unwanted event (Landoll, 2016, p. 25). Risk assessment is the process of identifying problems that have not yet occurred. SSE-CMM states that risk is assessed by examining likelihood of the threat and vulnerability and by considering the potential impact of unwanted incidents (SSE-CMM Project, 1999). Risk analysis is essential for security engineering because it offers a platform for identifying and prioritising dangers that are inherent to the software. A risk process promotes sufficient knowledge of the environment and the software in terms of threats and vulnerability present. It puts the whole system into perspective within the deployment environment (Landoll, 2011, p. 52). It involves assessing potential threats to the system, those it will be vulnerable to, and the risk impact in case the threat occurs. Furthermore, it is considered vital in all phases of the development, but especially in the early phases (Bartsch, 2011) as a means of gathering security requirements. Security Engineering is about reduction of risks, and therefore you have to identify the risks before addressing them. It is a practice not found in agile methodologies, but it is vital in building secure systems.

Note the three components that stick out, vulnerability, threat and risk. A proper and accurate understanding of these security components will be essential to the effective designing of a framework to mitigate risk.

Assume a simple example scenario of an application on a cell phone; in a city like Nairobi, theft can be a threat, sometimes it can be assumed to be beyond our means to prevent some threats but at least measures can be taken to reduce the threat, e.g. in this case statistics on likelihood of occurrence in certain areas can be used to advise users on measures to reduce the threat. Not having a data recovery plan in case, the phone is stolen can be considered a vulnerability. The risk is loss of important business data and information, business disruption, etc.

Note that it is possible to have a threat existing that the system is not vulnerable to or to have a certain threat not existing, but the system is very vulnerable to it. In both these cases, the risk impact is 0. Risk impact is felt when threat meets vulnerability (Shostack, 2014, p. 41). SSE-CMM defines risk as;

$$\text{Risk} = \text{Vulnerability} * \text{Threat} * \text{consequence}$$

The majority of the research available today on secure agile development has given very little attention to the process of security risk analysis at the early stages of system development. Most work has been focused on the modelling of security requirements where models like abuser stories, UMLsec (Jurjens, 2004), etc. have been proposed. There is actually no agreement on the importance of risk management in AM, the assumption is that risk will be taken care of within iterations as they manifest into issues.

The problem is, some security risks may never manifest into issues until years after deployment. In this work, we agree with the idea that disregarding security risks analysis at the early phases might lead to costly problems down the road. It is important to have some acceptable level of risk information for the development team to make decisions concerning the security of the system as requirements change during development.

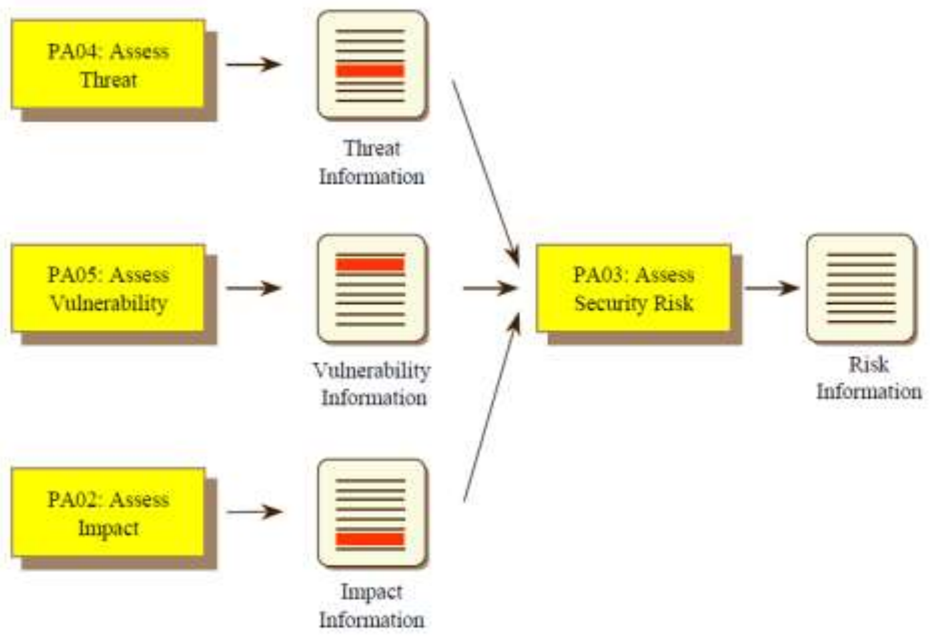


Figure 2.4: The SSE-CMM Risk Process Involves Threats, Vulnerabilities, and Impact (ISO, 2008)

A key principle in correctness by construction is to eliminate errors before testing (Kourie & Watson, 2012). It establishes that it is cheaper to deploy techniques that make it difficult to introduce errors in the first place before much progress is made on the development of software, arguing that testing is the second most expensive way of finding errors. The most expensive thing is to let your customers find them for you. This counters the argument that after deploying a prototype, security risks would be discovered by the customers or product owners and since AM can accommodate changes, then there will be a low cost for change. Though this might be true to some point, not all risks might be uncovered by users; in fact, sometimes users might not be having the expertise to inflict some of the threats. Also, the cost of this change brought about by the added security requirements might be significantly high.

Security system engineering works with other engineering disciplines to determine and implement solutions to the problems presented by the vulnerabilities and threats (Mailloux et al., 2016; Ross & Oren, 2016). This is where the implementation of security into the system occurs. Security engineering, like other engineering disciplines, is a process that proceeds through concept, design, implementation, test, deployment, operation, maintenance, and decommission. Figure 2.5 illustrates this process.

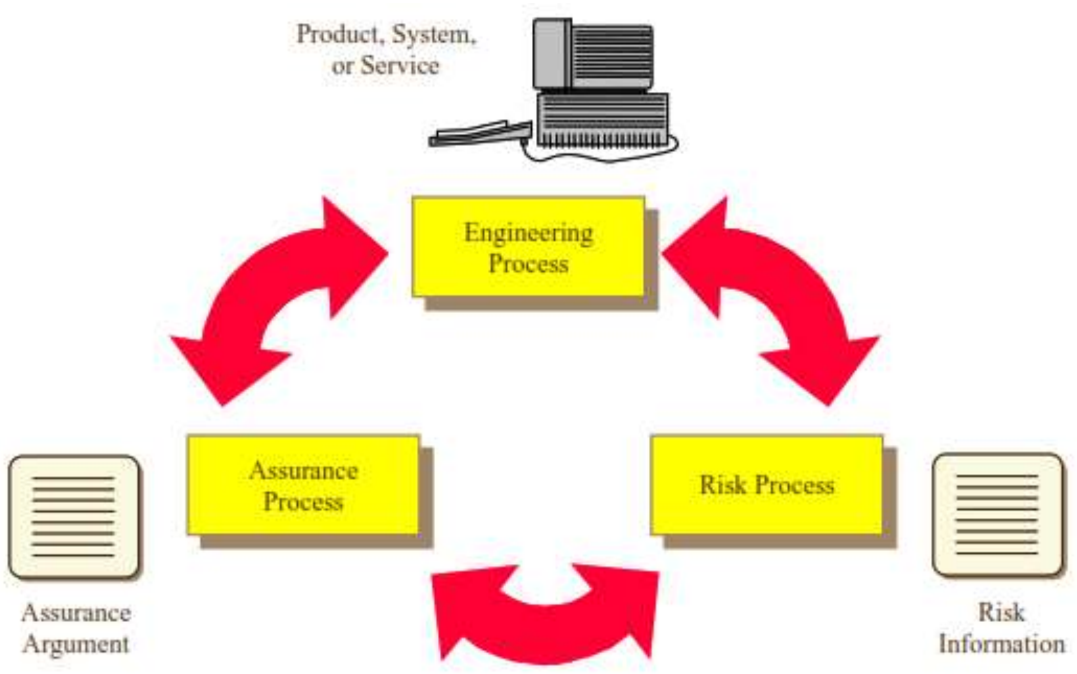


Figure 2.5: SSE-CMM's Security Engineering Process Overview (ISO, 2008)

From this approach, we can observe that Security should be explicit within the development process from the requirements level. Both overt functional security e.g., encryption and emergent characteristics should be covered during the security requirements. Building abuse cases has been suggested as a great way to cover emergent security issues (Bostrm et al., 2006; Mcdermott & Fox, 2002). This requires explicit coverage of what should be protected, from whom, and for how long.

Disregarding risk analysis at the design and architecture level will lead to costly problems down the road. At the code level (engineering), focus should be on implementation flaws.

Assurance is defined as the degree of confidence that security needs are satisfied (Mailloux et al., 2013; McGraw, 2006). It is considered a product of process. The mature software security processes models contribute to one aspect of assurance, the confidence in the repeatability of the results from the security engineering process. This confidence stems from the fact that a mature organisation is more likely to repeat results than an immature organisation.

2.7 Security engineering standards

SSE-CMM The System Security Engineering Capability Maturity Model was adapted from the well-known Capability Maturity Model (CMM) for software engineering. It is the basis of the ISO/IEC 21827 standard. It defines five capability levels for any organisation, and allows organisations to assess themselves, and put in place processes to improve their levels. This work uses this assessment criterion to evaluate the framework.

a) ISO/IEC 27001-27006

This is a set of related standards under the ISO/IEC 27000 family that provides an Information Security Management System. Unlike SSE-CMM the ISO/IEC 27000 family of standards specifies a set of requirements that organisations must satisfy (e.g., there should be a process to systematically evaluate information security risks). It is mainly based at management level and not a good fit for this thesis.

b) ITIL Security Management

ITIL (the Information Technology Infrastructure Library) is a well-known and comprehensive set of standards for IT service management. It was originally a set of recommendations developed by the U.K. government's Central Computer and Telecommunications Agency. The section on security management is based upon ISO/IEC 27002. It is broader and specifies processes like ISO/IEC 27002.

It does not really focus on the process but the specific application security standard, and thus it does not fit this work.

c) COBIT

ISACA, an international organisation devoted to the development of IT governance standards, has developed the Control Objectives for Information and Related Technology. This is a set of processes and best practices for linking business goals to IT goals, together with metrics and a maturity model. COBIT is broader in scope than ISO/IEC 27000, since it relates to IT governance. It is proprietary and did not fit the budget set for this work. Further, it is closely related to SSE-CMM and using SSE-CMM was deemed sufficient for this work.

d) NIST

The US National Institute of Standards and Technology has a number of white papers and other resources in its Security Management & Assurance working group. These are targeted at U.S. federal agencies; however, many of the recommendations will apply to other organisations as well.

The preference to SSE-CMM is guided by several reasons; first, it covers the whole security engineering area, secondly, it is a standard and suits the objective of this work, lastly, it defines capability levels which enable us to measure maturity of the process.

2.8 Capability Maturity Models (CMMs)

CMMs provide a reference model of mature practices for a specific engineering discipline. Organisations compare their engineering processes to this model to identify potential areas for improvement.

It’s critical to note that CMMs don’t generally provide operational guidance for performing the work i.e. the process definitions but rather they offer goal level definitions for and key attributes for specific processes including Software Engineering, Systems Engineering and Security Engineering. The*” what”* not the*” how”* (Lacerda & Wangenheim, 2018). It does not define the engineering process.

Currently, three CMMs are in widespread use, the CMMI framework, the FAA-iCMM, and the SSE-CMM. Of these, only the SSE-CMM was developed specifically to address security (Hefner, 1997; Mailloux et al., 2013).

In this research, as mentioned previously, we recognize the conflict in cultures between SSE-CMM and agile methodologies as SSE-CMM was built with classical software development methodology perspective. SSE-CMM is an accepted tool in security engineering, and therefore it is adopted as a school master for the discipline of security engineering best practices, goals and principles. The aim is not to completely adopt SSE-CMM, but to try to achieve what it was built to achieve in an agile manner. To achieve this, we first study SSE-CMM.

2.8.1 SSE-CMM

The Systems Security Engineering Capability Maturity Model (SSE-CMM) describes the essential characteristics of an organisation’s security engineering process that must exist to ensure good security engineering. It does not prescribe a particular process or sequence, but captures practices generally observed in industry. It is used to improve and assess the security engineering capability of an organisation.

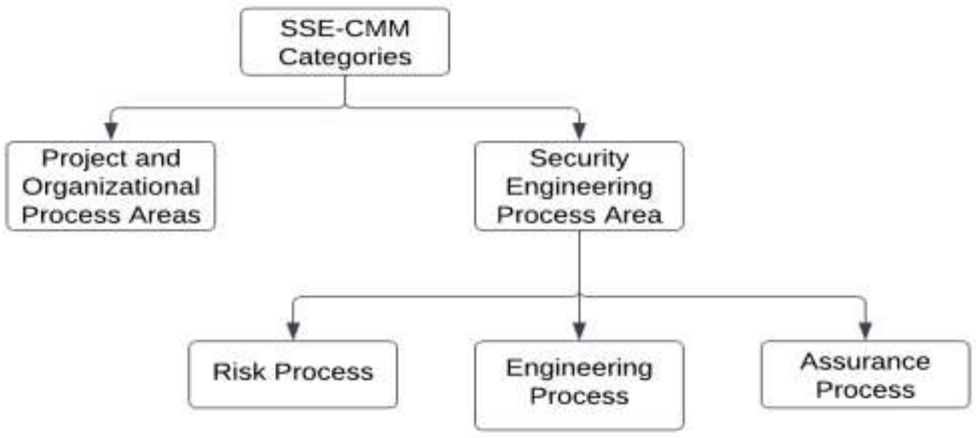

SSE-CMM Categories
Project and Organizational Process Areas
Security Engineering Process Area
Risk Process
Engineering Process
Assurance Process

Figure 2.6: SSE-CMM Organisation Structure

The model (SSE-CMM) is organised into two broad areas: Security Engineering, and Project and Organisational processes. Security Engineering in turn is organised into Engineering Processes, Assurance Processes, and Risk Processes. There are 22 Process Areas distributed among the three categories. Each Process Area is composed of a related set of process goals and activities. Because of the scope of this research, only Security Engineering will be the focus. Figure 2.6 displays the SSE-CMM Security Engineering area with its Process Areas and their place in the model. The International Systems Security Engineering Association (ISSEA) maintains the SSE-CMM.

It defines a comprehensive framework for evaluating security engineering practices against the generally accepted security engineering principles, thereby providing a way to measure and improve performance in the application of security engineering principles (Hefner, 1997). The field of security engineering has several generally accepted principles, but prior to the SSE-CMM it lacked a comprehensive framework for evaluating security engineering practices against these principles. The SSE-CMM also describes the essential characteristics of an organisation's security engineering processes. The SSE-CMM has also been adopted as the ISO/IEC 21827 standard.

Table 2.2: SSE-CMM Security Engineering Process, Goals and Area

SSE-CMM Process area	Goal	Process Areas
Risk	— Determine Metrics — Gather Threat, Vulnerability, and Impact Information — Identify and Assess Risks	PA04: Assess threat PA05:Assess vulnerability PA02:Assess impact PA03: Assess risk
Engineering	Identify, Implement and track security	PA10:Specify security needs PA09: Provide security input PA08: Monitor security posture PA01:Administer security control PA07:Coordinate security
Assurance	Build Security argument	PA11: Verify and validate security PA06: Build assurance argument

Because of the above arguments, wide acceptability of its activities and processes, and also easy certification if at all required in future, this research has borrowed heavily from the SSE-CMM security engineering process as the foundation model to learn and build arguments for an agile Security engineering model.

A common assumption to note with the use of CMMs, including SSE-CMM is that too much documentation is necessary. This assumption is countered in SSE-CMM, arguing that they point out only the type of information to be available (ISO, 2008). In an agile set-up, we can work with any evidence that can be tabled as an alternative to formal documents; this can be artefacts compiled or otherwise that do not interfere with agility. Another assumption is that the CMM defines the engineering process. This is not true according to SSE-CMM, as it only provides guidance for organisations to define their processes and improve over time.

2.8.2 CMMI

The Capability Maturity Model Integration (CMMI) framework helps organisations increase the maturity of their processes to improve long-term business performance.

It provides the latest best practices for product and service development, maintenance, and acquisition, including mechanisms to help organisations improve their processes and provides criteria for evaluating process capability and process maturity. Improvement areas covered by this model include systems engineering, software engineering, integrated product and process development, supplier sourcing, and acquisition.

2.8.3 CMMI and SCRUM

CMMI provides solid support for what disciplines need to consider. Jakobsen and Johnson (2008), argue that when CMMI is applied the disciplines create a focus on important aspects of agile methods that are usually not elaborated, for example how to ensure a proper quality of a product backlog or how to ensure a proper "production line" for the project.

For small agile projects, this guidance may not be necessary, but as the use of agile models continues to grow into various, larger, more complex and sometimes even critical projects, agile projects will need to address these issues related to increased size and complexity. When product security requirements are a priority due to the nature of the product or even the environment of deployment e.g., cloud, you need more discipline in the development. A more disciplined sprint zero, risk management, and various checklists have been suggested in (Jakobsen & Sutherland, 2009), and observed to bring slightly more discipline into your project with minimal effort.

This research believes and seeks to prove that the application of CMMI to a secure scrum model would instil the discipline that is sometimes missed by teams and which we deem as very necessary to the success in achieving secure agile projects. Jakobsen and Sutherland (2009), through the study of practitioners' experiences from combining CMMI and Scrum, identified examples of explicit guidance from CMMI that help to execute normal Scrum activities even better. These activities can be implemented in the spirit of the agile manifesto and principles and by doing so agile methods can be augmented and matured to ensure that even larger and more complex projects in the future can and will benefit from agile - with a twist of CMMI (Jakobsen & Johnson, 2008).

Looking at these from a security perspective within the scope of this work, the following benefits have been generated:

1) CMMI planning can be considered a kind of disciplined sprint zero, where it is ensured that an optimal high level framework for the project is established, a high level architecture for the project can be developed with risk and vulnerability features identified, including a high quality product backlog with security requirements, a production line definition, and well known security targets and vision for the project as a whole. This promotes a higher level of common understanding of security issues across the board.

2) CMMI quality planning specifies more accurately and efficiently the quality targets of the project and helps developers to a better interpretation of completion criteria and sprint goals. This is implemented at the beginning of the project (sprint zero), the question; "What needs to be protected?" is attempted and answered at this level.

3) CMMI will ensure that the project is tracked as a whole allowing the Scrum Team to concentrate on the current sprint, knowing that they periodically are informed of overall project status.

4) Scrum requires discipline regarding automatic tests, a nightly build, and integration. CMMI supports this need for discipline. The measure has proven to be cheap to establish, easy to understand, and therefore facilitating good habits.

5) CMMI expects the project to seek objective measures of performance of the project's processes. In Scrum, progress (of sprints) is primarily measured through the sprint burn down chart and the sprint review meeting. The project manager tracks the project based on selected measures within key areas like, product size, earned value, schedule, and quality. CMMI project planning provides good overall plans for the complete project, where each completed sprint is valuable input.

6) CMMI ensures that agile methods are institutionalised, including

- Consistent implementation throughout the organisation and continuous improvement, e.g., Systematic Scrum Guidelines, story inspection checklist.
- Role based training of all roles, e.g., Scrum Master and Product Owner.

In Jakobsen and Johnson (2008) the following CMMI activities are recommended to agile projects:

1. Establish your own sprint zero and include activities in item 2-6 below in it.
2. Use Risk Management to proactively address risks before they are identified as impediments.
3. Decompose requirements into features on the product backlog. Prepare the product backlog by decomposing the highest prioritised features to stories, allowing for efficient sprint planning. (This defines what you are really going to do.)
4. Use 3-point effort estimates on elements of the product backlog during initial planning.
5. Analyse dependencies, stakeholders, and risk on elements of product backlog.
6. Establish milestone and delivery plan and their initial relationship to product backlog.
7. Use Story Completion Checklist to maintain high quality of stories produced.
8. Decide and communicate quality objectives including, what code and documentation to formally review to elaborate definition of done.
9. Establish standards for project "production line" including development, build servers, and test servers.
10. Automate test and nightly build, and measure performance.
11. Establish criteria for committing code to integration.
12. Maintain integrity of configuration management, by using a checklist for Work Product Evaluation and execute it by the end of each sprint.

2.9 Security and Agile Methods

Despite the effectiveness of agile processes in delivering systems that the customer wants, the rigour of assurance and safety are found to conflict with its lightweight and informal nature. One of its biggest obstacles is lack of adequate tracking of security requirements (Oueslati et al., 2015). Security requirements are not given the same citizenship as functionality requirements and reliability in the development process (Sven & Poller, 2017).

The IT industry is viewed as a product and service selling business. Developers tend to focus on extending features to enhance functionality and interoperability with other popular products to capture the market. This “customer-pleasing” drive has spurred the adoption of agile software processes in the industry, but at the same time it is done at the expense of security requirements. Customers end up buying information protection products and services e.g., firewalls, guards, crypto mechanisms, intrusion detection systems, security administration packages, etc. as a substitute to secure systems.

While security engineering is a rapidly growing research area, the current approach of Information system security is based on a heavyweight process which starts with an extensive risk analysis followed by the development of a security policy, then finally the implementation of that policy.

Within traditional agile, security requirements elicitation activities are not given residence. These include risk assessments, vulnerability assessments, etc. Another issue is that the fixed iteration time may not fit time-consuming security activities (Oueslati et al., 2015). There are also incremental development challenges such as the agile refactoring practices. Security requirements are usually seen as constraints on the functional requirements. The agile aspect of refactoring usually breaks security constraints (Schön, Winter, & Escalona, 2017). Changes of requirements and design breaks system security requirements posture (Karim et al., 2016). Continuous code changes make completing the assurance activities difficult. Another challenge identified is the tracing of the requirements to security objectives; this becomes difficult since in agile development, requirements change as the process progresses. The agile process should have mechanisms in place to handle security requirements changes.

As we have seen in the security engineering process model (traditional), security issues require detailed planning in an initial planning phase, typically resulting in a detailed security analysis (threat and risk analysis report generation), a security architecture, and instructions for security implementation (e.g., specification of key sizes and cryptographic algorithms to use). On the other hand, agile software development methods like Scrum reduce or even do away with these initial planning phases and focus more on producing running code. Scrum also allows fast adaption of the emerging software to changes of customer wishes. This (in relation to security) translates to a lack of detailed security architecture or security implementation instructions from the start of the project. It also means that a lot of design decisions will be made during the runtime of the project.

In summary, Agile models focus less on documentation and plans than traditional development models. Existing software security development started when traditional development models (waterfall, V-model) were considered best for large, complex systems. Though the security engineering models do not explicitly require traditional development models, their compatibility with agile models are not ideal. For instance, in traditional approaches, development of secure systems requires documentation as a means of providing assurance to customers and this is not a priority in agile. The task of developing detailed documentations slows down the process, and thus it is just not agile. Table 2.3 gives a summary of these security activities (drawn from SSE-CMM) compared to traditional agile (scrum) and highlights some research works on the process areas.

Table 2.3: Security Comparison between SSE-CMM and Scrum

SECURITY ENGINEERING PROCESS	**PROCESS AREA**	**PERFORMED IN STANDARD SCRUM**	**COMMENT/ MENTION IN LITERATURE**
<u>**RISK PROCESS**</u> **Goal:** –Determine Metrics –Gather Threat, Vulnerability, and Impact Information –Identify and Assess Risks	PA04: Assess threat PA05: Assess vulnerability PA02: Assess impact PA03: Assess security risk	No No No No	Most relevant literature cover the modelling of risk and do not give a comprehensive approach to risk (Ransome & Schoenfield, 2021a). Others like Siponen et al. (2005), Peeters (2005) suggested a simple risk analysis process (Pohl & Hof, 2015). Code analysis (Finch, 2009) for PA05
ENGINEERING PROCESS **Goal:** Identify, Implement and Track Security	PA10: Specify security needs PA09: Provide security input PA08: Monitor security posture PA01: Administer security control PA07: Coordinate security	Insufficient Insufficient No No	Developers transform security needs to engineering tasks. Use of automated tests has been suggested (Evans, 2008) and story-based tests (Sutherland et al., 2008). For PA07, PA08 the use of a "security manager" was suggested in (Azham, Zulkarnain, 2011)
ASSURANCE PROCESS Goal: Build security argument	PA11: Verify and validate security	Insufficient	Through tests and definition of done (Bostrm et al., 2006; Woody, 2013).
	PA06: Build assurance argument	Insufficient	Code documentation, test driven development (Sutherland et al., 2008).

2.10 Approaches to Secure Agile Software Development Process

There have been previous research attempts done on ensuring quality software products during agile development in the areas of reliability and security. Some are environment specific, while some are not. The last column of table 2.3 shows how some researchers have tried to solve the issue. In this section, the previous works will be discussed further.

The study (Baskerville, 2004; Woody, 2013), has shown the need for researchers to develop organisational approaches and methodologies that respond to the dramatically changing context of information security. The conflation of information warfare and short cycle development theories promise new security practices that anticipate threats and rapidly deploy necessary safeguards in the context of changing system environments (landscapes) and ever evolving systems threats (Baskerville, 2004). Baskerville (2004) further analysed important ways in which the context of Information security has become a more dynamic setting, leading to emergent vulnerabilities and highly novel threats. He then described the need for different forms of security organisations to respond to this new context. His work framed some research questions which must be addressed for Information security to respond to these needs. This study agrees with Baskerville in that information Security is a dynamic phenomenon in today's world and that there is a need for a review of the practice of information security. This work is not an attempt to extend or modify short cycle development, but to modify security engineering to fit into or to the agile short cycle without affecting agility. The goal is producing a theoretical security framework which will be tested in a development cycle. This work attempts to cover the whole security engineering development life cycle as laid out in SSE-CMM. It spans out from the risk analysis to engineering and finally to the assurance area. In this section, contributing work to the various phases are discussed as per the security engineering phases provided by SSE-CMM.

Risk assessment is the overall process of risk identification, risk analysis, and risk evaluation (ISO 73). It has been identified as essential for effective security engineering (Sommerville, 2018). Somerville (2018) also cements the importance of an initial preliminary risk assessment like the one proposed by SSE-CMM as a tool to gather security requirements. The difference is, while Somerville admits that at this level the details are incomplete, SSE-CMM follows the traditional approach assuming that all the detailed designs have been availed, and all conclusions can be made at this level. Apart from these, for security to be included throughout the process, it is important to have all members of the project understand the importance and have an awareness of security (Bartsch, 2011; Sommerville, 2018). This understanding will promote security throughout and reduce the conflict of stature between security and functional requirements. Bacca (2017) has also emphasised the need for customer involvement as a prerequisite for adequate security measures. Customers or a representative of them should be involved early in the project in addressing risks and non-functional security requirements. With respect to this, this research has included a preliminary risk assessment that will be carried out before the actual production begins. Within what has been dubbed as the security sprint zero. In this, all the stakeholders are involved as some roles like deciding controls are not the software engineers' roles but rather management.

Another aspect is the inclusion of security requirements in the product backlog. Azam et al. (2017) proposed a security backlog which would create a separate list from the traditional PB and a security manager to champion security within the process. This approach creates two sources of requirements and an introduction of an additional role. In scrum the PB is the sole authoritative source of requirements for the teams (K. Schwaber & Sutherland, 2011), hence their efforts redefine scrum. Further, if the PO does not understand security, there is no guarantee that security requirements will have equal consideration to functional requirements in the PB prioritisation. This is because the PO has the sole mandate to prioritise the PB. Other works suggest additional roles like the Security master, software security architects, security champions, etc. (Bezerra, Sampaio & Marinho, 2020; Ransome & Schoenfield, 2021). As already mentioned, this would lead to redefining scrum as scrum is defined by specific roles. These suggested roles carry out essential functions as far as security engineering is concerned, and they can be assigned as part of the cross-functional team without interfering with the basic model of scrum. This work integrates security requirements into the normal PB, with associations to related user stories. It emphasised the need for customer involvement as a prerequisite for adequate security measures.

Rindell et al. (2021) performed a survey and observed that the activities taking place early in the life cycle were also considered most impactful. A major contribution of this work is this initial risk area. From it, we define security goals, security abuse stories and initial security requirements. Clarity of these would guide the process and prioritise the security throughout the process. It would also be important in formulating the definition of done for security requirements.

In their work, Rindell et al. (2021), also observed a discrepancy between the level of use and the perceived security impact of many security activities was observed. They called for research and methodological development for better integration of security engineering activities into software development processes, methods, and tools. Integrating security futures in agile methods would serve to capacitate agile methodologies to handle security. The question is how. One research work (Siponen et al., 2005) has tried to answer this. It agrees that most existing security methods hinder development but argues that you can integrate security into agile methods seamlessly provided the security measures meet certain requirements. They suggested the following requirements.

1. Security approach must be adaptive to the agile software development methods.
2. They must be simple; they should not hinder the development project.
3. The security approach, to be integrated successfully with agile development methods, should offer concrete guidance and tools at all phases of development, i.e., from requirements capture to testing.
4. A successful security component should be able to adapt rapidly to ever-changing requirements owing to a fast-paced business environment, including support for handling several incremental iterations.

They suggested a generic security process consisting of key security elements (from information security “meta-notation,” or notation for notations) in distinct phases of software development (requirements analysis, design, implementation, and testing). The steps are not necessarily sequential, and, in any case, every step is optional. The practicality of their approach has not been tested in their paper.

Integrating some heavyweight activities with agile processes may reduce agility. An algorithm termed ART (Agility Reduction Tolerance) was suggested (Keramati & Mirian-Hosseinabadi, 2008). In this method, all the security activities are extracted from the software process and guidelines, and then the agility degree of activities is defined to measure their nimbleness. Integration issues of agile and security activities are handled and an algorithm to integrate security activities with organisation's agile process is introduced; finally, agility reduction tolerance (ART) parameter and its optimum value are discussed. Other studies have taken a similar approach, e.g., Real Agility Degree (RAD) (Sonia, 2011) to determine the compatibility of security activities with agile. In this work, the driving theory is Security engineering. This work looks at the goals of each phase of security engineering, and then through literature arguments it determined the best way to achieve them in an agile manner.

Another issue of concern in agile security is the generation of the assurance argument. Secure scrum (Pohl & Hof, 2015) recommended implicit documentation, such as the use of test cases, as well as other artefacts such as authorization policies. Another suggestion was towards moving the security documentation to code and, in effect, doing away with heavy security documentation (Bartsch, 2011). This approach has been backed by Baca and Carlsson (2011), they recommended doing away with "outdated security documentation and replacing them with implicit documentation". It suggested the use of test cases as well as other artefacts such as authorization policies as a way of moving away from the out-dated heavy documentation. Another suggestion in the paper was towards moving the security documentation to code.

This work is not a first attempt in trying to tackle this problem. In (Pohl & Hof, 2015), an improved scrum framework labelled 'Secure-Scrum' is introduced. It bears four components put on top of the standard Scrum framework. Secure Scrum influences six stages of the standard Scrum process. It was implemented in a small software development and found to be practical, though further investigations were recommended as it was only tested by a group of 16 students once. The components of Secure-Scrum also have not been evaluated for effectiveness on the software security.

Another attempt was the Viewnext-UEx model (Núñez, 2020). It comprises four development areas, which include some organisational level activities. These activities are out of the scope of this work. The present study focuses on the SDL methodology. Also, some of the recommendations like the testing done by a different team from the development team composed of security experts would compromise the agility of the process in this research's opinion. In some agile frameworks like scrum, they would disrupt the flow of the process. The paper does not detail the logic behind the framework organisation nor give much detail about these areas.

Sonia (2011) introduced an agile security framework bearing five phases of an agile process. They present a technique for eliciting security requirements overlapping abuser stories with attack trees, resulting in an approach adopting the best characteristics in an attempt of avoiding making the approach anti-agile. Though the attempt was modest, the practicality of the approach has not been tested.

This research has taken a retrospective approach rather than producing a whole new concept. An agile security-engineering framework modelled after the ISSEA’s System Security Engineering Capability Maturity Model (SSE-CMM) is designed and introduced. SSE-CMM does not specify processes but offer guidance and standards for organisations processes, and thus we use the guidance to try and formulate a model that aims towards a standardised security engineering approach. Case studies (Cho, 2008; Ghani et al., 2014; Núñez, 2020) have shown that it is possible to integrate the so-called heavy-weight security practices, in contrast to other sceptical literature. The idea is to tweak their implementation to maintain agility of the process. The design of the proposed framework is discussed in the next chapter.

CHAPTER THREE

OVERVIEW OF THE SECURE AGILE PROCESS FRAMEWORK

3.1 Overview The SSE-CMM emphasises that security engineers are part of a larger team and need to coordinate their activities with engineers from other disciplines (SSE-CMM Project, 1999). This helps to ensure that security is an integral part of the larger process, and not a separate and distinct activity.

In this work, we develop a framework to achieve security engineering in an agile development set up. Theoretically, Scrum is built for coordination of teams; the hypothesis is that this quality of Scrum can be used as an advantage to build security. To achieve this, the activities should be lean and should not collide with the agile method process. Also, the security engineer is part of the cross-functional scrum team. It should also not slow down the sprint to the extent that a sprint will not deliver the artefact at the set time.

It is important to acknowledge that sticking with the traditional security process and activities as they are doomed to fail due to a conflict in cultures. Preliminary research has produced the following recommendations for the framework development:

- SSE-CMM gives the general understanding of a security engineering framework. The aim is not to exactly match the activities, but to meet the security goals and targets.
- Discipline is necessary to achieve security. Discipline has proven to be an essential ingredient in building secure software (Barry Boehm & Turner, 2004; Jakobsen & Johnson, 2008). An example is the correctness by construction (CbyC) methodology (Kourie & Watson, 2012) mentioned in a previous segment which insists on application of rigour with remarkable results (Chapman, 2006). This work will demand some level of discipline to achieve security. This will be supported and implemented by the CMMI activities recommended in (Jakobsen & Sutherland, 2009).
- From the above, the framework should bring to focus security requirements, making them and their level of criticality visible to the entire team and to the

users. It should also provide a way to track the requirement through implementation, to testing up to the definition of done.

- Security engineer is a role, not an individual. Since agile teams are small (five to nine people in scrum) the addition of another security engineer to the team who will need to be involved in the daily meetings is redefining the scrum model.

3.2 Features of the Proposed Framework:

The agile security framework is divided into three parts, following the model of SSE-CMM:

- Agile risk process
- Agile security engineering
- Agile assurance.

In the spirit of agile software development, these three do not necessarily have to follow each other, in practice they will be iterative, overlapping, feeding, and improving each other as depicted in figure 3.1. The figure shows an iterative high-level framework that futures an initial risk analysis. This initial risk analysis is important in that it promotes a common understanding of the system and consensus on security objectives. It is almost impossible to design and build a secure application without security awareness (Taati & Modiri, 2015). This phase ensures there is security awareness across the board, i.e., within the development team as well as among the stakeholders.

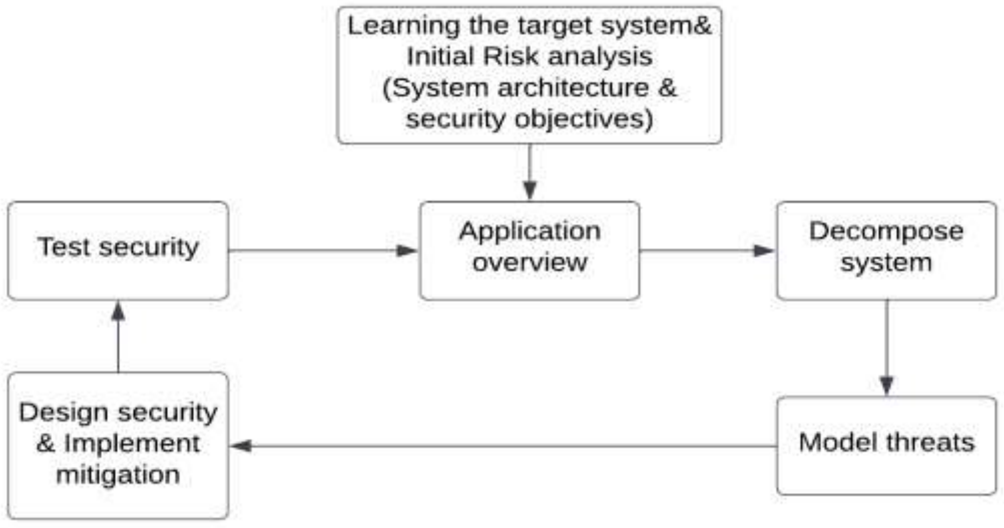

Learning the target system& Initial Risk analysis (System architecture & security objectives)
Test security
Application overview
Decompose system
Design security & Implement mitigation
Model threats

Figure 3.11: High Level Security Engineering Framework Design

Notice from the diagram (Figure 2.7) that risk analysis continues within the development iterations, this is because the design is more detailed and hence security being an emergent feature can now be described in detail.

Security engineering appears within the development iterations in the form of design and implementation of mitigations. Lastly, the assurance comes in the form of tests and the development of lightweight documentation.

The following section describes this framework in terms of the three areas; risk, engineering, and assurance. Later, how to implement it in scrum is explained.

3.2.1 Agile Risk process

Security Risk purpose is the identification of the combinations of threat, vulnerability and impact that deserve further attention (Mailloux et al., 2013). The specific goals being the determination of metrics, gathering threat, vulnerability and impact information as well as identifying and assessing risks (ISO, 2008). In SSE-CMM the risk process end product is the risk information which embodies the goal of the risk process i.e. basically to find the combinations of threat, vulnerability and impact that are deemed sufficiently risky to justify action. This information will form the basis of the security needs that will develop the security requirements.

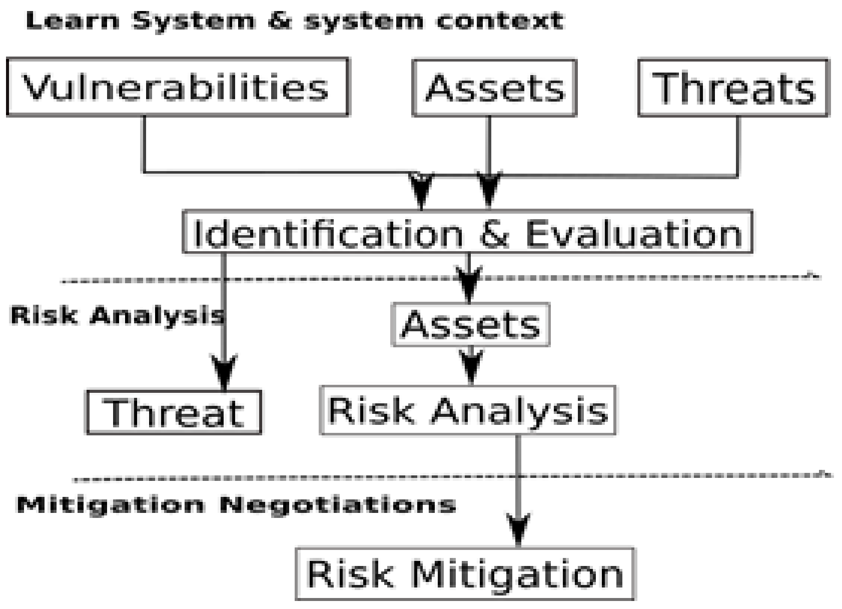


Figure 3.2: Security Risk Analysis Model (In et al., 2005)

According to SSE-CMM the information from the risk process and other information about system requirements, will be used to produce relevant laws, and policies. Also, security engineers working with the customer will be able to identify security needs. Once needs are identified, security engineers identify and track specific requirements.

The SSE-CMM's risk process is modelled as depicted in the figure 2.1. It is divided into four areas; 1) assess threats, 2) assess vulnerabilities 3) assess impact 4) assess security risk. Note that all these are interdependent and do not have to follow the sequence given. The threat report, vulnerabilities report and impact all are used to generate the risk report.

The information on the potential impact of risks should be considered important while choosing from the product backlog what should go into the sprint (sprint planning). It can be used to give weight to the security requirement and in that way make the relevance of the particular security item explicit to the user.

This work advocates for an initial security risk process which will highlight threats and assets. In a bid to reduce cost of change, it would aim at providing a minimum-security threshold and awareness from the onset of the project that would go a long way in building a secure system.

Requirements engineering builds a bridge to design and construction (Ransome & Schoenfield, 2021). This bridge can originate from the project's holders (e.g., managers, customers, end users) or from a broader system definition, where the software is viewed as a component of a larger system domain. In the specific case of security requirements, one cannot rely on project stakeholders alone; this is because software security issues are best understood by experts (Oueslati et al., 2015). The stakeholders can know the data they want protected, but they might not know how it needs to be protected within that system.

From the book "Risk Centric threat modelling" (Ucedavélez & morana, 2015), risk assessment formats and frameworks, including the one proposed here, seek to answer the following basic questions:

- What needs to be protected?
- Who/What are the threats and vulnerabilities?
- What are the implications if they were damaged or lost?
- What is the value to the organisation?
- What can be done to minimise exposure to the loss or damage?

In this work, while trying to maintain agility, the initial risk analysis should not be exhaustive or even conclusive as it is in the case of traditional security engineering methodologies where it is assumed requirements are static. This risk assessment model will be used to produce the basic threats, vulnerabilities, and impact, and seeks to give a skeleton risk report in the form of abuser stories, which are associated with some risk impact metric. This will provide a starting point which will be refined further within the iterations or even changed.

Traditional security engineering standards like CbyC (Chapman, 2006) and SSE-CMM naturally inherited the waterfall approach of requirement gathering, which was based on a sequential, non-iterative lifecycle and assumed stable development environments where project plans and security requirements are defined, fixed, and documented upfront. This approach is quite the opposite of agile methods, in that they bore extensive documentation and were criticised for being both time and resource consuming processes. Agile methods do not really need an initial deep knowledge of the requirements. According to (Sillitti & Succi, 2005), it is accepted that:

- A precise understanding of all customer needs is not practical, requirements are not stable, the customers may even change their requirements during the development process, and therefore a priori specification of requirements in a complete way is not possible.
- Apart from that, there are also some software features that are irreversible, or hard to change after they are implemented without seriously impacting the scheduling and the budget of the project.
- Often, the customers are not even able to specify all required main functionalities of the applications.

Within agile Scrum, the Product Owner is responsible for the Product Backlog, including its content, availability, and ordering. The early PB is just a list of initial and best understood requirements. The scrum development team will participate in the refinement of the product backlog throughout the lifetime of the product. All scrums process (the kick-off, the sprint planning meeting, the sprint, the daily scrum, and the sprint review meetings) provide an avenue for iterative processing.

In SCRUM, due to the fact that developers get to choose from the PB what they should work at and the high stakes given to customer satisfaction pushes security requirements further lower the pecking order but if security requirement is attached to a particular user story that user story will not be defined as done if the security requirement is not implemented. To achieve this, it is important to first identify the risks which inform the security requirements and ensure they are implemented and tested.

Requirement elicitation and management in agile is a topic of interest in itself (Sillitti & Succi, 2005), but in the case of security it is important to note that the requirements are different in that: 1) The customer may not be the best source of what is needed, the customer may know the business value of his/her assets but has no idea of the threats and vulnerabilities present in order for the customer to be able to understand and be able to make the right decisions, he/she must have a picture of the risk scenario. 2) We have seen that some level of discipline is needed to have security requirements tracked and implemented. To track the implementation of security, you must have a clear understanding of what you are protecting to start with. Once you have a clear understanding of the threats and vulnerabilities within the system and the system environment all together, it makes it easier to see new risks as user requirements evolve during development.

This work adopts the suggestions of (Sutherland et al., 2008) discussed previously right from the beginning in the following manner; the initial risk evaluation be carried out in the Sprint zero. The sprint zero should be used to produce the basic skeleton and plumbing for the project and not exhaustive requirements to leave room for the agile flexible requirements to be refined within the iterations. Jacobsen recommended the use of user stories (abuser stories/abuse case in the case of security) but we find this to be unsatisfactory, that is, mare stories would be insufficient in advising the product owner in terms of priority levels. It would be of great contribution to extend these stories to include threat levels and risks.

In the formulation of the agile process, we consider focusing on promoting an understanding of risk knowledge rather than the documentation generated. Within sprints, we can assume the risk of knowledge as the “customer value” added and the developing team as the customer since they are the main consumers of the information.

SSE-CMM views risk as a product of vulnerabilities and potential threats. We adopt this view and recommend a semi structured approach to achieving the risk information. We borrow from the model in figure 3.2 (In et al., 2005) to formulate the risk assessment process, as it follows the same approach as SSE-CMM. Figure 3.3 depicts our proposed semi structured risk analysis approach.

The risk method proposed in this work takes a system-centric approach, as opposed to either asset or attacker centric. Asset-centric approach is the common-sense approach, but it is not actually the best. According to (Mailloux et al., 2013) system centric approach is the best because it dissolves differing understandings and unrealistic assumptions about the system among developers and the stakeholders at large. In this model, assets assessment helps to determine criticality and costs, but not to drive the process. This research finds this particularly attractive because the design details are not yet defined at this stage. Risk can be investigated from a high-level system architecture and grow the understanding as the project progresses. The intention is to continue discovering threats and risks with sprint iterations as the design continues to take shape.

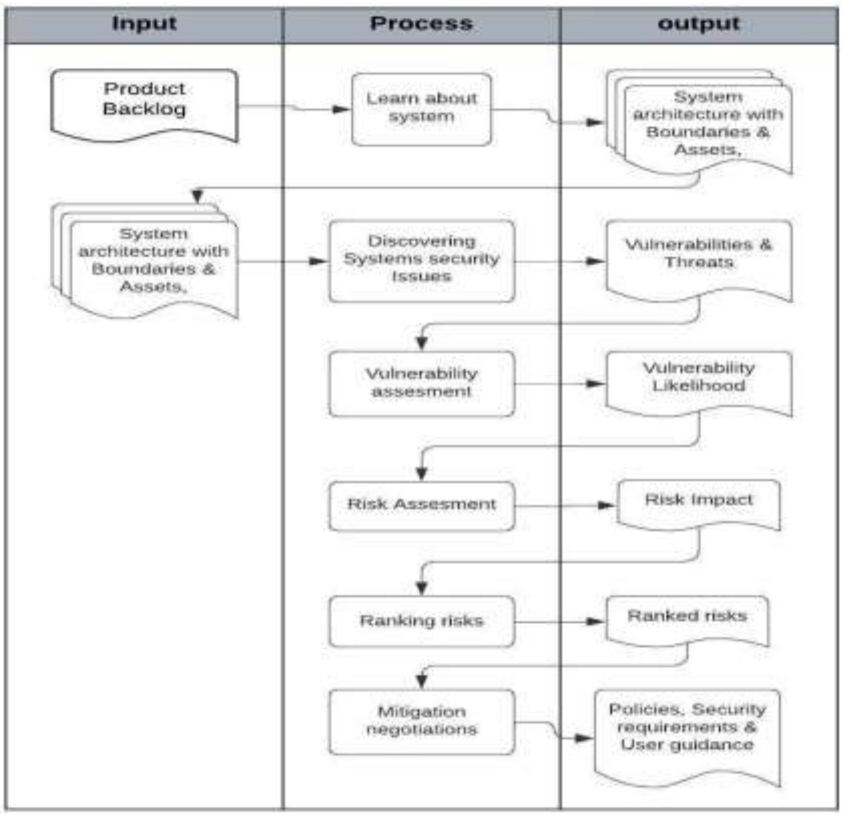

Input
Process
output
Product Backlog
Learn about system
System architecture with Boundaries & Assets,
System architecture with Boundaries & Assets,
Discovering Systems security Issues
Vulnerabilities & Threats
Vulnerability assesment
Vulnerability Likelihood
Risk Assesment
Risk Impact
Ranking risks
Ranked risks
Mitigation negotiations
Policies, Security requirements & User guidance

Figure 3.3: Proposed Risk Model.

The general process of this architectural risk analysis is to analyse the application's architecture; identify potentially vulnerabilities that may allow a user, mistakenly, or an attacker with malicious intent, to compromise the application's security; and evaluate the risk of security breach; finally suggest the implementation of countermeasures (Karim et al., 2016).

All scrum roles should participate and if deemed necessary a representative from the client or end user should be available. This is employed to build consensus about security requirements and eventually improve their prioritisation stake in the product backlog.

The risk process has the following steps:

1. Learning about the target System: it takes in the user stories from the PB as input. This involves learning about the analysis target and promoting a common understanding across the board. It will include going through and understanding specifications, understanding the business environment including assets, the access rights, discussing and brainstorming with the group; producing a high-level architecture diagram; determining system boundary and data sensitivity/criticality; discussion of current system if any exists.
2. Uncovering security issues surrounding the software: taking the system information as input, it involves arguing about how the product works and determining areas of disagreement; establishing trust boundaries; identifying system assets, identifying threats, and agreeing on relevant sources of threat. Identifying vulnerabilities (aided by using tools or lists of common vulnerabilities where possible); mapping out exploits and discussing criticalities; exploiting current and planned security controls (Shostack, 2014). The actual method for this part is left to the team to decide and can range from informed brainstorming to the use of other techniques.
3. Determine the probability of compromise: Involves assessing attack scenarios for vulnerability exploitation and balancing controls against threat capacity to determine likelihood (Viega & McGraw, 2011).

4. Perform Risk impact analysis: Sub-steps include determining the impact on asset and business goals and considering the impact on security. Note that:

   Risk=Impact X Threat Probability (ISO, 2008)

5. Rank risks
6. Develop mitigation strategies: Involves recommending countermeasures to mitigate risks.

For example, consider a team with a task to develop online payment functionality in an e-shop. The asset will of course be the money. The security goal can be to protect users' money. This can then be broken down to protect users' Personal Identifiable Information (PII) data.

The threats can be hackers, rogue employees, etc.

An abuser story for an attack method can be e.g. *A crook obtains and later misuses operator passwords for the e-shop by tapping messages sent through a compromised network host during operator log on.*

The probability of such an attack in such systems is obtained, and the risk impact can be calculated as a product of this probability and asset loss value metric.

The team will decide whether the risk is acceptable or not.

The methods to use are beyond the scope of this work. The only concern is that the security engineering objectives or goals defined are met in an agile manner. Any of the methods proposed in other works would suffice as long as they adhere to the values and principles of agility.

As mentioned, the security requirements produced should not be exhaustive nor conclusive, but rather dynamic and hence subject to change with consecutive iterations.

Including security requirements into the product backlog bears its own challenges. Security comprises security attributes, functions and architectural futures. The security attributes desired are actually quality aspects and therefore addressed during the sprint retrospective. The architectural and functional attributes on the other hand provide greater challenges in that; 1) There are some that are obvious and mostly stakeholders request for them. This will fit in the PB like any other feature. 2) There are others that can only be defined during system or feature design. This happens within the sprint. 3) Most security features are connected to certain levels of functional requirements. In this case, we adapt the s-tag (Pohl & Hof, 2015) approach of security tag linking user stories to security issues.

The concept of s-tags was introduced by Pol and Hof (2015). S-tags link the security requirement to a product backlog item. We find this appropriate since you now have one pool to pick from. This work extends it further by putting weights to security requirements. These weights will be determined by the business cost of breach to the user or the system. The user will be involved in assigning these weights to promote awareness of security requirements to the user. After calculating the 3-point effort estimate of each task, it is now easier to pick tasks from the backlog which can be done in pairs with the security implementation or separately depending on the effort estimate and the threat level and risk involved.

In sprint, risk evaluations would be carried out in almost an equivalent manner. The reason it is important to have this form of evaluation is: 1) the design is more elaborate and thus the risks are clearer. 2) In anticipation of changing requirements in the iterative agile environment. Also, tests might reveal new risks that were not factored into the initial backlog.

3.2.2 Agile Security Engineering

This is where the security is built in. SSE-CMM security engineering area general purpose is solving engineering problems involving security. It seeks to determine customers' security needs, develop solutions and guidance on security issues, coordinate the security engineering with other involved engineering groups involved in the project as well as monitoring the security posture (ISO, 2008). The security engineering consists of the PAs specified in table 2.2. These can be summarised as taking the security inputs, designing, and implementing the right solutions that best suit the organisation's goals as well as meeting user needs. It emphasises that Security engineers should coordinate their activities with other team members.

From this, security engineering, like other engineering disciplines, is a process that proceeds through concept, design, implementation, test, deployment, operation, maintenance, and decommission, though it is important to note that SCRUM does not provide any process (sequence of activities) of actual development like waterfall but rather collaboration and schedule of meetings.

In this research, this collaboration is viewed as a strong point in Scrum since it is one of its main emphases. We make use of Scrum's transparent mechanisms to coordinate security. Security Engineers (in case of any), as other members of the team, should be able to give daily feedback to the entire team in the Scrum meetings. Also, we propose involving everyone in the risk meetings, to promote a mutual understanding of security for every stakeholder including the users, hence satisfying PA10 (ISO, 2008).

Security engineer is a role which can be acted by a team member or an invited domain or security expert, depending on the stage of development. Since scrum has a fixed number of members for a developing team, increasing the individuals would be redefining the framework. Scrum teams are supposed to be cross-functional, and therefore assigning a security expert as part of the development team does not alter the dynamics of the framework. For highly security critical software, a security expert can be brought in as a team member.

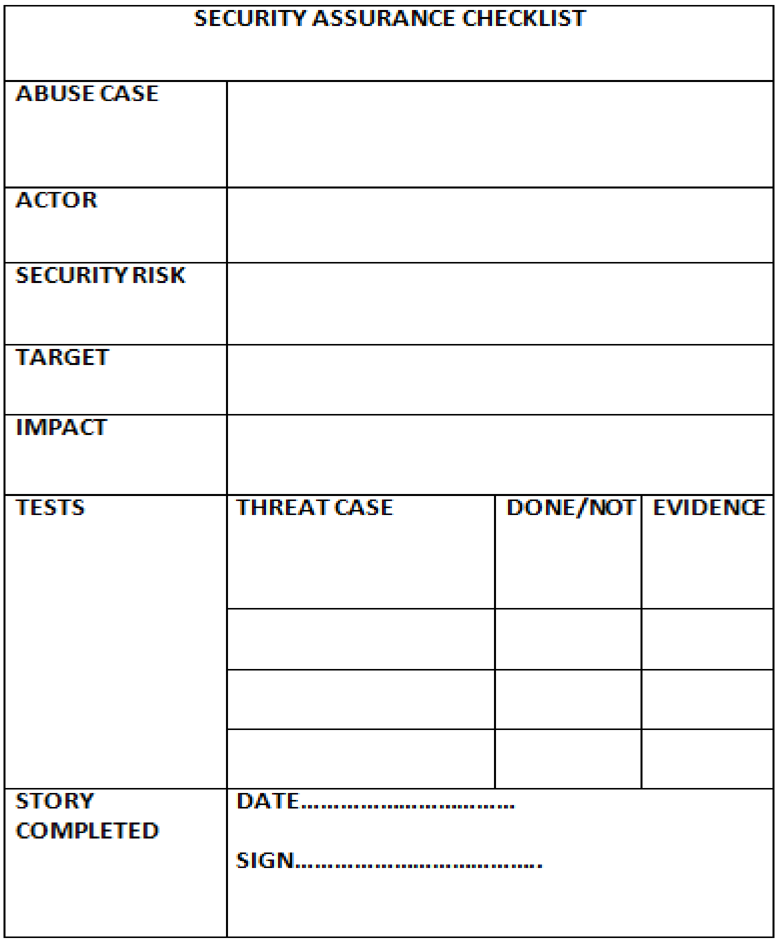
Current work also supports the recommendation from previous research of security training for the whole team to create a security aware team, at least at the basic level (Rindell, 2021).

In the proposed framework, the security engineering area will have the team design, implement, and test the security requirements. The design like any other PB item is done during the sprint planning and within the sprint. A team member (preferably with security knowledge) should be assigned to develop tests based on threat scenarios associated with the user story being worked on. Static code and dynamic code analysis are recommended. These involve the use of testing tools that a developer can subject generated code to uncover vulnerabilities within it. These tests should mark the definition of done (K Schwaber & Sutherland, 2011) for the security requirements and hence check the effectiveness of the security safeguards implemented. This provides a mechanism to monitor security posture within the process (PA08). In a case where there is a need to give the user guidelines on how to safely interact with the system implementation without creating breaches, it should be noted, discussed in the sprint review, and added into the PB as an item to be implemented by the training team or to be included in the user documentation. Tests might also generate new security concerns that should be discussed in the sprint review, and depending on the decision they might end up in PB.

This work also recommends extending the role of a product owner to also include playing the role of a security advisor who is an expert in security, preferably with some experience. The security advisor role would be; 1) Advising the client on requirement prioritization. 2) Coordinating security development. 3) Training the team on security. This will ensure that at any one point a team member can be used to implement security. Also, it ensures that no additional requirements that would be brought in by third parties would interfere with the strict timelines (Sven & Poller, 2017).

3.2.3 Security Assurance

It is considered a product of process (Mailloux et al., 2013). The maturity of the software security processes contributes to one aspect of assurance (Chaudhary & Chopra, 2017), the trust generated from a reputation of successful results achieved from a particular process system employed over time. In simple terms, the fact that a mature organisation is more likely to repeat results than an immature organisation instils trust. Traditionally and in SSE-CMM assurance is often communicated in the form of an argument which includes a set of claims about properties of the system. To support claims, evidence is required which is usually in the form of documentation developed during the normal course of security engineering activities. In scrum we have two main challenges: 1) Time is limited therefore the production of detailed documents might not be accommodated by the stipulated time. 2) Heavy documentation is not a priority in agile development.



SECURITY ASSURANCE CHECKLIST			
ABUSE CASE			
ACTOR			
SECURITY RISK			
TARGET			
IMPACT			
TESTS	THREAT CASE	DONE/NOT	EVIDENCE
STORY COMPLETED	DATE………………………… SIGN…………………………….		

Figure 3.4: The Proposed Security Assurance Checklist

This thesis also recommends documentation within code, as suggested in previous works (Pohl & Hof, 2015). Artefacts resulting from the process can be compiled as documented evidence. Examples are security stories developed.

Automatic Project management tools (for example Jira, Trello, etc. for scrum) can also be used to keep track of security throughout the process. Their reports alongside other automatically generated evidence like the automatic tests' reports can be components of the assurance argument. In case of a manual board, a member can be assigned to take security notes that will be compiled into a document.

A story completion checklist was introduced in previous research (Jakobsen & Johnson, 2008). We use the same concept to introduce “security story completion checklist” (Figure 3.4) to complement and add to the assurance argument. The product owner prepares it by filling the top part from security board details. Then the team fills the middle part as they design the tests. After the review demonstration, the product owner will fill in the evidence provided, and finally during the retrospective the validity will be assessed before signing. This provides a strong mechanism to monitor security and provide evidence.

This chapter has provided an overview of the proposed framework design; it has given general expectations at every level. For each component, agility must be maintained as well as meeting the goals stated in SSE-CMM. In the next section the framework is subjected to tests, apart from the earlier mentioned principles of agility, agility will be measured with a4-Dimensional Analytical Tool (Asif Qumer & Henderson-Sellers, 2006b). This was discussed in a previous section, as this work finds it simple as well as capturing all the aspects of agility.

In achieving this it is the belief of this research that the goal of a framework for secure agile software development will be achieved.

3.3 The proposed framework and its implementation in scrum

Figure 3.1 gives a high-level description of the proposed framework, showing the area of implementation or influence in the general Scrum process for each level of the framework. Note that the three security areas defined do not have a definite order, and sometimes overlaps are expected. As mentioned earlier, it is the security goals that we seek to achieve rather than define a strict process. Note that most of the actual implementations are left to the agile teams to determine.

To achieve security following the standardised SSE-CMM model, standard scrum had to be enhanced. We seek to do this while maintaining the identity of scrum, i.e., we are not introducing a new framework but rather achieving security within scrum.
The contributions of the framework to standard scrum are as listed below:

1. The security sprint zero, accommodating a semi structured risk analysis approach as a means of gathering security requirements, setting security goals, setting security metrics, and agreeing on security policies e.g., access policies etc. it also promoted awareness across the board.
2. Security requirements presence in the PB.
3. The security story completion checklist, as an added artefact. To enhance monitoring as well as strengthening the assurance argument.
4. Within sprints, security development, tests, and monitoring.
5. Security assurance in the sprint review and retrospective meetings.

The risk process:

It is used to identify security issues and assign value to them according to the risk calculated. The issues identified are included into the product backlog as abuser stories. It is implemented during the initial sprint zero, product backlog refinement and in the sprint planning and sprint review.

In the security sprint zero, vulnerabilities are identified and analysed, threats are also identified and characterised by modelling the threats, and impact is assessed by assessing the risk models developed.

In the PB refinement meeting security user stories are developed, security coding guidelines are set, prioritisation and estimations are also calculated.

Sprint planning involves, decomposing the security requirements in the form of user stories to tasks and the cost calculated. Also, this is the stage where decisions are made on the compromises to be made in cases where security might affect other factors like efficiency.

Note that at this stage the understanding of customers' security needs is further made concrete and security requirements are being determined, this has slipped over to security engineering but are all influenced and guided by the abuser stories developed in the risk process.

Security engineering:

This is used in sprint planning and daily scrum. In SSE-CMM it solves security engineering problems by determining customer security needs, developing solutions and guidance, and monitoring security posture of the system.

In this framework, security engineering activities start at the sprint planning meeting with determination of customer security needs and arriving at specific solutions. It continues through the daily sprints where the actual modelling, implementations and testing are done. In this case, abuser case guided testing might be necessary as well as other security tests such as static code analysis. In case of emerging security risks after testing, abuser stories are drawn and added into the PB to be considered in the next sprint planning meeting.

Assurance:

Confidence can be communicated through test results e.g., results from automated tests like static code analysis, confirmed through definition of done defined for each security user story, and lastly in the sprint review discussions, security goals not met will be discussed and improvements to the process will be suggested. In these, a sufficient argument can be drawn to ensure customers' confidence about the security of the system is upheld.

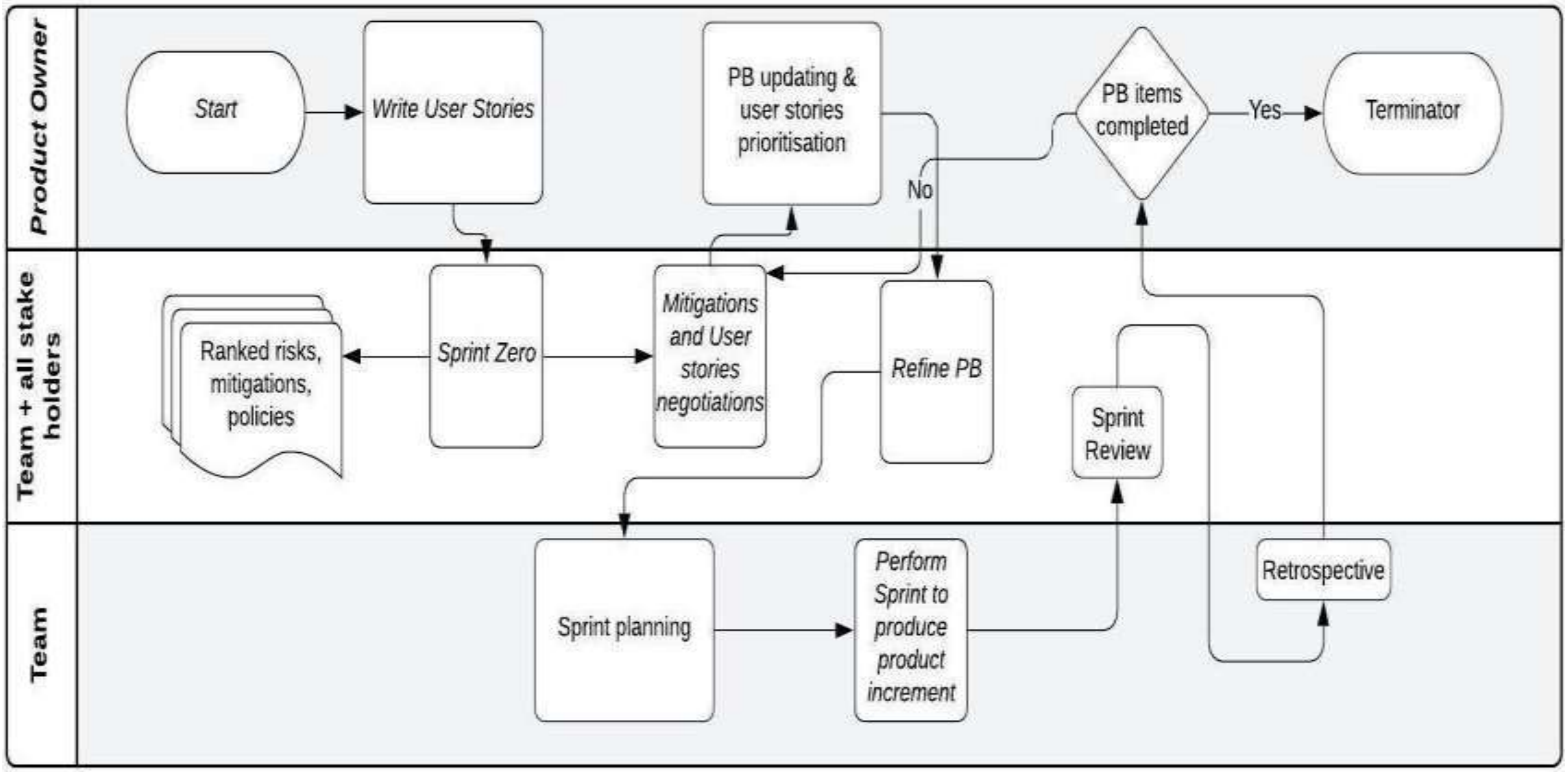

Product Owner
Team + all stake holders
Team
Start
Write User Stories
PB updating & user stories prioritisation
PB items completed
Yes
Terminator
No
Ranked risks, mitigations, policies
Sprint Zero
Mitigations and User stories negotiations
Refine PB
Sprint Review
Sprint planning
Perform Sprint to produce product increment
Retrospective

Figure 3.5: Proposed Framework Activities Within The Scrum Circle

CHAPTER FOUR

METHODOLOGY

4.1 Introduction

The methodology assumed for this work is a combination of several techniques that were modelled to form an empirical research, designed to meet the goals of this work. This work assumes a design science research (DSR) approach (Wieringa, 2014). DSR is the design and investigation of artefacts in context. The artefact in this case is the framework designed to interact with the problem context (security in agile development) to improve the situation. This work aims at improving the performance of agile methodologies where the development of secure software is concerned. It is an improvement problem and therefore a design problem (Wieringa, 2014).

A major strength of DSR is the practical exhibition of the utility of the artefact by placing it in the area in which it is designed to work and using it to solve a real problem (Peffers et al., 2007). DSR provided room for the design as well as demonstrate the hypothesis of the research.

Figure 4.1 gives an overview of design science. From it, two concepts are introduced; design problem and knowledge questions.

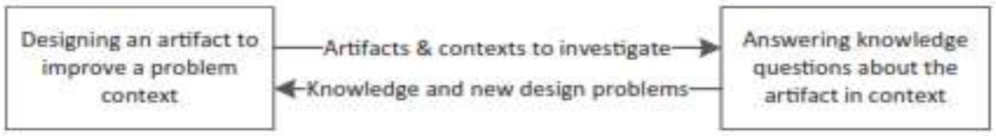

Designing an artifact to improve a problem context
Artifacts & contexts to investigate
Knowledge and new design problems
Answering knowledge questions about the artifact in context

Figure 4.1: Design science iteration: between design problem and answering knowledge questions.

The design in this work commenced with a literature review of knowledge context that aided in the design, followed by a case study aided by a combination of tools to collect, and analyse the findings. This contributed knowledge to the knowledge base. The main reason for choosing an empirical case study for the test and investigation was to get feedback from an actual group of developers involved in an actual project in which the main aim is not actually security. The intent being to generate and affirm knowledge concepts based on observation and inductive conclusions (Runeson et al., 2012). The rationale for adopting the methodology and tools for this study is discussed below, as well as a detailed elaboration of the research design.

4.2 Empirical Research in software development

Empirical research in software engineering collects and studies quantitative/qualitative data to understand and improve the software product, software development process and software management (Xu, 2017). In simple terms, it is a test that compares convictions to what we observe. It is crucial in such disciplines since it allows for incorporating human behaviour into the research approach taken (Easterbrook et al., 2008). The main motivation for an empirical study is that, from an engineering perspective, it is needed to allow for informed and well-grounded decisions. Empirical methods are best suited to gather information about the costs and benefits of software tools and methods, hence promoting founded decisions.

Empirical research methods are classified into controlled experiment, case study, surveys, and post-mortem analysis. These methods all have known flaws, and each can only provide limited, qualified evidence about the phenomena being studied. However, each method is flawed differently, and good research strategies use multiple methods, chosen in such a way that the weaknesses of each method are addressed by use of complementary methods (Runeson & Höst, 2009). Runeson et al. (2012) concluded that complementary between both qualitative and quantitative research methodologies could provide better solutions and eliminate limitations and bias of individual approaches.

In this work, a case study approach is chosen as the main method since it is a thorough analysis of one or more objects (cases). Through it, a broad and detailed knowledge of these objects is acquired (Runeson, 2009). This will deepen the knowledge about a problem not sufficiently defined, to encourage understanding, suggesting hypotheses and/or development of the theory (Wieringa, 2014).

The problem in this research is to try to investigate whether security can be implemented in a scrum project. The proposed solution introduces a new phenomenon as a theory to the agile environment that calls for investigations to gain detailed knowledge about it as well as ascertain hypotheses. Runeson et al. (2012) champions case studies in software engineering as a tool that facilitates the testing of theories and the collection of data in an “unmodified setting”.

4.3 Research Design

It refers to the overall strategy chosen to integrate different components of the study in a coherent and logical way, thereby facilitating the effective address of the research problem; it constitutes the blueprint for the collection, measurement, and analysis of data (Runeson et al., 2012). In the process of selecting a method or methods for a particular research problem, one must tap into its strengths, while mitigating its weaknesses (Easterbrook et al., 2008, p. 62). Validity of the results is determined by the ability of the research design to compensate for the weaknesses of the methods.

An important aspect of design science in relation to this research is the knowledge context. This consists of existing theories from science and engineering, specifications of currently known designs, useful facts about currently available products, lessons learned from the experience of researchers in earlier design science projects, and plain common sense (Wieringa, 2014). In this work, it would be made up of knowledge garnered from literature, including standards like SCC-CMM, previous research on this topic, etc. This knowledge is used to create or design the framework. The research seeks to add to it by producing an innovative design and answering some knowledge questions.

The commencement point for this study was to understand security engineering concepts, including their goals and objectives and their impact in an agile development life cycle found in the existing literature that made up the knowledge context. At this point, a qualitative research approach in the form of a literature review was adopted. The surveyed literature covered agile methodologies with an aim to give a brief background on the practice, and on security engineering. There was also an inclusion of some CMMs that are security oriented to enhance the understanding of the study area. In understanding security engineering discipline, this work also embarked to investigate their impact on agility. Then a study on some recent research on security in agile software development was done. This literature was meant to aid in the understanding of available contributions, their contexts, general current state of implementing security in the software development industry, and recommendations that have been put forward for further improvements. This contributed to designing the proposed theoretical framework.

After the conceptual framework was designed, there was a need for it to be tested and the performance observed for improvement. At this stage, a case study approach was adopted to observe the phenomena in an "unmodified setting". The data gathered (both quantitative and qualitative) were analysed to validate either a success or failure verdict. This approach was both exploratory in that feedback from the developers was used to improve the framework in the next iteration, and confirmatory in that it either confirmed the hypothesis or failed it. Concepts and practices of software engineering taken from experiments and observations should have empirical observations. Easterbrook et al. (2008) argue that empirical validation helps link theory with practices. They also encourage the same, especially for academic study of software engineering projects.

4.4 Research Methods

It is sometimes necessary to combine several research methods in order to fully understand a problem (Easterbrook et al., 2008). The combination of several research methods provides a platform where the flaws of one method are complemented by the other. In selecting a research method (or methods), several factors are put into consideration, for instance the theoretical stance of the researcher(s), access to resources (e.g., students or professionals as subjects/participants) and how closely the method aligns with the question(s) that have been posed (Easterbrook et al., 2008).

The general goal of this research work was to assess the perception of Scrum team members on the implementation of the agile security development framework in terms of agility and also test its impact on the resulting software. An additional intention was to generate knowledge through observations that will be used to improve the process and test the improvement.

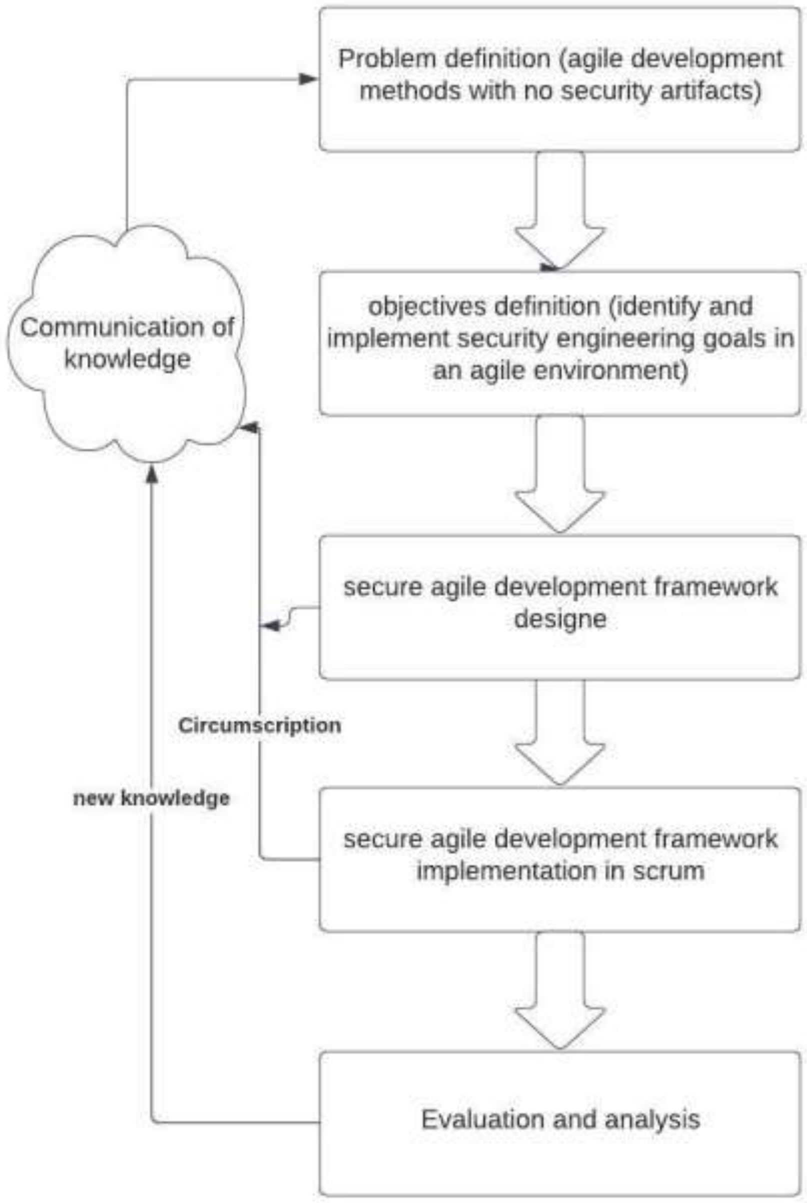
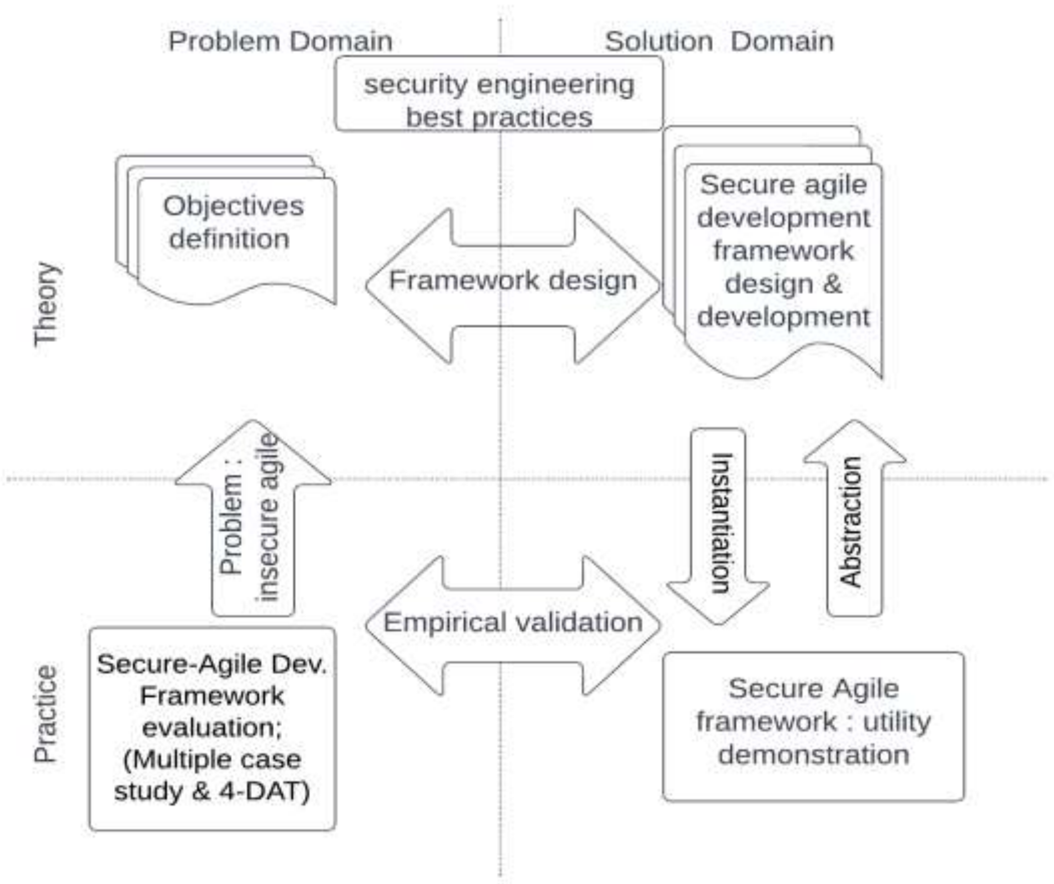

Problem definition (agile development methods with no security artifacts)
objectives definition (identify and implement security engineering goals in an agile environment)
secure agile development framework designe
secure agile development framework implementation in scrum
Evaluation and analysis
Communication of knowledge
Circumscription
new knowledge

Figure 4.2: Knowledge Flow in the Research design (Vaishnavi, Kuechler & Petter 2017)


Problem Domain
Solution Domain
security engineering best practices
Objectives definition
Framework design
Secure agile development framework design & development
Theory
Problem : insecure agile
Instantiation
Abstraction
Empirical validation
Practice
Secure-Agile Dev. Framework evaluation; (Multiple case study & 4-DAT)
Secure Agile framework : utility demonstration

The arrows illustrate the knowledge-creating activities, and the boxes represent the levels and types of knowledge that is created.

Figure 4.3: Interplay between problem and solution, theory and practice of the research. (Engstrom et.al.)

A case study was the preferred approach. The data would be collected via a triangulation of observations, structured interviews and questionnaires. The case study was used in order to observe the novel concept to the Scrum development framework, and the questionnaire further complimented by structured interviews was used to determine if the introduction of these factors would achieve security goals and have an impact on the development process from the perspective of the Scrum team members. The interviews enabled the interviewer to ask arising questions and therefore extract more information.

4.4.1 Methodological triangulation

In order to gain a broader picture and thus gain a stronger argument for the conclusion, a case study complemented by a survey was used. The case study is exploratory in nature while the survey was descriptive (Runeson, 2009). Further, the kind of case study picked was representative in nature of the normal programming environment. In order to provide data source triangulation, data was sought through different methods and approaches like observations and interviews. The following section describes the two methods.

4.4.2 Case study

A case study in software engineering is an empirical enquiry that draws on multiple sources of evidence to investigate one instance (or a few instances) of a contemporary software engineering phenomenon within its real-life context, especially when the boundary between phenomenon and context cannot be clearly stated (Runeson et al., 2012, pg 12).

The applicability of case studies in software engineering has been discussed before (Peffers et al., 2007; Runeson et al., 2012). The case study in this thesis has deductive characteristics by trying to test and apply theory in a real-world setting. They provide a platform for scrutinising the phenomenon (proposed framework) in its natural settings.
In this work, the case study was conducted with minimum interference from the researchers to avoid bias. The study chose a case which took a period of one month and employed various ethnographic data collection techniques such as observation, structured interviews and questionnaires. The literature survey was used to build the base concept and the case study was used to investigate the framework in a real development environment with an aim of proving the hypothesis as well as generating knowledge through observation as well as extracting developers' opinions through structured interviews. The detailed research design follows in the next section. An additional questionnaire survey was also included to further ground the evidence.

4.4.3 Justification for Case study

Case study research has a proven track record in capturing knowledge in real life settings, and in the subsequent extrapolation and presentation of theoretical findings (Runeson & Höst, 2009). Furthermore, case study research is usually employed when the researcher has little or no control over the events occurring within the real-life context; and should be used to contribute to existing knowledge of “individual, group, organisational, social, political and related phenomena”(Easterbrook et al., 2008), especially when how and why type questions are being posed(Roopa & Rani, 2012). These depictions of case study traits complement both the research questions and the approach of this thesis, this being to explore the practices of Scrum teams as they go about delivering software systems for clients. The literature reflects an abundance of case study reports in related research fields, including agile software development (Alvarez et al., 2012; Runeson, Per, 2009; Runeson et al., 2012; Xu, 2017) and software management (Voss et al., 2002; Miettinen et al., 2010). Best practice guidelines are available in a plethora of publications, such as (Chaudhary & Chopra, 2017; Chaudhary & Chopra, 2017; Easterbrook et al., 2008; Taherdoost, 2018; Xu (2017).

Among the several types of case study designs considered, single case was found most suitable for this particular work, putting into perspective the time limit and willingness of developers to participate in research work. The need for triangulation and avoiding results subjected to a particular group dynamic was countered by the survey that was carried out. Triangulation was achieved by the increased number of data sources.

In this case, the security framework was a well formulated theory that needed to be tested. The case chosen represented a typical development case in regard to the software industry, and therefore it would generate further knowledge about the phenomenon under study. Further, the selected case had to be critical since the subject of investigation is related to security. The knowledge gained from case study has several applications; for example, in this work, it would guide in deciding between several methods and techniques look for quantitative relationships among variables and also confirm theories.

The case would also be ideal to observe other aspects of agility, e.g., the interactions and collaborations within a scrum software development environment. 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 processes in a real-time, real-life situation.

Participant's selection criteria were:

1. Knowledge or experience using scrum. Some of the participants had gained scrum experience during their internship and were quite comfortable.
2. Knowledge about secure software development, at least in theory

The group was constructed to harbour all required skills to develop the application. In this case, the key aspect of investigation was security of the resultant software as well as agility of the process. This was not revealed to the participants in order to avoid bias. In terms of security, the number of vulnerabilities generated, security requirements generated by the team and the ones that were achieved were considered. The effect on agility was also considered, in terms of the speed as well as calculating the overall using an agility measuring tool. The standards of measuring these two were SSE-CMM standardisation for security and the principles of agility embodied in the 4-dimensional tool by (Qumer & Henderson-Sellers, 2006b).

4.4.4 The Case

The case was conducted in a software development company that would not be mentioned due to a request made by the company to remain anonymous. The selected team was mainly made up of new recruits fresh from graduating with little experience in scrum. They were trained in the development framework as a company policy to develop secure software, and then given a task to develop a prototype for an online car hiring company. The company had requested the developing company to develop an online e-wallet platform where the clients can use to manage transport cost and even earn credit facilities from the application to facilitate transport payments either on public commuter means or taxis. This was aimed at giving the company an edge over its competitors. The system was supposed to be as secure as possible and also very user-friendly.

Because of time constraints, the findings were only made for the first two sprints. The results gained coupled with the survey findings were found to be sufficient to deduce a valid conclusion on the subject of research. Further, the agility of the process and the degree of attainment of SSE-CMM security process areas goals were evaluated in terms of practitioners' opinions. This was facilitated as a focus group discussion in the scrum retrospective meeting. The evaluation method required for participants to be interviewed to gather this information. They were allowed to give further details and their opinions about the process and further contributions to better the process.

4.4.5 Sampling for Case and project organisation

While sampling for a suitable industrial case some of the factors taken into consideration were; first the company had to be using agile and especially scrum as a development framework since it would be less cumbersome in terms of training to implement the concept than with a team that is not familiar with agile methodologies. Another reason for this familiarity with agile methods meant that they would easily notice anytime the addition of the security framework interferes with the agility of the whole software process. Secondly, the project had to be security critical.

The participants were preferred to have at least some knowledge or experience with SCRUM and also security engineering. Also, the particular task being web based required familiarity with the technologies to be utilised. Because of this, Purposive sampling was conducted. It is the method where the researcher chooses the sample with a particular purpose in mind. The participants were deliberately chosen based on their qualities (in this case those with experience of security and software development).

The researcher together with the company's project coordinator made the decision about the participants' prerequisites. They were selected from the pool of the company's employees and interns with the success of the project in mind, in that the cross-functional team was made up of different skills required in the project. Apart from the many experienced team members, an experienced scrum master as well as a security expert was provided for this project. This was critical in this work to maintain the agility of scrum, as well as not missing out on security concerns due to inexperience and a lack of skills.

4.4.6 Preparation Training

The training course prepared the participant for the project. It familiarised them with software development using the security framework and also some basics of security engineering.

The team was taken through training as a new company policy to give the company an edge over competitors. The participants were required to view each project as a normal development task.

4.4.7 Data Collection techniques

This work settled on ethnographic techniques in its data collection. Ethnography is often defined as a research framework in its own right (Creswell, 2009) and is commonly based on observational work within a certain environment (Silverman, 2013). It is important to note that ethnography and fieldwork are not synonymous. It is an approach used to enhance and support Case studies like the one in this study.

Ethnography provides the basis for data analysis in that it highlights not how data will be collected, but rather how this data will be interpreted. The process is flexible and typically evolves contextually in response to the lived realities encountered in the field setting (Creswell, 2009; Zhang et al., 2019). It basically recreates the situation under study for the reader.

Despite the fact that ethnography is immersed within the research environment, a significant level of separation maintained between the researcher and the participants in their work environment is necessary for ethnography to work best. In this work a balance had to be struck in that though the researcher did participate in certain project activities, he could not influence the activities of the developers and management of the project.

The guidelines and principles that were considered in the ethnographic data collection were as proposed by Zhang et al. (2019). These were found to be most appealing to the nature of this work. To begin with, he emphasised on the importance of in-depth knowledge of the environment and the settings where this work is done. For this research, the environment aspects that were taken into consideration were a company that uses scrum for its development work and also seeking to build security critical software. The experience of scrum was important, and also the scrum version had to be the traditional scrum, that is not a variant or hybrid version. All these were ascertained in the findings also, as a plus, there was a fairly good pool of skills to pick from.

The second guideline this thesis adhered to was not to strictly prescribe to early pre-determined research questions. This is in contrast to the positivist stance, which sets up the experiment early on and religiously follows that plan throughout the study. This was integral to this work since it is strongly aligned with the hermeneutic circle of thinking, whereby the fieldwork can have an influence on theory formulation and vice versa.

The importance of filtering the data collected during interviews and observation could not be overlooked. This was because of the logic that human beings are not always able to accurately portray their daily tasks or elaborate on their thought processes (Stol & Fitzgerald, 2020).

Sometimes the behavioural and organisational patterns which are mostly occasional might occur and appear to the researcher as the norm. Triangulation of different data sources including literature were considered before drawing conclusions.

To complement these guidelines, Zhang et al. (2019) proposes three basic principles for conducting ethnography as follows: First, it is essential to consider suitable techniques, such as observation, formal and informal interviewing, and viewing of artefacts. Secondly, the theory being used for data analysis influences these techniques. And lastly, the application of the chosen techniques is considerably influenced by the philosophical background of the study.

Techniques such as semi-structured interviews, observation, and artefact analysis are used in this work governed by the guidelines and principles aforementioned. The next section discusses these techniques in detail.

4.4.7.1 Observation

Observation is an important tool for data collection in a qualitative study as it builds confidence by giving a sense of "being there" (Stol & Fitzgerald, 2020), a first-hand witness if you may. Creswell (2009) has defined observation as "the act of noting a phenomenon in the field setting".

In an observational case study, observation of a case is done without intervening; i.e., any influence the researcher may have on the case is minimised.

Observational case studies are useful for implementation evaluation and problem investigation because they give potential access to the underlying mechanisms that produce real-world phenomena. Their advantage is that they may give access to all aspects of the studied phenomena. Their disadvantages are that they may disturb the phenomena (being observed is disturbing), that they give information about only a few cases, and that about each case an unanalysable mountain of data may be collected (Wieringa, 2014).

There were instances where the researcher participated in project activities such as training workshops, scrum zero and sprint retrospectives in different capacities. When participating, the researcher was mindful not to influence the outcomes.

For observation the guidelines conformed to, were as presented in Silverman (2013). They were: 1) determining which project team members and processes will be observed and for how long; 2) the observer role; and 3) the structure and method of recording observations.

Immediate recording of observations was highly regarded and adhered to. In this case our observations were mainly on the teams' behaviour with regard to agility as well as the security requirements, from generation to implementation taking note on the particular solutions and security setup.

4.4.7.2 Interview procedure

For the researcher to collect practitioners' views on the phenomenon of interest interviews were used. They provided a means for the researcher to gain further clarity on ambiguous and difficult to comprehend data originating from other ethnographic techniques (Runeson, 2009). They were used to eliciting opinions and comments from the individuals who formed part of the research environment. Interviews are used for gaining insight into a situation. They are flexible since they involve asking questions and listening for the answers, further the interviewer can frame follow-up questions to gain more understanding on the subject. In such, interviews allow for structure towards the purpose of the study and at the same time allow for the researcher to further clarify or probe into points that are raised.

Techniques and guidelines proposed by Easterbrook et al. (2008); Taherdoost (2018) provided basic directions when conducting the interviews. According to the first guideline, the interviewees were assured of the confidentiality of the information they provided. Secondly, the questions asked were open-ended, and the researcher only stated the contextual area for discussion via the use of prompts based on the security engineering process areas identified during the literature review.

Towards the end of an interview session, interviewees were encouraged to make any comments or express opinions on the research or express any other related opinions. Probing was the third technique of the interview. This occurred when the researcher was unclear on certain responses and required clarity, or when the response required decomposition. All interviews were recorded using a smart phone. This is the fourth guideline proposed by Roopa and Rani (2012), who suggests that open-ended interviews be recorded verbatim. Respondents were assured that the recording was for the researcher's ears only and would not be published as a podcast or any similar technology. The fifth guideline was adhered to by the researcher being mindful of age, education, gender difference, and not presenting himself such that he appeared superior to the respondent. The final guideline aimed to overcome vocabulary and terminology barriers. This did not present a huge problem as all interviews were conducted in English and the terminology was well understood by both the participant and the researcher due to a shared familiarity with Scrum.

4.4.7.3 Questionnaire survey

Surveys are used to basically collect information from a group of people by sampling individuals from a large population. The rationale for including a survey was in order to achieve triangulation of the methods. The case study alone could not harbour a large enough sample size to draw sufficient conclusions, hence the need for further investigations. Questionnaire surveys were particularly picked since they provide the ease of access and their similarity to full interviews (Runeson et al., 2012) which were used in the case study. Questionnaires are also recommended by SSAM as a data collection method alongside interviews and process evidence analysis.

Sampling for the survey followed the same criteria as the participants for the case study, i.e., knowledge or experience with scrum as well as security engineering knowledge. The researchers' targeted 100 participants and received 43 responses, of which 32 fitted the criteria for the research.

Questionnaire development

The questionnaires were distributed in order to solicit practitioners' perception about the development framework with respect to agility and security. To achieve this, some ideal requisites for a questionnaire from Roopa and Rani (2012) were adopted for this work. The questions had to be drawn from the research questions under investigation in order not to divert from the goals of the research. Also, SSAMs directive that the questions must directly reflect the contents of the model was adhered to. The question's design was aimed at capturing respondents' degree of agreement or disagreement with the hypothesis put forward about the use of the framework in terms of the security sprint zero and other activities within the agile development environment. This formed the quantitative element of the questionnaire, but there were few other questions that were open in order to elicit perceptions in a qualitative manner. SSAM gives a sample questionnaire but since this work aims at achieving the SSE-CMM's Process Area goals the questions were framed to solicit respondents' view in regard to this.

First, we came up with some hypothesis statements related to the research questions stated in section 5.4.

Table 4.1: Hypothesis Statements from Research Questions

H1	Standard software security can be achieved in an agile development environment.
H2	An initial risk analysis will go a long way in discovering security requirements which can be included in the backlog and prioritised factoring the risk weight.
H3	The implementation of these requirements through security activities within the agile life cycle can be achieved without affecting agility?
H4	Software assurance can be achieved in agile techniques

The following example demonstrates how the questionnaire questions were drawn from the given hypothesis:

Consider H1 from table 4.1: It takes into account the general goal of security engineering; that is, to apply scientific and engineering principles to identify security vulnerabilities and minimise or contain risks. This means that security requirements need to be identified, prioritised and managed to ensure implementation. The supporting questions derived from this hypothesis are:

> *Was it easy to identify security requirements (H1, H2)*
> *Was it easy to prioritise security requirements (H1, H2)*
> *Did you manage security requirements effectively (H1)*

For H3, considering agility in terms of the five variables; Flexibility (FY), Speed (SD), Leanness (LS), Learning (LG) and Responsiveness (RS) (A. Qumer & Henderson-Sellers, 2008), the participants are given an opportunity to give their opinion using a Likert scale. Example: In the initial sprint zero, you found it easy to adapt to changing requirements. (1) Strongly disagree (2) Disagree (3) Don’t know (4) Agree (5) strongly agree.

Table 4.2: Mapping of Questionnaire Questions to Hypothesis and Process Area

Hypothesis	Question	SSE_CMM PA	Metric/Measurement
H1,H3	Were Security controls properly configured in terms of mechanisms put in place?	01	Likert scale
H2	Was security awareness and a common understanding of security needs reached between all applicable parties, including the customer achieved?	01	
H1,H2	Were everyone's responsibilities as far as security is concerned clear?	01	
H3	Were the activities effective in maintaining a security posture?	08	
H2	Were you able to identify and characterise security impacts of risks to the system?	04,05,02	
H2	Were you able to understand the security risk?	03	
H2	Were you able to prioritise the risks in the PB?	03	
H4	Were you able to build the assurance argument to satisfactory levels?	06	
H2	Were you able to capture and monitor security requirements changes?	04,08	
H3	Was the system security designed and implemented according to the understanding established?	11	
H3, H4	Did the solutions meet the system's security requirements?	11	
H3	Was agility affected in any phase? Please describe the phase and how?		
H4	Was assurance achieved?		

Table 4.2 gives a mapping of the questions to the hypothesis; the full questionnaire is attached in the appendix.

4.4.8 Evaluation Method

To assess how well the new framework handles secure development in an agile manner, a combination of two methods were used and customised to be in line with the nature of this particular research work in terms of the scope and goals sought to be achieved. As already mentioned, the SSE-CMM Appraisal Method (SSAM) was selected reasons being it is an appraisal method primarily intended for internal process improvement (Barzin, 2007) in regard to the capability of the process in handling security engineering, and also it allows for customization to fit the goals of the specific organisation (ISO, 2008). The fact that SSE-CMM was our benchmark tool for security engineering; it was found appropriate to use its recommended evaluation tool for the evaluation. In simple terms, SSAM is a framework used to assess the availability and capability of security activities in the security process. The SSAM is an organisational or project-level appraisal method, in this work the focus will be the level achieved within the project since there are no organisations that have adapted the model at this stage. The project that was analysed was a pilot project for the process of which the researchers had gained consent from the owners for use in the research work. SSAM also recommends the use of multiple data gathering methods to obtain information on the processes being practised within the organisation or project selected for appraisal (SSE-CMM Project, 1999). The data gathering methods put forward by it consists of 1) questionnaires that directly reflect the contents of the model, 2) a series of structured and unstructured interviews with key personnel involved in the performance of the organisation's processes, and 3) review of security engineering evidence generated.

In this case, the evaluation was conducted only for the security engineering process areas due to the scope of this work. The approach taken checked if the goals of the process area were met, and not actual activities or artefacts produced.

The evaluation takes four phases:
The planning phase: The framework under which the appraisal will be conducted is established, as well as looking at the logistical aspects for the on-site phase. In this work,

the process areas to be looked at were selected from the SSE-CMM. The security engineering process areas were selected for this due to the scope of the work and also considering the fact that the framework was being evaluated for a project and not for an organisation's continuous production. The evaluation dimensions were also discussed putting into consideration agility and the process areas application in an agile manner.

The appraisal would be conducted during the sprint retrospective meeting where every member would contribute their opinions and also an interview with the product owner and the scrum master would further validate the findings.

Preparation phase: this is the familiarisation phase, where the interviewees' team conducting the appraisal is guided through the tasks ahead. For this work, since the appraisal was primarily conducted by the researcher, there was found no need for familiarisation. For the appraisal that was conducted during the training of the team, an informal interview was conducted in terms of questions and discussions to get preliminary data on the levels of security activities available. This was found to be profound in finding a point of comparison between the theoretical and the practitioner's view. It gave insight about the security engineering knowledge to the team.

On site: at this point, the practitioners are given an opportunity to participate in the data collection and validation process. In that, an interview is conducted to assess their views on the phenomenon of interest. This will be carried out during the sprint retrospective meeting.

Post appraisal: Data analysis phase where data will be analysed, and conclusions drawn. This will be presented in chapter V of this report.

4.4.8.1 Capability Evaluation

The SSE-CMM provides an appraisal method known as SSAM (SSE-CMM Appraisal Method). It provides for organisational or project-level appraisal using multiple data gathering methods to extract information about the practices within the organisations or project's processes (ISO, 2008). It provides five levels that point to the ability to deliver secure products:

- Level 1: Performed informally, - just the base practices are incorporated in the project process.
- Level 2: Planned and Tracked, - project team define, plan and check performance issues at their level.
- Level 3: Well Defined, - the organisation has adopted a disciplined approach tailored from defined processes.
- Level 4: Quantitatively Controlled, - focuses on measurements that are attached to the business goals.
- Level 5: Continuously Improving

As mentioned, we conducted post-implementation interviews as mentioned. These were within the sprint retrospective meeting, augmented by a review of process evidence. The interviews were semi-structured to give the participants room to express themselves and give their opinions on improvement. The general interview questions were designed to gather the practitioners' views concerning security guided by the SSE-CMM. Since it was a group set up, negotiations were allowed to check on differing views.

4.4.8.2 Maturity level assessment:

An important tool to use in this assessment is a maturity level model. In this work, we adopt the maturity level criteria in Table 4.3 (Riadi & Prayudi, 2016), which include ranges to each level.

This offers a more accurate identification framework for each level of maturity. Further, the analysis levels are grouped into the major divisions of risk process, security-engineering process and assurance to give clear pointers to areas of improvement that have a level of grouping capabilities of the company.

Table 4.3: Maturity Level Criteria

Maturity index	**Maturity level**
0.00-0.50	0 – Non-Existent
0.51-1.50	1 – Initial / Ad hoc
1.51-2.50	2 – Repeatable but Intuitive
2.51-3.50	3 – Defined Process
3.51-4.50	4 – Managed and Measurable
4.51-5.00	5 – Optimised

The interview questions were structured to check for the availability of SSE-CMM best practices with a focus on the goal achieved and not particular activities. The position of the questions and subjects on the dimension were then used to give them a numerical value.

CHAPTER FIVE

CASE STUDY

5.1 Introduction Chapter II revealed some limitations agile methodologies have in the development of secure software. The literature reviewed highlighted the discrepancy between agile methodologies and traditional security engineering, but it also pointed out commendable progress in the attempts to make agile methodologies more secure. In building up on that, this chapter clearly describes the core contribution of this work. Its discussions are concentrated on an implementation of the proposed framework in a scrum-controlled software development process.

5.2 Demonstrating the Utility of the Framework –The Case

The project discussed in this section is the industrial case that was selected. The application developed was an online booking system for car rental systems. It was found suitable since it is a web application, and the researchers were privileged to have access to the code. Further reasons behind choosing a web application were the manifold risks in the environment (the web) as well as the characteristics of web applications development is most compatible with agile. Rao et al. (2011) listed the characteristics of a web application building project.

In this section, we give a report on how the development was undertaken as a sample for demonstration. Only one project was picked for this purpose to avoid repetition since all the cases were carried out using the secure scrum framework, but all the results were analysed. The reason for picking this particular project was the experience and professionalism of the developers, coupled with the fact that the researchers were allowed access to the code.

One of the major shortcomings of ASD in the development of secure systems has been pointed to the failure of these methodologies in prioritising security requirements in the development (Irvine & Nguyen, 2010) this can be traced to the fact that security requirements are not really functional requirements that the user puts forth (Adams, 2015, p. 56). Most security problems are usually unintended functionalities in the system, and the user is most of the time unaware of these. It has been noted that failure of most systems can be traced to the inability to capture and trace requirements in the development cycle (Sonia, 2011). The failure of ASD to capture and trace security requirements of a system is a major impediment in the development of secure software. In an attempt to tackle this issue, this work proposes a security sprint zero to capture security requirements and to put forth an argument for their importance to the development team and also an extension of the product owner roles to that of a project security champion. Also, this work has adapted a concept of a security aware product owner so that he will advise the customer on security issues and champion security as a customer requirement to the team.

The rest of this report spans through the scrum phases of the project, from the sprint zero to the sprint review. It starts with giving an understanding of the roles with respect to security in this project. It only covers one sprint, as this was found sufficient to demonstrate applicability of the framework.

5.3 Roles

As mentioned, this section reflects on the responsibilities the scrum actors played to produce secure software. Not all SCRUM roles are affected by the proposed framework, only the product owner and the team. This is because the main goal is to make security requirements have a significant presence in the process and this cannot be achieved without the input of the product owner who is the main custodian of the product backlog. The second goal is to make the team more security aware so that they can implement security within iterations and consider security while picking items for the sprint backlog. The aim is to incorporate security minded thinking into all the phases of development. This calls for specific measures to be taken by the team to ensure this. The scrum-master's role will remain the same, with a few security inputs regarding his/her services to the PO as well as the scrum team.

5.3.1 Product Owner

The availability of a security aware product owner can be vital to creating security awareness to both the customer and the development team. It would be vital in capturing and tracking security requirements all through the development cycle. This work proposes the extension of the product owner role to include a security advisor role. This is important as:

1) He can advise the customer on the importance of protecting their business assets and how, and hence enlighten them on what to expect from a security point of view i.e., their security needs.
2) He is responsible for developing the product backlog, and hence it is unlikely for him to overlook security in his prioritisation.
3) Coordinate security during development since he works closely with the teams during development.
4) He will also be responsible in conducting security trainings to the team

The above calls for the product owner to have, as a minimum; domain knowledge of security and some security experience. The roles of the product owner will be discussed further as we move through the development cycle.

The other contribution of this work is a security sprint zero. Within it, the team plus the stakeholders were introduced to the system concept, developed a high-level architecture, and highlighted within it the business assets, threats and vulnerabilities involved and the risk impact in business value. A common general knowledge of the risk model is key and important to the developers as they develop.

In this project, the product owner was well experienced in scrum and had a good grasp on online security.

5.3.2 Scrum Master

His primary responsibility is to ensure that scrum is understood and enacted. This includes ensuring scrum theory; practices and rules are understood and adhered to. He is also responsible for ensuring that the stakeholders have understood and are practising agility (Schwaber & Sutherland, 2011). To achieve this, he offers several services to the various roles in scrum; he is regarded as a servant leader.

For the success of this framework, the scrum-master needs to be very alert in his role to keep the team from diverting from agility. In assisting the product owner in finding techniques for effective Product Backlog management and ensuring the PO has the capacity to arrange the Product Backlog to maximise value, there should also be a consideration for security in all these. For example, the choice of metrics should include metrics that quantify the security criticality of a component or requirement. Value should be redefined; it should not be limited to business gain when the functionality works properly only, but also the protection of loss if the undesired happens. In this project, the scrum-master had a scrum-master certificate coupled with more than five years in agile software development. With this, he was professionally qualified in assessing the deviation from agility whenever they would appear and correct.

5.3.3 Scrum Development Team

A key element of this framework is a security conscious scrum team. The participation of the development team in the security sprint zero introduced earlier is supposed to expose them to security in terms of its importance and the critical areas within the system in development.

In scrum, the team is self-organising and cross-functional (Schwaber & Sutherland, 2011). This work proposes at least one security expert within the team for complex projects. This will promote learning for the whole team, as well as a holistic understanding of the security requirements of the system and how to implement them. Note that Scrum does not recognize any other titles for development team members. Regardless of the role one plays within the team, to scrum he/she is just a developer. It also does not recognize sub teams within the development team; the collective responsibility lies with the team. The aim in this work is to build a team that with time and experience they can have the capacity in terms of the necessary skills to build a system that is secure within the set domain.

The teams were supposed to have enough capacity to design and implement security, as well as test and propose new security futures.

5.4 The project

The project included all phases of the application development lifecycle, combining requirements, design, construction, and test into an overall system pack that could, with further development, be deployed in the real world. In parallel, an analysis of each phase, highlighting the challenges and lessons gained, is documented, and discussed within this document.

It involved nine people: a Scrum Master, a dedicated Product Owner, a Security expert (in basic scrum, part of the development team in the role of a developer). The project brought specialists from these various teams together, at least online every day, for the 15-minute stand-up meeting. Due to the team's multiple physically separated locations, the meetings were without exception held virtually. The teams were utilised in distinct phases of the project in such a way that only the Scrum Master, security developer (i.e., the architect) and the Product Owner had personal activities in every single sprint throughout the project. The developers were part of a larger resource pool and drawn into the sprints or spikes in various phases of the project whenever their expertise was required.

The project was to develop an online platform for a small car rental company to manage their bookings and fleet. It also provided their customers a platform to book vehicles. The requirements of the system were wide-ranging. They were gathered with the help of the product owner and organised into a product backlog. Appendix 233 gives the initial product backlog.

The technologies required were Node.js, NPM registry, VS Code, Postman to test the API. The project was developed with MERN Stack as per the preference of the developers. GitHub was used for version control and Coverity Scan Static Analysis, an open-source tool for testing web apps and web APIs security.

5.5 Events

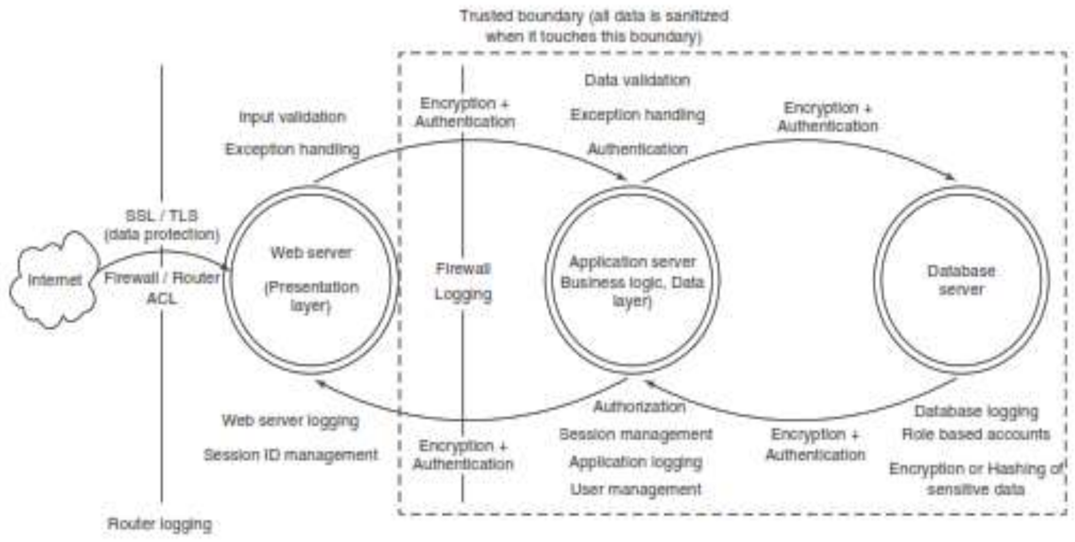
To achieve security, security engineering must be part of the development process (Ross & Oren, 2016). Scrum defines several events that are integral to it. The purpose of this section is to describe scrum events and what can be done within the events to improve the capture and residency of security within the process, including tracking security requirements, generating new requirements as well as the “definition of done” within the security concern.

This work advocates for an inclusion of a security sprint zero as a security requirement capturing tool as well as a learning and awareness building activity. Apart from this, it proposes metrics for gauging security criticality of requirements as well as how security engineering activities can be coordinated within the scrum events.

5.5.1 Security Sprint Zero

In previous sections, the sprint zero has been introduced as a pseudo sprint that does not necessarily deliver customer value. It was outlined as to what it should be and what it should not be, as it can easily be misunderstood and take us back to the traditional long and heavy pre-planning with static requirements. In (Baca & Carlsson, 2011) it was suggested that from a security engineering point of view the artefacts produced are the product, so in this case since a risk process is a security engineering procedure, we can rightfully assume a risk report to be a product. So, in this case, the expected outcome of the security sprint zero is a risk impact report, security requirements in form of abuser stories that are weighted according to an agreed metric that reflects business value.

Chapter 3 gives a high-level risk process. Note that the actual methodologies and metrics to be used are not given, this is left to the teams to decide. It depends on the project as well as the skill and experiences within the team. This work recommends that heavy documentation should be avoided, since this is against agility. The initial risk process should not be exhaustive but should give a general picture of the risks involved in running the application in the given environment. Its agenda is not to transform the SCRUM into a heavy, document centric, plan-driven process, but to build security awareness throughout the development process.



Trusted boundary (all data is sanitized when it touches this boundary)
Input validation
Exception handling
Encryption + Authentication
Data validation
Exception handling
Authentication
Encryption + Authentication
SSL / TLS (data protection)
Internet
Firewall / Router ACL
Web server (Presentation layer)
Firewall
Logging
Application server Business logic, Data layer)
Database server
Web server logging
Session ID management
Encryption + Authentication
Authorization
Session management
Application logging
User management
Encryption + Authentication
Database logging
Role based accounts
Encryption or Hashing of sensitive data
Router logging

Figure 5.1: High Level Data Flow Diagram of the Application Architecture.

The product owner with help from the security expert coordinated this sprint. The system was identified by developing a high-level architecture diagram. The PO explained that the purpose of the meeting was to develop a general understanding of the task ahead, both in terms of achieving functional requirements and other desirable qualities like security. To achieve this, it was important to determine what was being protected and why.

He took the team through an initial overview of the system, developing an architectural design of the system like Figure 5.1 above. This was guided by the initial PB. This exercise helped identify the essential components of the system and provided a visual understanding of the flow of data as well as potential attack points. In this case, the components identified included the web server, application server, and database server. Further, some security controls were also agreed upon such as encryption, authentication, input validation, authorisation, session management, etc. Trust boundaries were also determined relative to the design at this point, together with the different interfaces where the data is entered.

The security engineering activities involved were not detailed, since the design was not elaborate at this point. The overall essence of these was to conduct a preliminary secure architecture analysis to reveal the presence of security controls at the level of the system architecture. For example, in this case, a database storing confidential data requires the data to be protected by enforcing the access to only authorised and authenticated users.

A general vulnerability assessment in key areas, threat assessment, and impact analysis with respect to business loss in case of an attack at architectural level was carried out. The product backlog ensured that scope was maintained. The threat identification was carried out using the OWASP Top 10 list method. This provided consistency and an uncomplicated way of identifying and listing threats with their probability of occurrence. Other methods like STRIDE, threat trees, etc can be employed in this area as already stated in previous sections.

This will give way to an in-sprint security risk analysis that would be useful in identifying solutions as well as generating further security requirements, hence embracing agile dynamism.

Table 5.1: Partial Risk Table

Risk	***vulnerability***	***Threat***	***Threat Source***	***Likelihood***	***Severity***	***Risk rating***
R1	Unvalidated inputs	DDOs Attacks	Malicious humans	medium	critical	medium
R2	Storage mechanisms not redundant	Storage failure	Structural	Low	High	Medium
R3	Purchase validation	DoS	Malicious humans	medium	Critical	Medium

A statement of Information security goals was also developed, and privilege levels determined. From this, abuser stories were created and analysed in terms of the risk. In this work risk was in terms of qualitative business value (Table 5.1) that the loss would bring considering also the likelihood of occurrence. This was given mostly by the OWASP vulnerability report (Appendix C). In doing this, good risk information was created in the form of risk analysed abuser stories.

Table 5.2: Partial Risk Analysed Abuser Stories

Abuse case ID	***ABUSE CASE***	***RISK ID***
AB1	*As a malicious user, I want to expose people's accounts information.*	R3
AB2	*As a malicious buyer, I would book cars to create a non-existent shortage.*	R3

The last part of the process is undertaken during the product backlog refinement process. This can either be an event on its own or part of the sprint zero for small projects. This is where the abuser stories are checked and attached to a corresponding user requirement future that it is associated with. Negotiations for the mitigations are performed, and a score is attached to the requirement. This will lead into a refined product background with integrated security requirements.

5.5.1.1 Prioritisation and Estimations

The items on the product backlog should be prioritised and with estimations before the sprint planning so that the sprint planning would achieve its objectives.

The estimations and prioritisation are not inherently valuable in scrum, but they are a means to an end. In this framework the importance is a little bit higher since it gives the guidance on what objects to be picked, especially highlighting the state of security requirements within the PB. Many prioritisation and estimation formulas have been suggested (Zulkarnain, 2011; Pohl & Hof, 2015; Sillitti & Succi, 2005; Woody, 2013). This research builds on several of these ideas to produce the following recommendation. It seeks to achieve security requirements within the PB having the same residence as functional requirements. This is partially achieved through the awareness created.

Including security requirements into the product backlog bears its own challenges. Security comprises security attributes, functions, and architectural futures (Sven & Poller, 2017). The security attributes desired are quality aspects and therefore addressed during the sprint retrospective. The architectural and functional attributes on the other hand provide greater challenges in that; 1) There are some that are obvious and mostly stakeholders request for them. This will fit in the PB like any other feature. 2) There are others that can only be defined during system or feature design. This happens within the sprint. 3) Most security features are connected to certain levels to functional ones. In this case, this work adapts the approach of security tag linking user stories to security issues (Pohl & Hof, 2015).

For example, for integrating credit card payments in the system, the team had agreed on the mitigation “the transaction should be encrypted and also data should not be accessible beyond the context of the application.”

This would appear on the PB as an item “secure payment security information”. The problem with this is that the user might not be aware of this or know how to validate that it was implemented. Other security features might seem not as important as other functionalities that relate to them directly or seem to directly contribute to business value. So, for updating PB and prioritisation the following were taken in consideration;

(i) Security requirement’s risk, which is calculated considering the impact and the likelihood of occurrence as already stated. This should be taken in perspective with the general understanding of risk drawn by the process.

(ii) The security requirements relating to existing PB items.

In this work we draw an association between the PB item and the abuse case and also the risk table item so that all these can be considered and looked at together. Sometimes security requirements are related to some functional requirement, for instance in the example already given, the PB item can be related to the item “facilitate for credit card payments”. We make relationship marks that are classified in two levels; 1) Tightly tied and 2) Loosely tied. If tightly tied, it means that the item cannot be completed if the security item has not been completed, as in the case in the given example. Otherwise, the Definition of Done of the item is independent of any other item in the list.

5.5.2 Threat modelling and secure coding

5.5.2.1 Sprint Planning

In this event, the work to be performed in the particular sprint is planned. The planning involves the collaboration of the entire scrum team. It is a time boxed to a maximum of eight hours for a one-month sprint, and it answers the following questions;

1. What can be delivered as an increment of the project (software) in the upcoming sprint and
2. How will the work be achieved?

The figure 5.2 below gives an overview of the sprint planning meeting.

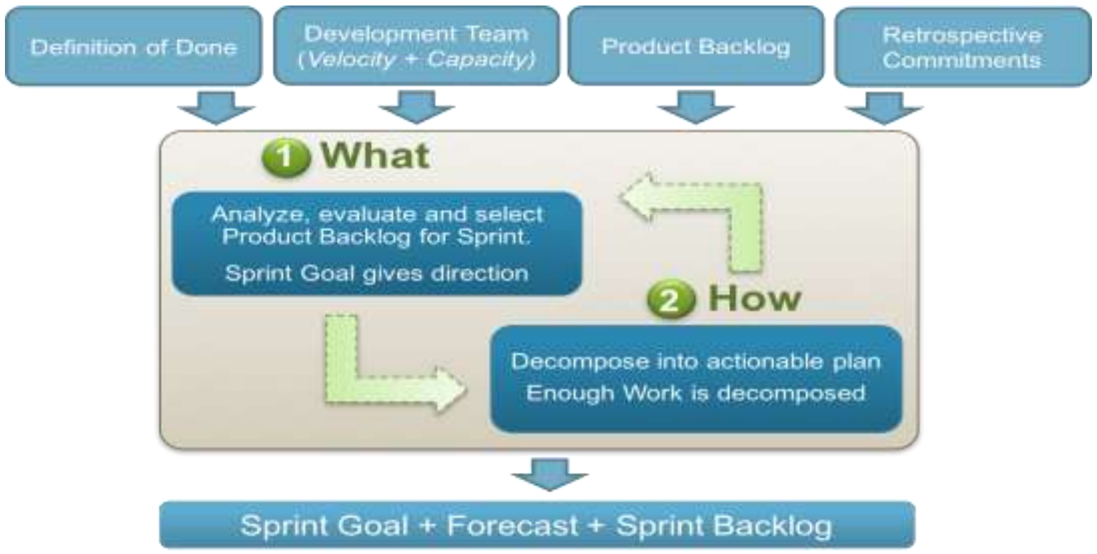

Definition of Done
Development Team (Velocity + Capacity)
Product Backlog
Retrospective Commitments
1 What
Analyze, evaluate and select Product Backlog for Sprint.
Sprint Goal gives direction
2 How
Decompose into actionable plan
Enough Work is decomposed
Sprint Goal + Forecast + Sprint Backlog

Figure 5.2: Sprint Planning

The key for a good sprint planning meeting is a good refined PB, this makes it easy to figure out what.

The sprint backlog needs to be detailed enough so that the development team can forecast the work needed. The team should keep in mind that the aim is a forecast and not a blueprint plan of the work to be done.

The framework introduced in this work provides the following benefits when carrying out the sprint planning;

- A good, refined product backlog with integrated security requirements
- Everybody has a good grasp of the importance of the security requirements

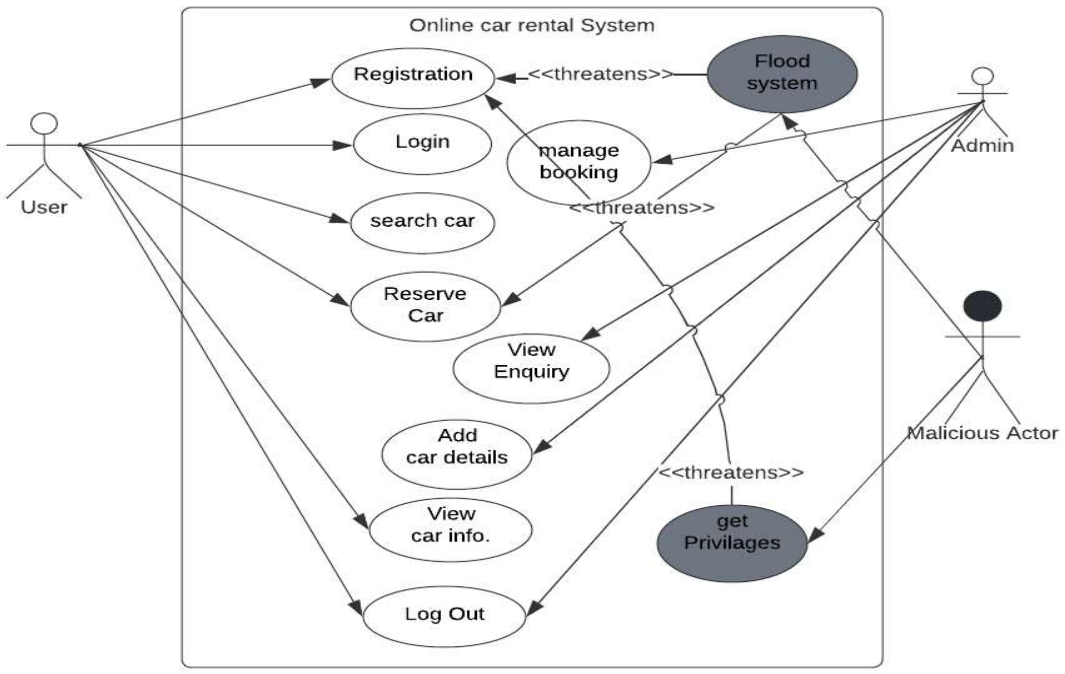


Figure 5.3: Use Case Diagram with Malicious Actors

In sprint planning, the security activities to look at are:

1) drawing of mitigations where there are two viable solutions to a risk situation:
   - o Develop security user stories; i.e., functions that counter abuser stories are conceived. For example; "*Encrypt all communications between user A and Database.*"
   - o Definition of secure coding standards; not all the abuser stories can be countered by security user stories, some are contributed by the security concerns of chosen implementation language, environment e.g., implementation platforms, frameworks, etc. for example in a case where C is the language of implementation; "all the listed vulnerable C functions should not be used."

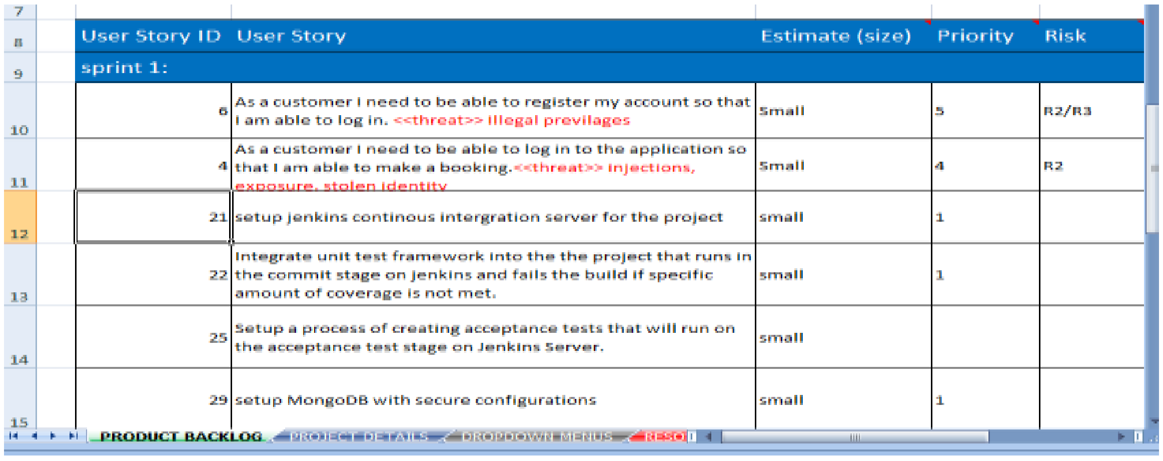
2) Security negotiations where the product owner together with the team guided by abuser stories risk analysis and cost of implementation negotiate the appropriate mitigation.

Note: The planning should be sufficient enough to help the team know just what is required of them. The end product being the sprint backlog should not be a bulletproof plan but a forecast of what is to be expected (Pohl & Hof, 2015).

In this case, the use case diagram, (Figure 5.3) was developed basically giving a pictorial view of the actors including the malicious actors. It was agreed that the first sprint would mainly involve setting up of the environment, which would involve research that would inform secure configurations to meet the requirements set. The team also picked two other user stories to fulfil.

In considering these use cases, the team considered the following security requirements:

The Sprint backlog is basically made by designing the system through decomposing the PB items taken into the sprint. For sprint 1 the backlog was as given in the figure 5.4 below.

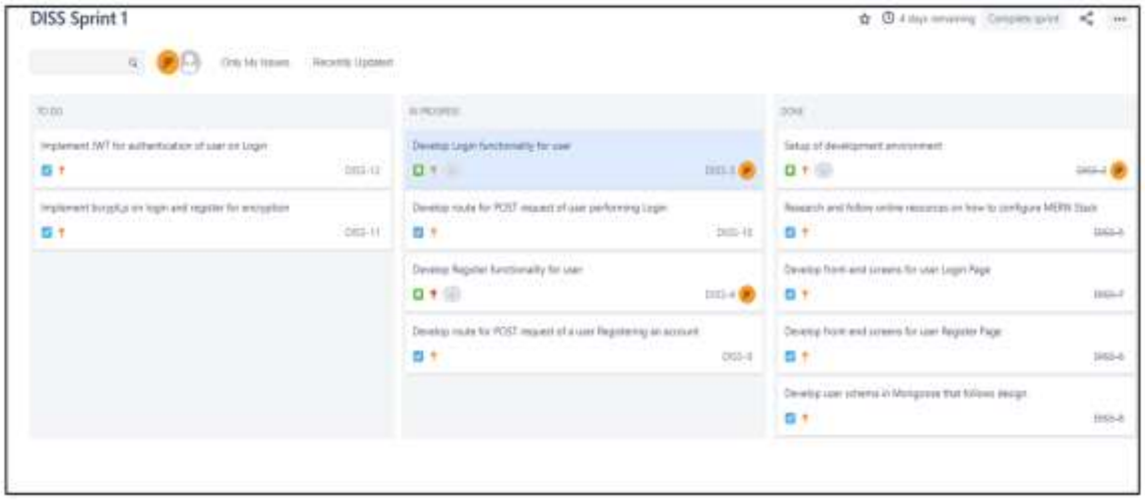


User Story ID	User Story	Estimate (size)	Priority	Risk
sprint 1:				
6	As a customer I need to be able to register my account so that I am able to log in. <<threat>> illegal previlages	Small	5	R2/R3
4	As a customer I need to be able to log in to the application so that I am able to make a booking.<<threat>> injections, exposure, stolen identity	Small	4	R2
21	setup jenkins continous intergration server for the project	small	1	
22	Integrate unit test framework into the the project that runs in the commit stage on jenkins and fails the build if specific amount of coverage is not met.	small	1	
25	Setup a process of creating acceptance tests that will run on the acceptance test stage on Jenkins Server.	small		
29	setup MongoDB with secure configurations	small	1	

Figure 5.4: Screenshot of Sprint Backlog for Sprint 1

5.5.2.2 Sprint development with code review

At this stage, the developers were set to translate their design into code. As recommended in the secure framework, developers are advised to use a development environment they are familiar with. In this project Node.js was used for back end, NPM registry as the package management, VS Code was the coding environment, Postman to test the API. The project was developed with MERN Stack as per the preference of the developers. GitHub was used for version control and Coverity Scan Static Analysis, an open-source tool for testing web apps and web APIs security. This enabled the source code to be reviewed for security issues and fix them by introducing code changes prior to release.


DISS Sprint 1

4 days remaining · Complete sprint

Only My Issues · Recently Updated

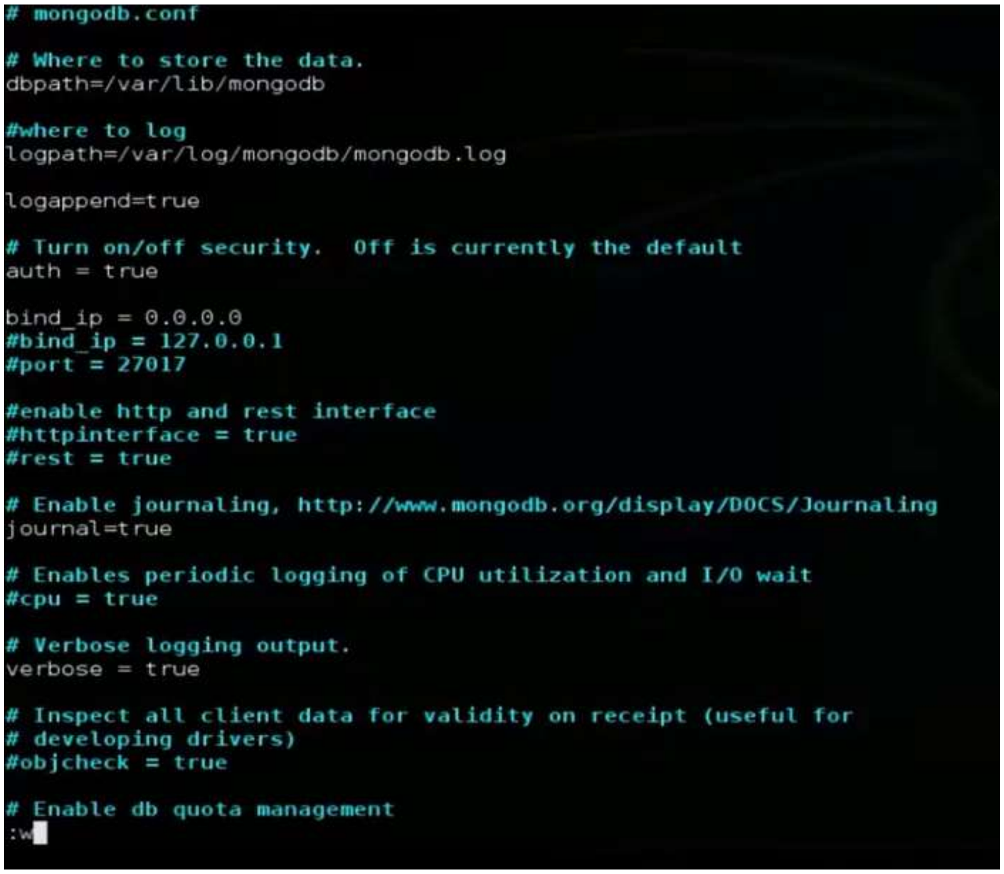
TO DO
- Implement JWT for authentication of user on Login — DISS-12
- Implement bcryptjs on login and register for encryption — DISS-11

IN PROGRESS
- Develop Login functionality for user — DISS-3
- Develop route for POST request of user performing Login — DISS-10
- Develop Register functionality for user — DISS-4
- Develop route for POST request of a user Registering an account — DISS-9

DONE
- Setup of development environment — DISS-2
- Research and follow online resources on how to configure MERN Stack — DISS-5
- Develop front-end screens for user Login Page — DISS-7
- Develop front-end screens for user Register Page — DISS-6
- Develop user schema in Mongoose that follows design — DISS-8

Figure 5.5: Jira Board for Midway through Sprint 1

Sprint 1 consisted of research and set up of the development environment. With Jenkins server for managing continuous integration, the technical stack chosen as MERN, it was now required that the environment be set up and working as expected. In addition to the Setup, two user stories were picked from the initial sprint: (i) a user being able to register an account and (ii) a user being able to log in using their registered credentials. These two user stories were chosen for several reasons. It would be the initial step for both employee and customer when they access the application, and it would also test the full functionality of the stack, as these stories involve writing to, and reading from, the database via user inputs on the front-end. The steps involved in these user stories also required the use of a Node.js framework called Mongoose.js, and several libraries. A task that was mutual for both user stories of this sprint was to define the user schema with use of the Mongoose framework. Although the database is schema less it is useful to define a schema to be loosely followed, Mongoose allows for the developer to specify what attributes are required on the creation of a user. It is worth noting that the access level field is default: 0 if the user is an employee of the company and admin will be required to alter the access level to a value of 1. These access values were implemented as per what was decided initially. Figure 5.5 shows a screenshot of the Jira board midway through sprint 1.



```
# mongodb.conf

# Where to store the data.
dbpath=/var/lib/mongodb

#where to log
logpath=/var/log/mongodb/mongodb.log

logappend=true

# Turn on/off security.  Off is currently the default
auth = true

bind_ip = 0.0.0.0
#bind_ip = 127.0.0.1
#port = 27017

#enable http and rest interface
#httpinterface = true
#rest = true

# Enable journaling, http://www.mongodb.org/display/DOCS/Journaling
journal=true

# Enables periodic logging of CPU utilization and I/O wait
#cpu = true

# Verbose logging output.
verbose = true

# Inspect all client data for validity on receipt (useful for
# developing drivers)
#objcheck = true

# Enable db quota management
:w
```

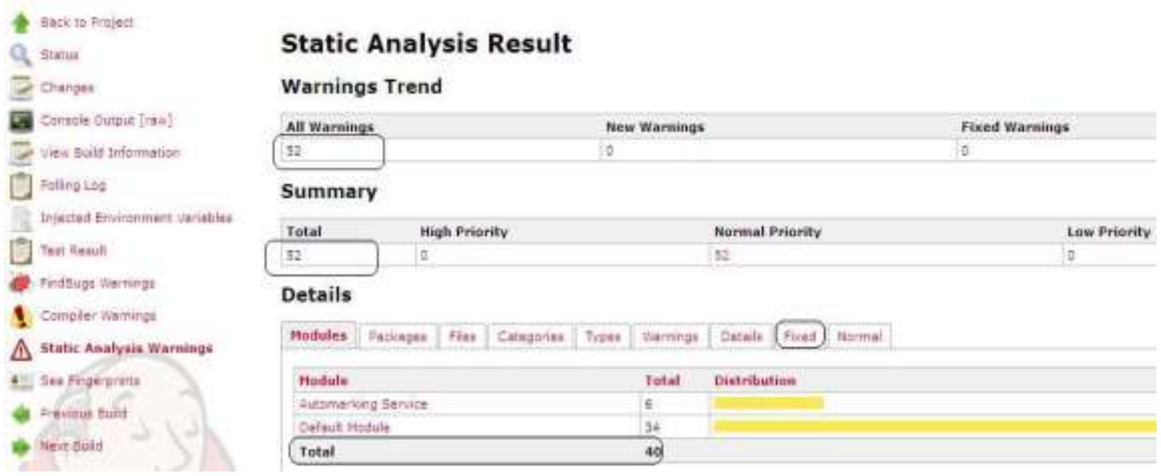
Figure 5.6: MongoDB Configuration

5.5.2.3 Testing within sprint

Configuration for MongoDB was done with security in mind with the awareness that by default Mongo Db does not have authentication. The team configuring the bind-up with only the necessary interfaces. Figure 5.6 above shows a screenshot of this configuration. Notice the authorisation has been enabled while the http and rest interfaces have been removed.

As for the login, considering the architecture diagram and the risks associated with it the story could not be completed if it did not address injections, encryption and validation. The tests are shown in the appendices.

The static analysis could not be performed at this juncture because there was no code to be analysed. Figure 5.7 show sample static analysis code that would have been produced during the integration push.



Static Analysis Result

Warnings Trend

All Warnings	New Warnings	Fixed Warnings
52	0	0

Summary

Total	High Priority	Normal Priority	Low Priority
52	0	52	0

Details

Module	Total	Distribution
Automarking Service	6	
Default Module	34	
Total	40	

Figure 5.7: Static Analysis Results

5.5.3 Sprint review

Usually held at the end of sprint to facilitate the collaborative inspection of the increment and consequently adapt the PB in case of new requirements.

It involved the Scrum team and the stakeholders holding an informal meeting where the increment is presented with the intention of eliciting feedback and foster collaborations.

All the work should meet the definition of done that the team has agreed upon. In this project, it also involved checking the security checklist and filling it up. The checklist is appended to the document.

5.6 Data collection

The main methods of data collection were focus group discussion and artefacts analysis. The researcher also performed informal observations during the development process, in particular participation in scrum meetings, observing the generated security artefacts, number of security requirements and how they were handled, as well as playing the scrum master role. During these sessions, it was possible to track participants' progress and answer questions pertaining to problems they had in applying the methodology. After using the methodology to develop their systems, a focus group discussion was held with the group to establish the views of the participants on the applicability and effect of the practices as proposed in the case theory. A two-hour long focus group discussion was held at the end of the sprint. It was conducted to collect the participants' perceptions on the applicability of the practices embedded in the methodology. A research assistant was responsible for the data capture of the responses. A focus group guide and data capturing template was prepared to help with the data capture. The group was enlightened about these contents before the meeting started by the moderator also noting key points. Appendix III bears the focus group guide, while the data capture template with sample data for the first question is attached in Appendix IV. For systematic data capturing, this work adapted Nili, Tate and Johnstone (2017)'s template. The template was designed to capture ten participants' responses per question, but since the focus group discussion had numerous participants (thirty-seven) a modification of the template designed using Microsoft Excel was used. After the discussion, the researcher's captured points were synchronised with the research assistant notes. The researcher also analysed the documentation for intermediate artefacts defined in the methodology. The intermediate artefacts' analysis was done to confirm the perceptions of the participants on the usability of the secure framework. Intermediate artefacts analysed include prioritised product backlog, sprints, and sprint backlogs, use cases and misuse cases, design models, test cases, contents of the security repositories, security, and reasons for not having them.

In the following subsections, results of the data collected in this case study are presented. The next section focuses on the group discussion results. This work has adopted editing and template analysis as suggested by Wohlin et al. (2012) mainly because of the use of pre-formulated questions during the discussions. The responses collected by the researcher and the research assistant were synchronised into a single document. Responses were captured per question. Table 5.3 shows general opinions gathered about the emergent process.

Table 5.3: General Opinions from Focus Group Discussions

	Question	Responses
1	Is the Agile Secure Development Framework a solution to a real problem/need in the agile software development environment currently?	-To some extent, when working on some crucial software one needs guidance. -I found it to be filling a gap that exists at the moment -It is to some extent. - Due to my lack of experience, I can't tell.
2	Would you rate the practices embedded in the methodology adequate to build quality and secure software, if not what would you add?	-The practices are adequate -the help of more automated tools to support the process would greatly improve the utility of the methodology.
3	Was it easy to follow the practices in the process? Which practices would you consider helpful, and which would you consider to be not?	- The methodology is not that easy to follow -Following a methodology while developing software is not an easy task - The lack of security engineering knowledge and experience hindered application of the process. -At times users do not have time for meetings -Security design and testing are not easy
4	Did you at any point feel you were asked to do more than just developing software?	- security engineering seemed like a derailment to the actual work of software development -There seems to be more documentation and extra meetings -Models can be used selectively
5	What available tools would you suggest to ease the development process at any of the methodology stages?	-Jira, Trello, etc. for project management -Brackets (free open source front end editing and web development) -Bootstrap eases website development -IBM Watson Assistant API -Node.js, supports both front end and back end development

6	What practices in the framework would you consider to be important in developing quality and secure software?	-All the practices are necessary, but that should depend on the kind of software -the secure sprint zero seemed to be the core of the process -Secure coding to me was new, and I feel is key -To me, designing test cases seem to serve for the expected quality
7	What improvements would you add to the methodology if you were given the opportunity to?	-Automate most activities -At times users are too busy for user education -User education should only highlight the user's roles
8	Would you consider using the Secure process in your future projects?	- I would use it on serious projects -Yes, it brings order into the development process -I would, it makes the user think I know what I am doing -I would use it but trim some practices
9	Would you recommend the methodology to any fellow developers?	-I think developers should adopt the methodology, particularly for online applications -just to be used on large and critical projects
10	Do you think the Secure process can be used to develop any kind of software system?	- To some extent -I feel it can be adapted to any environment

A phase-by-phase evaluation of the methodology was also conducted through the questionnaires and the results are as shown in table 6.1.

CHAPTER SIX

DATA ANALYSIS AND DISCUSSIONS

6.1 Introduction This chapter presents the findings of the evaluation cases. The data analysis objective is to test the hypotheses of this research. The groups engaged in the research were questioned about their experience. As already stated, the questions were designed to reveal if there was an effect on the agility and the management of security requirements as well as assurance. This was required to verify the hypotheses of this study. The results include a comparison between the diverse groups and correlation analysis.

From the group discussions, the participants agreed that the secure agile development framework was a solution to a real problem that sometimes goes unnoticed, especially to novice developers. The general perception was that the framework can be used in development environments to build quality software and in the process creating information assurance. The practices in the methodology are perceived as important in building quality software. Some developers had reservations on the models produced in the initial stages. They argued that if not properly guided, the design can be too detailed. This reservation was also raised by the industry developers. While developers have these reservations on models, they agree that the system understanding generated went a long way in justifying why security implementations were crucial during development. Also, the risk analysis products e.g., use case and misuse cases are important in modelling user requirements. Table 6.1 also shows that developers would opt to use the methodology in future projects, and they would also recommend the methodology to other developers. They also felt that the methodology could be used to develop any type of software with minimal adjustments.

An overview of the table is as follows; in all phases, the participants found the practices adequate. However, there are some who felt that in practice it was difficult to adhere to some of the practices, citing inadequate experience and training as a major hindrance.

Table 6.1: Summary of Practitioners View of the Process in Light of Best Practices

Process Area	Best Practices	Implementation During Process (Practitioners View)
PA01- Administer Security Control	Management of security awareness, training, and education programs for all users and administrators; establishment of responsibilities, accountability for security controls, their effective communication, and management of periodic maintenance of security services.	-Everybody was aware throughout the development course. Communications was maintained through the scrum board
PA02 / PA03 / PA04 /PA05 - Assess Impact/ Assess Security Risk/Assess Threat / Assess Vulnerability	the identification of assets that support key operations of the system in focus, the selection of appropriate metrics for assessment in the different areas, the characterization and identification of impacts, risks, threats and vulnerabilities, and the monitoring of on-going changes to ensure that the understanding of processes is preserved.	Initially achieved through the sprint zero. Assets were identified, metrics determined. Within the sprints and after tests (including automated) and goals provided a reference point for monitoring.
PA06 - Build Assurance Argument	Identification of security objectives, identification and gathering of evidence supporting these objectives, defining a strategy to address the security objectives, and providing an assurance argument that clearly demonstrates that the security needs of the customers are met.	Although the process brought some level of assurance. The information on the Trello board as well as test software reports was insufficient and not easily understandable to all. It was suggested that a person be assigned to compile and create simple security documentation in future iterations as they were going on. They should be looked at as the deliverables for the SE process.
PA07 – Coordinate Security	This particular practice is concerned about security coordination and open communication among all project personnel and external groups.	Scrum inbuilt communication mechanisms assisted by the tools used were sufficient enough for facilitating these communications.
Process Area	**Best Practices**	**Implementation During Process (Practitioners View)**

PA08- Monitor Security Posture	Security Posture serves as a follow up of every external or internal security related event and ensures that all breaches and potential events that could lead to a breach are reported and addressed properly. Hence this practice is also very closely related to the vulnerability and security assessment practices previously discussed.	The knowledge promoted in sprint zero plus the security assurance checklist helped maintain this posture throughout the process.
PA09/PA10 - Provide Security Input/ Specify security Needs	The specification of security needs involves defining the security requirements of the system in order to meet the legal, functional, and organisational objectives taking into account the current environment of the system. This process should identify the purpose of the system and provide high level views of system operations, as well as the goals of the system under a security context.	The sprint zero which produced Use Cases and related Misuse Cases (Sindre & Opdahl, 2005) provided a clear view of needs, actors, and also gave guidance to the security design.
PA 11 - Verify and Validate Security	Ensures the right measures are put in place in the right way. In this process area, the applied solutions would also need to be verified against specific operational security needs of the customer.	Similar to PA06, assessment tools such as network vulnerability scanners/auditing, enterprise product solutions like TSOM (Karim et al., 2016), or in-house software validation tools can be used. In this project Coverity scan was also used. This helped to verify the applied solutions against the security requirements and architectures using testing, demonstration, analysis and other methods - while still providing traceability between solutions and requirements.

6.2 Maturity Level Analysis

As mentioned, only the security engineering process areas were considered. This was even more so because the study was being based on a single project process and not an entire organisation.

For each of the process areas, the practitioners' opinions about activities were gathered through the questionnaires, sampled and were analysed. For example, for PA01-Administer Security Controls, the chart in table 6.2 is a summary of the findings.

Table 6.2: Security Engineering Process Areas Index and Levels Attained

	Process Area	**Index**	**Level**
1.	PA01	3.65	4
2.	PA02	2.66	3
3.	PA03	2.21	2
4.	PA04	1.89	2
5.	PA05	2.06	2
6.	PA06	2.1	2
7.	PA07	2.40	2
8.	PA08	2.67	3
9.	PA09	2.45	2
10.	PA10	2.86	3
11.	PA11	2.31	2
	Average	**2.478**	**2**

It can be concluded that most of the practitioners felt that security awareness was achieved to great levels while periodic maintenance of security controls was not well-defined, managed, or measurable.

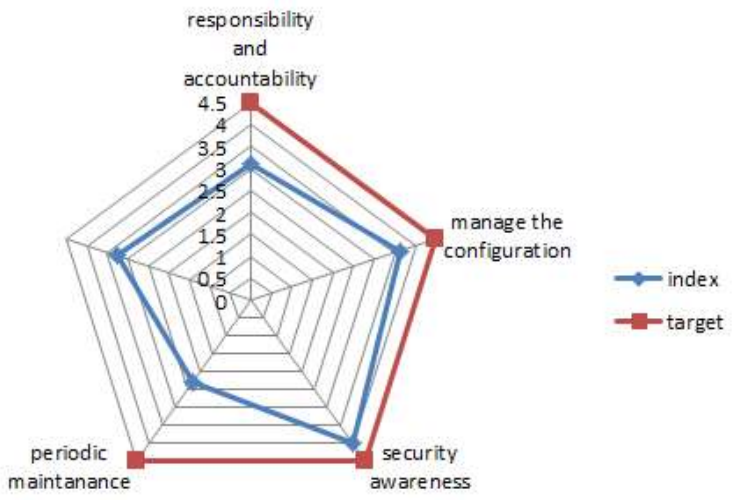


Figure 6.1: Administer Security Control

The calculated average mean for the whole Process Area was 3.26 which according to the maturity index meant it was at level 3.

A summary of the whole process area indexes is as in the table 6.2 and figure 6.4. The results were categorised according to the three main security engineering areas of SSE-CMM i.e., risk area, security engineering area and assurance areas to evaluate and compare the area of strength of the framework.

6.2.1 Security Risk Area

This area is made up of PA02- assess impact, PA03- assess security risk, PA04- assess threat and PA05- assess vulnerabilities. Note, this work does not define the particular process, but it was left for the team to define the approach that they would use. For example, in the particular case study project undertaken the team chose a reference list of common vulnerabilities to identify possible vulnerabilities, OWASP top ten in particular. Then the PO presented the client's assets, they discussed and included in the list some system assets. The threats were modelled, and impact assessed, in the sprint 0 and later designed and implemented within sprints. The summary findings in this area are given in figure 6.2.

An average index of 2.105 was calculated, which means as far as the risk area is concerned, the framework has managed to assist the organisation achieve level 2 maturity according to our analysis.

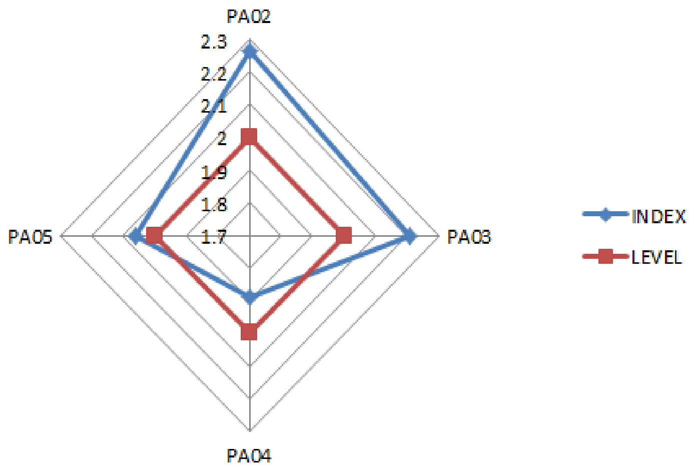


Figure 6.2: SSE-CMM Risk Area Maturity Level Attained

6.2.2 Security Engineering Area

From a security point of view, Information from the risk process, requirement, policies and even governing laws are considered to identify security needs and produce actual security requirements. SSE-CMM defines five process areas for the engineering area; PA10, PA08, PA09, PA01 and PA07. These PAs shift the process from requirement definition, through designing solutions, implementation, monitoring and finally coordination.

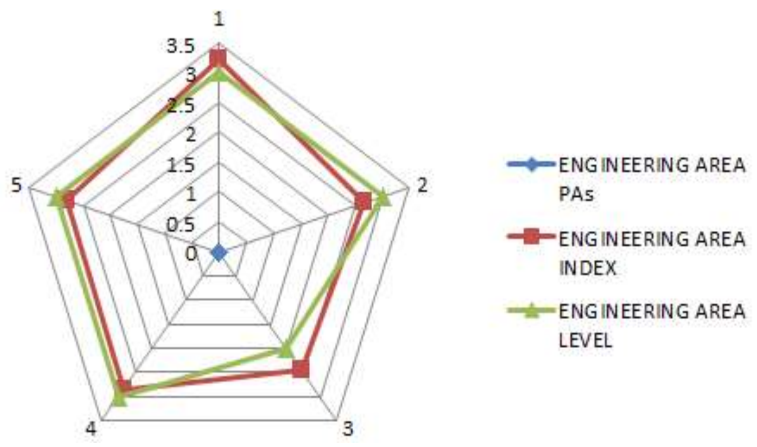


Figure 6.3: SSE-CMM Engineering Area Maturity Level Attained

The results from the questionnaire in this area are summarised in the chart, figure 6.2. The average index was 2.9075 placing it at level 3.

6.2.3 Assurance Area

This is developing the confidence that the measures taken will work. SSE-CMM guides that the Verification and validation of security (PA11) results together with evidence from other Process Areas be used to build the assurance argument (PA06). In the case study, the evidence from automated tests and security goals among other artefacts like the definition of done for security items were used to build the argument. The average score from the research was low, with a level of 2 achieved.

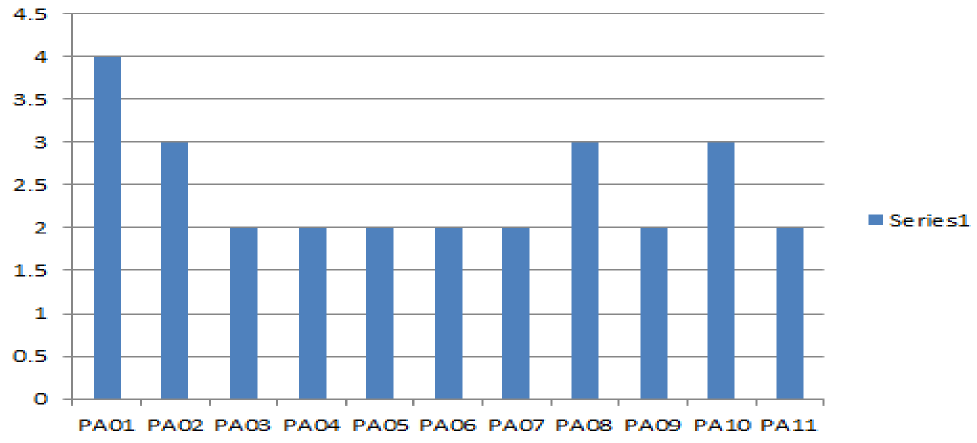


Figure 6.4: Maturity Level Summary Graph

6.3 Agility Level Analysis

In this subsection, the final version of the resulting methodology is evaluated using the four-dimensional analytical tool (4-DAT) (Qumer & Henderson-Sellers, 2008; Asif Qumer & Henderson-Sellers, 2006b). This model evaluates the secure development model compliance with agile principles. It has been used in other research (Ghani et al., 2014; González-Sanabria et al., 2017; Sánchez et al., 2020) for similar purposes.

Using this model, a methodology is assessed according to four dimensions. The four dimensions are method scope, method agility, agile values characterisation and software process characterisation. The4-DAT framework is a flexible framework. Method evaluators (or researchers) can choose what dimensions to evaluate the methodology against, depending on the purpose of the evaluation. In this thesis, only two dimensions are applied; the method agility and the agile values' characterisation since these met the goal of the evaluation.

Method agility evaluates the existence of five agility features, (i.e., flexibility, speed, leanness, learning and responsiveness) in the method practices and phases. The framework suggests the computation of the degree of agility of each method component as a fraction out of five, based on whether a feature is available (1) or not (0). Degrees of agility can be computed for both practices and phases in a methodology. Agile values characterisation seeks to identify those components of the methodology which portray the six agile values.

Just to mention, the method scope is normally used to perform a high-level analysis of a given method (Qumer & Henderson-Sellers, 2008, p. 285) including at the organisation level, which is beyond the scope of this work. Similarly, characterisation evaluates a methodology's processes for their ability to support project management as well as process management. In this dimension, a methodology is evaluated to check its coverage of the system's development cycle (SDLC). Process characterisation also checks for the existence of practices focused on project management, and therefore it falls beyond the scope of the current work.

In the 4-DAT framework, flexibility is defined as the measure of the ease with which an object or process accommodates emergent changes. This assesses the ability of method processes to respond to changes. Speed is the time taken by a process to obtain the expected deliverables. Leanness refers to the general resources used by a process to achieve a desired outcome. Learning assesses the ability of a process to support knowledge management. Knowledge management is important in agile since it anchors process improvement.

Finally, responsiveness is the ability to adapt to the environment (Asif Qumer & Henderson-Sellers, 2006b)

Table 6.3: Agility level attained by the process

Process phase	**Agility Features**					
	FY	**SD**	**LS**	**LG**	**RS**	**Total**
Sprint Zero	1	0	0	1	1	3
Sprint	1	0	0	1	1	3
Reviews	1	1	0	1	1	4
Agility	3/3	1/3	0/3	3/3	3/3	10/15

Key 1: FY- Flexibility, SD- Speed, LS-Leanness, LG-Learnability, RS-Responsiveness

Table 6.3 shows the agility level attained by this framework. Sprint zero and in sprint produced the least agility. It was agreed that it was easy to manage knowledge, and it had sufficient mechanisms to respond to changes, but it lacked speed and leanness. This was attributed to the extra documentation involved, plus the fact that the framework was being used for the first time by the group. The inadequate knowledge of security engineering as a whole in the team also contributed to the slowness of the process. In the second sprint there was slight improvement in this, and it was concluded that with practice it can be refined further.

All the phases of the resultant process have agile degrees greater than 0.5, with the stages of the methodology having an average agility degree of 0.67. A value of greater than 0.5 is regarded as agile. In addition, the analysis of the practices embedded in the framework shows that all the agile values are supported by at least one practice in the methodology. The practices: individuals and interactions over processes and tools; working software over comprehensive documentation; responding to change over following a plan; and keeping the process agile are each supported by three or more practices. Customer collaboration over contract negotiation is supported by two practices, while keeping the process cost-effective is supported by one practice. This evaluation therefore serves to show that the secure framework fulfils one of its objectives of maintaining agility in the overall process.

CHAPTER SEVEN

CONCLUSION AND FUTURE WORKS

7.1 Introduction This work developed a framework for agile security and implemented it in scrum by adding a secure sprint zero among other security practices. Its aim was to achieve a resultant process that meets industrial standards. To test it, a case study was undertaken, and data gathered through questionnaires and interviews of some key players. The results from the questionnaires produced a 2.48 average index, which meant a level 2 (planned and tracked) of maturity for that particular project. Note that it only missed 0.02 points for it to make it to level 3. The model scored fairly highly on the risk area, high on the engineering, but poorly on the assurance area. Further, the strengths of the model were; provision for gathering of security requirements, involving everyone in the project, disseminating responsibilities and providing a mechanism for monitoring security. As for the assurance argument, the feedback suggested dissatisfaction in the evidence, the evidence was scattered and sometimes needed interpretation.

Table 7.1: How the process fulfils SSE-CMM process areas

Sse-Cmm Security Eng. Area	**Intended Product**	**Equivalent In The Secure Agile**
Risk	Risk information	- Ranked Risks - Vulnerability list - Asset list
Engineering	Product, system or service	Secure system
Assurance	Assurance argument	- Stakeholders agreement since they are involved. - Security assurance checklist. - Tests and test results

The interviewed practitioners suggested a person be assigned to gather and compile the evidence and present it in a manner that is understandable to even non-technical users in order to build a stronger assurance argument. Further, it was also suggested security training and experience would also improve the levels. The general expectation is, with every iteration circle, the self-improving teams would achieve higher levels. Finally, considering the results of the case study for a single sprint, it can be concluded that the model introduced can achieve appropriate maturity levels of security for an agile team.

7.2 Research Goals achieved

In the literature review, the first and second objectives were addressed. To be specific, in section 2.3-2.5 a background discussion of agile software engineering and security engineering is discussed. Further, section 2.6 provides a discussion of some capability maturity models to elaborate further the standard status quo of security issues handling in software development. In 2.7 the impacting issues of agile and the traditional security engineering are highlighted and lastly in 2.8, work by other researchers in this area is discussed. This literature study informed the design of the framework presented in the consequent chapter.

For objective three, an agile framework based on the SSE-CMM security engineering model was developed with security engineering goals in mind. The SSE-CMM model gave the goals to be met by the agile framework. It also broke down the security engineering process into three phases i.e., Risk process which can be regarded as the requirement gathering, Security engineering and assurance. To implement the goals, agility was a key consideration. The agile design was informed by previous works as well as the agile principles. Finally, the framework is implemented in scrum.

To show practicability, a case study was conducted and evaluated for performance and also to gather recommendations for future improvements. The results were good in that the framework achieved security to acceptable standards, Agility loss was negligible as well as there was room for improvement with automation and an increase in learning.

7.3 Contributions

In this work, a secure scrum zero was implemented; it bore a semi structured, system centric risk process. It also introduces a security assurance checklist as an additional artefact to strengthen the assurance argument. This artefact proved to be simple enough to fill during the course of the process and displayed a contribution to customer's confidence in the case of the system developed.

This research work also showed that standard security engineering can be achieved in an agile environment. By basing the agile secure development framework after the SSE-CMM standard, it achieved the desired benchmark of level 1 and attained level 2.

Throughout this work knowledge was generated which contributed to the development of the framework. A scan through the research process from the literature to the design up to the evaluation it's evident that new knowledge in the area is built. In the literature, the conflicting issues between agile and traditional security engineering were highlighted and also existing solutions were discussed. The framework itself contributes to knowledge in relation to the research hypothesis in agreeing that security can be achieved and further explaining how.

The research produced a research publication (Kagombe et al., 2021) and presented in ICICM 2021: The 11th International Conference on Information Communication and Management

7.4 Limitations of the study

As a limitation in the design, the knowledge used to design might have been limited in terms of the literature studied. Some papers which could have contributed to the research were not accessible within the researchers' rights. Another limitation was experienced during the case selection. A multiple case in diverse companies would have been ideal to remove subjectivity to a particular group. To counter this, a survey was carried out.

Time constraints as well as finding the right company to carry out the evaluation were a challenge.

Another limitation is that for the most part, the evaluation was based on practitioners' perceptions rather than evidence from the case. This would have greatly countered the subjectivity of the results' argument.

7.5 Recommendations for future works

Further research can build on this work by conducting more evaluation in terms of case studies or even controlled experiments. They could go further and evaluate the quality of the products built in terms of security and compare them with others not built with the framework.

Further, a project management tool can be developed to assist the developers in working with the framework. It can include AI analysis and identification of risks. This would make work easier and eliminate sprint bottlenecks further.

REFERENCES

Abbas, J. (2016). Quintessence of traditional and agile requirement engineering. *Journal of Software Engineering and Applications*, *09*(03), 63–70.

Abdel-Hamid, A. N., & Hamouda, A. E. (2015). Lean CMMI: An Iterative and Incremental Approach to CMMI-Based Process Improvement. *Agile Conference (AGILE), 2015,* 65–70.

Adams, K. MacG. (2015). *Non-functional requirements in systems analysis and design*. New York: Springer.

Agile adoption in organisations. (2016). In *the Art of Agile Practice* (pp. 402–443). Auerbach Publications. http://dx.doi.org/10.1201/b13085-16

Alotaibi, M. (2015). Extending Scrum Framework to Emphasise Security: How a 'Security Owner' is integrated into the Scrum Team. *Proceedings of the Eighth Saudi Students Conference in the UK*.

Alvarez, A., Matalonga, S., & Feliu, T. S. (2012). A case study on process composition using the Enterprise SPICE model. *International Conference on Software Process Improvement and Capability Determination. Communications in Computer and Information Science, 290,* 85–92.

Anderson, R. J. (2010). *Security engineering: A guide to building dependable distributed systems*. New York: John Wiley & Sons.

Association, I. S. S. E. (2002). *Information technology - Systems security engineering - Capability maturity model (SSE-CMM): International standard*.

Azham, Zulkarnain, I. G. (2011). Security Backlog in Scrum Security Practices. *2011 Malaysian Conference in Software Engineering. IEEE*, 414–417.

Baca, D., & Carlsson, B. (2011). Agile development with security engineering activities. *Proceedings - International Conference on Software Engineering*, 149–158.

Bajta, M. El, Idri, A., Fernández-Alemán, J. L., Ros, J. N., & Toval, A. (2015). Software cost estimation for global software development a systematic map and review study. *2015 International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE)*, 197–206.

Bartsch, S. (2011). Practitioners' perspectives on security in agile development. *2011 Sixth International Conference on Availability, Reliability and Security*, 479–484.

Barzin, P. (2007). SSE-CMM. *Datenschutz Und Datensicherheit-DuD*, *31*(12), 917.

Baskerville, R. (2004). Agile Security for Information Warfare: A Call for Research. *European conference on information systems (ecis)*, 13.

Beck, K., Beedle, M., Bennekum, A. van, Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., … & Thomas, D. (2001). *Agile Manifesto*. Retrieved from http://agilemanifesto.org/

Bellenzier, M., Audy, J. H. N., Prikladnicki, R., & Luciano, E. M. (2015, October). How the scrum adoption relates to productivity of software development teams? *2015 6th Brazilian Workshop on Agile Methods (WBMA)*.

Bezerra, C.M.M. Sampaio, S.C.B. & Marinho, M. L. (2020). Secure Agile Software Development: Policies and Practices for Agile Teams. *Shepperd M., Brito e Abreu F., Rodrigues Da Silva A., Pérez-Castillo R. (Eds) Quality of Information and Communications Technology. QUATIC 2020. Communications in Computer and Information Science*, *1266.*, 343–357.

Boehm, B., & Turner, R. (2003, June). Observations on balancing discipline and agility. In *Proceedings of the Agile Development Conference, 2003. ADC 2003* (pp. 32-39). IEEE.

Boehm, Barry, & Turner, R. (2004). Balancing agility and discipline: Evaluating and integrating agile and plan-driven methods. *Proceedings - International Conference on Software Engineering*, *26*, 718–719.

Bostrm, G., W yrynen, J., Bodn, M., Beznosov, K., & Kruchten, P. (2006). Extending xp practices to support security requirements engineering. *Proceedings - International Conference on Software Engineering*, *2006-May* 11–17.

Brauch, H. G. (2011). Concepts of Security Threats, Challenges, Vulnerabilities and Risks. In H. G. Brauch, Ú. Oswald Spring, C. Mesjasz, J. Grin, P. Kameri-Mbote, B. Chourou, P. Dunay, & J. Birkmann (Eds.), *Coping with Global Environmental Change, Disasters and Security: Threats, Challenges, Vulnerabilities and Risks* (pp. 61–106). Springer Berlin Heidelberg.

Carlson, D., & Soukop, E. (2017). Why is Sprint Zero a Critical Activity. *Cross Talk*, 35–37.

Chapman, R. (2006). Correctness by Construction: A Manifesto for High Integrity Software. *ACM International Conference Proceeding Series*, *162*, 43–46.

Chaudhary, M., Chopra, A., Chaudhary, M., & Chopra, A. (2017). CMMI Overview. *CMMI for Development: Implementation Guide*, 1-7.

Cho, J. (2008). Issues and Challenges of Agile Software Development with Scrum. *Issues in Information Systems*, *9*(2), 188–195.

Conboy, K., & Fitzgerald, B. (2004). Toward a Conceptual Framework of Agile Methods. *Extreme Programming and Agile Methods - XP/Agile Universe*

2004.Lecture Notes in Computer Science, *3134*, 105–115.

Creswell, J. W. (2009). *Research design: Qualitative, quantitative, and mixed methods approaches* (3rd ed.). London: SAGE Publications, Inc.

Daneva, M., & Wang, C. (2018a, August). Security Requirements Engineering in the Agile Era: How Does it Work in Practice? *2018 IEEE 1st International Workshop on Quality Requirements in Agile Projects (QuaRAP)*.

Daneva, M., & Wang, C. (2018b, August). Security Requirements Engineering in the Agile Era: How Does it Work in Practice? *2018 IEEE 1st International Workshop on Quality Requirements in Agile Projects (QuaRAP)*.

Dingsøyr, T., & Lassenius, C. (2016). Emerging themes in agile software development: Introduction to the special section on continuous value delivery. *Information and Software Technology*, *77*, 56–60.

Easterbrook, S., Singer, J., Storey, M.-A., & Damian, D. (2008). Selecting Empirical Methods for Software Engineering Research. In F. Shull, J. Singer, & D. I. K. Sjøberg (Eds.), *Guide to advanced Empirical Software Engineering* (pp. 285–311). London: Springer.

Erdogan, G., & Per, H. (2010). *Security Testing in Agile Web Application Development - A Case Study Using the EAST*. 14–27.

Evans, K. (2008). *Testing in Scrum Projects*. London: Ross Publishing.

Finch, L. (2009). *Towards Fit for Purpose Security in High Assurance and Agile Environments*. United Kingdom: Cardiff University.

Gandomani, T. J. (2014). Agility assessment model to measure agility degree of agile software companies. *Indian Journal of Science and Technology*, *7*(7), 955–959.

Ghani, I., Azham, Z., & Jeong, S. R. (2014). Integrating software security into agile-Scrum method. *KSII Transactions on Internet and Information Systems, 8*(2), 646–663.

González-Sanabria, J. S., Morente-Molinera, J. A., & Castro-Romero, A. (2017). DeSoftIn: a methodological proposal for individual software development. *Revista Facultad de Ingenieria*, *26*(45).

Goodpasture, J. C. (2015). *Project Management the Agile Way, Second Edition: Making it Work in the Enterprise*. J. London: Ross Publishing.

Hefner, R. (1997). Lessons learned with the systems security engineering capability maturity model. *Proceedings - International Conference on Software Engineering, September*, 566–567.

Hneif, M., & Ow, S. H. (2009). Review of Agile Methodologies in Software Development 1. *International Journal of Research and Reviews in Applied Sciences*, *1*(1), 2076–2734.

In, H. P., Kim, Y., Lee, T., Moon, C., Jung, Y., & Kim, I. (2005). *A Security Risk Analysis Model for Information Systems 2 Security Risk Analysis Model*. 505–513.

Irvine, C. & Nguyen, T.D. (2010). Educating the Systems Security Engineer's Apprentice. *IEEE Security & Privacy, 8*(4), 58–61.

ISO. (2008). *ISO/IEC 21827: 2002-10-01 (e) Information technology—Systems security engineering capability maturity model (SSE-CMM)*. Washington, DC: International System Security Engineering Association.

J. C. S. Núñez, A. C. L. and P. G. R. (2020). A Preventive Secure Software Development Model for a Software Factory: A Case Study. *IEEE Access, 8*, 77653–77665.

Jakobsen, C. R., & Johnson, K. A. (2008). Mature agile with a twist of CMMI. *Agile 2008 Conference*. 10

Jakobsen, C. R., & Sutherland, J. (2009). Scrum and CMMI - Going from good to great: Are you ready-ready to be done-done? *Proceedings - 2009 Agile Conference, AGILE 2009,* 333–337.

Jamissen, H.-G. (2012). *The Challenges to the Safety Process When Using Agile Development Models*. London: Ross Publishing.

Jurjens, J. (2004). *Developing Security-Critical Applications with UMLsec A Short Walk-Through*. London: Ross Publishing.

Kagombe, G. G., Mwangi, R. W., & Wafula, J. M. (2021). Achieving Standard Software Security in Agile Developments. *ACM International Conference Proceeding Series*, 24–33.

Karim, N. S. A., Albuolayan, A., Saba, T., & Rehman, A. (2016). The practice of secure software development in SDLC: an investigation through existing model and a case study. *Security and Communication Networks*, *9*(18), 5333–5345.

Keramati, H., & Mirian-Hosseinabadi, S. H. (2008). Integrating software development security activities with agile methodologies. *AICCSA 08 - 6th IEEE/ACS International Conference on Computer Systems and Applications*, *May*, 749–754.

Koc, G., & Aydos, M. (2017, October). Trustworthy scrum: Development of secure software with scrum. *2017 International Conference on Computer Science and Engineering (UBMK).*

Kourie, D. G., & Watson, B. W. (2012). *The correctness-by-construction approach to programming*. London: springer science & Business media.

Lacerda, T. C., & Wangenheim, C. G. (2018). Systematic literature review of usability capability/maturity models. *Computer Standards & Interfaces*, *55*, 95–105.

Landoll, D. (2011). The Security Risk Assessment Handbook: A Complete Guide for Performing Security Risk Assessments, Second Edition. In *Auerbach Publications*. Retrieved from https://books.google.ae/books?hl=en&lr=&id=ek3 MBQAAQBAJ &oi=fnd&pg=PP1&dq=security+risk+assessments+%22complete+guide%22&ots=Em8yYbEuUp&sig=6exYzeiR2mtGs6xo4zonjr7QggY&redir_esc=y #v=onepage&q=security risk assessments %22complete guide%22&f=false%0Ahttp://

Larman, C., Deemer, P., Vodde, B., & Benefield, G. (2012). Scrum Primer: A Lightweight Guide to the Theory and Practice of Scrum. In *Scrum Prime* (2nd ed.). Boston: scrum alliance. Retrieved from https://scrumprimer.org/scrumprimer20.pdf

Leffingwell, D. (2007). *Scaling Software Agility - Best Practices for Large Enterprises*. New Delhi: Pearson Education.

Leffingwell, D. (2011). *Agile Software Requirements; Practices for Teams, Programs and the Enterprise* (Addison Wesley (ed.)). New Delhi: Pearson Education.

Lucassen, G., Dalpiaz, F., van der Werf, J. M. E. M., & Brinkkemper, S. (2015, August). Forging high-quality User Stories: Towards a discipline for Agile Requirements. *2015 IEEE 23rd International Requirements Engineering Conference (RE)*.

Mailloux, L. O., Grimaila, M. R., Colombi, J. M., Hodson, D. D., & Baumgartner, G. (2013). System Security Engineering for Information Systems. In *Emerging Trends in ICT Security*. Elsevier Inc. London: Ross Publishing.

Mailloux, L. O., McEvilley, M. A., Khou, S., & Pecarina, J. M. (2016). Putting the "Systems" in Security Engineering: An Examination of NIST Special Publication 800-160. *IEEE Security & Privacy*, *14*(4), 76–80.

Maqsood, H., & Bondavalli, A. (2020a). Agility of security practices and agile process models: An evaluation of cost for incorporating security in agile process models. *Proceedings of the 15th International Conference on Evaluation of Novel Approaches to Software Engineering*.

Maqsood, H., & Bondavalli, A. (2020b). Agility of security practices and agile process models: An evaluation of cost for incorporating security in agile process models. *Proceedings of the 15th International Conference on Evaluation of Novel Approaches to Software Engineering*.

Martin, R. C. (2013). *Agile software development: Principles, patterns, and practices*. London: Ross Publishing.

Matinnejad, R. (2011). Agile model driven development: An intelligent compromise. *Proceedings - 2011 9th International Conference on Software Engineering Research, Management and Applications, SERA 2011*, 197–202.

Mcdermott, J., & Fox, C. (2002). Using Abuse Case Models for Security Requirements Analysis. *15th Annual Computer Security Applications Conference (ACSAC'99)*.

McGraw, G. (2006). Software Security- Build Security In. In *addison-wesley*. London: springer.

McGraw, G., Chess, B., & Miques, S. (2011). Building security In maturity model. *2012 Faulkner Information Services, May* 1–61.

Merkow, M. (2019). *Secure, resilient, and agile software development*. Boston: CRC Press.

Mushtaq, Z., & Qureshi, M. R. J. (2012). Novel Hybrid Model: Integrating Scrum and XP. *International Journal of Information Technology and Computer Science (IJITCS)*, *6*, 39–44.

Oueslati, H., Rahman, M. M., & Othmane, L. Ben. (2015). Literature review of the challenges of developing secure software using the agile approach. *Proceedings - 10th International Conference on Availability, Reliability and Security, ARES 2015*, 540–547.

Paudel, S., Tauber, M., & Brandic, I. (2013). Security standards taxonomy for Cloud applications in Critical Infrastructure IT. *2013 8th International Conference for Internet Technology and Secured Transactions, ICITST 2013*, 645–646.

Peeters, J. (2005). Agile Security Requirements Engineering. *Symposium on Requirements Engineering for Information Security*, *12*.

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of Management Information Systems*, *24*(3), 45–77.

Pohl, C., & Hof, H.-J. (2015). Secure Scrum: Development of Secure Software with Scrum. *ArXiv Preprint ArXiv:1507. 02992*. Retrieved from http://arxiv.org/abs/1507.02992

Poppendieck, B. M., & Poppendieck, T. (2003). *Lean software development: an agile toolkit Software Development Managers*. In *Computer* (Vol. 36, Issue 8).

London: Ross Publishing.

Prakash, A. (2022). *What Is Sprint Zero? Motivation for Sprint Zero in a software project*. Retrieved from https://resources.scrumalliance.org/Article/sprint-zero

Pries-Heje, L., & Pries-Heje, J. (2011, August). Why scrum works: A case study from an agile distributed project in Denmark and India. *2011 AGILE Conference*.

Qumer, A., & Henderson-Sellers, B. (2008). An evaluation of the degree of agility in six agile methods and its applicability for method engineering. *Information and Software Technology, 50*(4), 280–295.

Qumer, Asif, & Henderson-Sellers, B. (2006a). Comparative evaluation of XP and scrum using the 4d analytical tool (4-DAT). *European and Mediterranean Conference on Information Systems (EMCIS) 2006*.

Qumer, Asif, & Henderson-Sellers, B. (2006b). Measuring agility and adoptability of agile methods: A 4-dimensional analytical tool. *The IADIS International Conference on Applied Computing 2006, January*.

Ransome, J., & Schoenfield, B. S. E. (2021a). Secure design through threat modeling. In *Building in Security at Agile Speed* (pp. 185–235). Boston: Auerbach Publications.

Ransome, J., & Schoenfield, B. S. E. (2021b). *Building in security at agile speed*. New York: CRC Press.

Rao, K. N., Naidu, G. 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.

Riadi, I., & Prayudi, Y. (2016). *A Maturity Level Framework for Measurement of Information Security Performance*. *141*(8), 1–6.

Rindell, K., Hyrynsalmi, S., & Leppänen, V. (2015). A comparison of security assurance support of agile software development methods. *ACM International Conference Proceeding Series*, *1008*(January), 61–68.

Rindell, K., Ruohonen, J., Holvitie, J., Hyrynsalmi, S., & Leppänen, V. (2021). Security in agile software development: A practitioner survey. *Information and Software Technology*, *131*, 106488.

Roopa, S., & Rani, M. (2012). Questionnaire Designing for a Survey. *The Journal of Indian Orthodontic Society*, *46*(December), 273–277.

Ross, R. O. N., & Oren, J. C. (2016). Systems Security Engineering. *NIST Special Publication*, *800*, 33.

Rover, D., Ullerich, C., Scheel, R., Wegter, J., & Whipple, C. (2014, October). Advantages of agile methodologies for software and product development in a capstone design project. *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*.

Runeson, P., Host, M., Rainer, A., & Regnell, B. (2012). *Case Study Research in Software in Software*. Retrieved from http://www.worldcat.org/title/case-study-research-in-software-engineering-guidelines-and-examples/oclc/828789615&referer=brief_results

Runeson, Per, and M. H. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, *14*(2), 131–164.

Sameen Mirza, M., & Datta, S. (2019). Strengths and weakness of traditional and agile processes - A systematic review. *Journal of Software*, *14*(5), 209–219.

Sánchez, M. C., De Gea, J. M. C., Fernández-Alemán, J. L., Garcerán, J., & Toval, A. (2020). Software vulnerabilities overview: A descriptive study. *Tsinghua Science and Technology*, *25*(2), 270–280.

Schön EM., Winter D., Escalona M.J., T. J. (2017). Key Challenges in Agile Requirements Engineering. *Lecture Notes in Business Information Processing*, *283*. Retrieved from https://doi.org/10.1007/978-3-319-57633-6_3

Schwaber, K, & Sutherland, J. (2011). *The Scrum guide*. *2*(July), 17. Retrieved from http://www.scrum.org/scrumguides/%5Cnhttp://pdf4420.psxbook.com/scrum_1868546.pdf

Schwaber, Ken, & Sutherland, J. (2017). *The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game*. London: Ross Publishing.

Shawky, D. M., & Ali, A. F. (2013). *A Practical Measure for Software Development Process Agility*. *November 2010*. London: Ross Publishing.

Shostack, A. (2014). *Threat modeling: designing for security*. John Wiley & Sons, Inc.

Sillitti, A., & Succi, G. (2005). Requirements engineering for agile methods. *Engineering and Managing Software Requirements*, 309–326.

Silverman, D. (2013). *Doing Qualitative Research: A Practical Handbook* (4th ed.). London: SAGE Publications, Inc.

Singh, N., Patel, P., & Datta, S. (2021, December). A survey on security and human-related challenges in agile software deployment. *2021 International Conference on Computational Science and Computational Intelligence*

(CSCI).

Singh, R. (2011). Impact of requirement engineering processes on software development cost. *Indian Journal of Applied Research*, *4*(5), 200–209.

Siponen, M., Baskerville, R., & Kuivalainen, T. (2005). Integrating security into agile development methods. *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, 185a--185a.

Sonia, S. A. (2011). Development of Agile Security Framework Using a Hybrid Technique for Requirements Elicitation. *Advances in Computing, Communication and Control. ICAC3 2011, Communications in Computer and Information Science*, *125*, 178–188.

SSE-CMM Project. (1999). *Systems Security Engineering Capability Maturity Model SSECMM Model Description Document*. London: Ross Publishing.

Stol, K.-J., & Fitzgerald, B. (2020). Guidelines for Conducting Software Engineering Research. In M. Felderer & G. H. Travassos (Eds.), *Contemporary Empirical Methods in Software Engineering* (pp. 27–62). London: Springer, Cham.

Sutherland, J., Jakobsen, C. R., & Johnson, K. (2008). Scrum and CMMI level 5: The magic potion for code warriors. *Proceedings of the Annual Hawaii International Conference on System Sciences*.

Sven, T., & Poller, A. (2017). *Managing Security Work in Scrum: Tensions and Challenges*. *SecSE*. London: Ross Publishing.

Taati, A., & Modiri, N. (2015). An Approach for Secure Software Development Lifecycle Based on ISO / IEC 27034. *International Journal of Information Technology (IJOCIT)*, 601–608.

Taherdoost, H. (2018). Validity and Reliability of the Research Instrument; How to Test the Validation of a Questionnaire/Survey in a Research. *SSRN Electronic Journal*, *5*(3), 28–36.

Telemaco, U., Oliveira, T., Alencar, P., & Cowan, D. (2020). A catalogue of agile smells for agility assessment. *IEEE Access*, *8*, 79239–79259.

Terlecka, K. (2012). Combining kanban and scrum -- lessons from a team of sysadmins. *2012 Agile Conference*. http://dx.doi.org/10.1109/agile.2012.20

UCEDAVÉLEZ, T., & MORANA, M. M. (2015). *Risk Centric Threat Modelling: Process for Attack Simulation and Threat Analysis*. New York: John Wiley & Sons, Inc.

Viega, J., & McGraw, G. (2011). *Building Secure Software: How to Avoid Security Problems the Right Way (paperback) (Addison-Wesley Professional Computing Series)*. New York: Addison-Wesley Professional.

Wieringa, R. J. (2014). *Design Science Methodology for Information Systems and Software Engineering*. Springer. https://doi.org/10.1007/978-3-662-43839-8

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., & Wesslén, A. (2012). *Experimentation in Software Engineering*. Berlin Heidelberg Springer.

Woody, C. (2013). Agile Security – Review of Current Research and Pilot Usage. *SEI White Paper*.

Xu, S. (2017). Empirical research methods for software engineering: Keynote address. *2017 IEEE 15th International Conference on Software Engineering Research, Management and Applications (SERA)*.

Zhang, H., Huang, X., Zhou, X., Huang, H., & Babar, M. A. (2019). Ethnographic Research in Software Engineering: A Critical Review and Checklist.

ESEC/FSE 2019: Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 659–670.

APPENDICES

Appendix I: Questionnaire

1. What is your role in the project? ________________________________
2. How long did the sprint take? _______________
3. Security engineering activities questions, indicating the level attained by each measure implemented.

For each question, answer according to the level attained indicated by the key table provided.

		0	1	3	Additional comment
	Question				
1	Were Security controls properly configured in terms of mechanisms put in place?				
2	Was security awareness and a mutual understanding of security needs reached between all applicable parties, including the customer achieved?				
3	Was everyone's responsibility as far as security is concerned clear?				
4	Were the activities effective in maintaining a security posture?				
5	Were you able to identify and characterise security impacts of risks to the system?				
6	Were you able to understand the security risk?				

7	Were you able to prioritise the risks in the PB?				
8	Were you able to build the assurance argument to satisfactory levels?				
9	Were you able to capture and monitor security requirements changes?				
10	Was the system security designed and implemented according to the understanding established?				
11	Did the solutions meet the system's security requirements?				
12	Was agility affected in any phase? Please describe the phase and how?				
13	Was assurance achieved?				

Appendix II: Initial product backlog

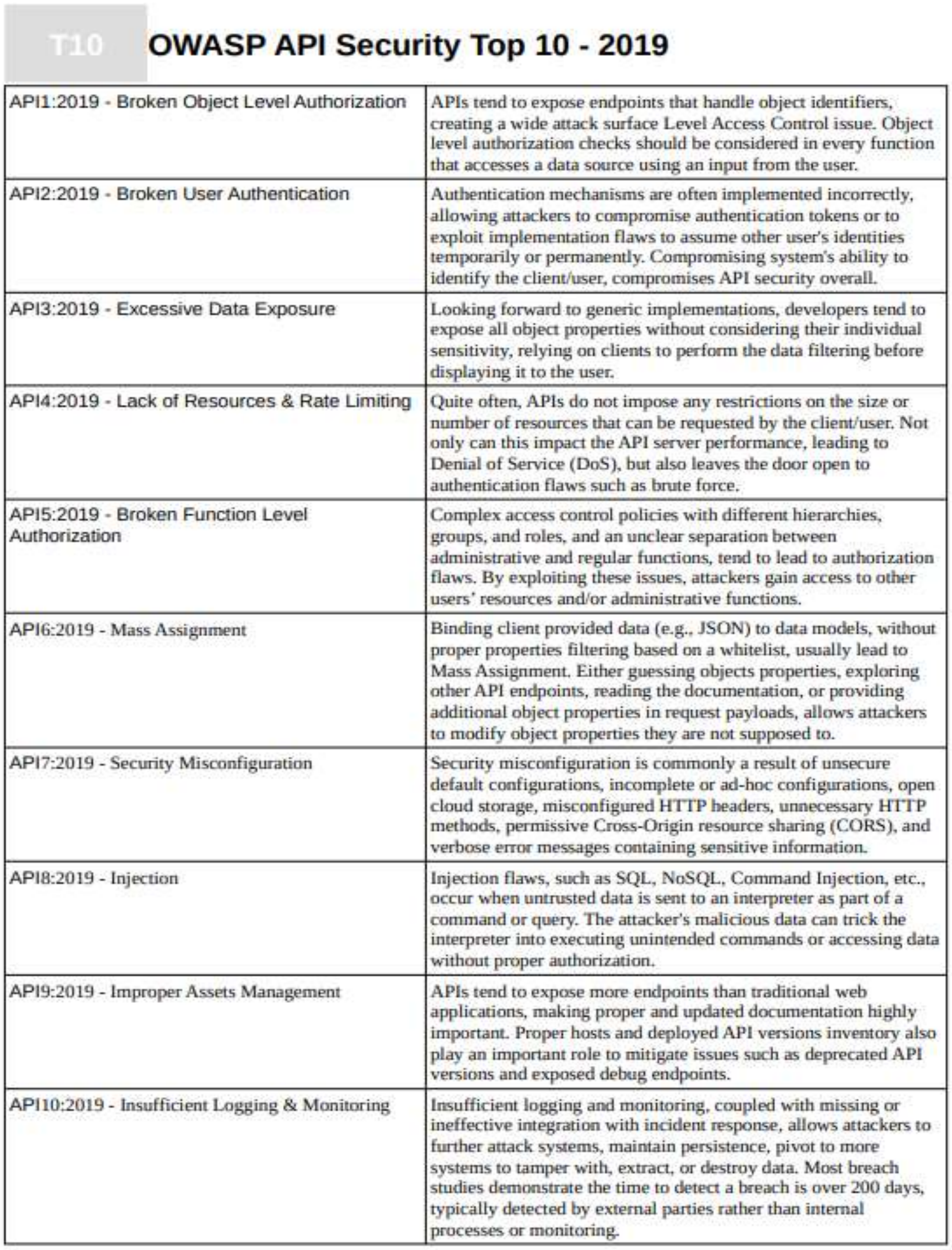
ID	User Story	Priority	Story Point
1	As an owner/manager/employee I want to login to my account so that I have access to my drive application.	A	5
2	As an owner/manager/employee I want to logout of my account so that the computer no longer has access to my drive account.	A	2
3	As an owner/manager/employee I want to access the map view so that I can view the rental car’s location in real time.	A	8
4	As an owner/manager/employee I want select a car so that I can see information about it.	A	5
5	As an owner/manager I want to add another user to a company so that he has access to the company.	A	3
7	As an owner I want to assign a tracking device to a company so that the company has access to its tracking information.	A	5
31	As an owner/manager I want to add information about a car so that I can remember information about it.	A	5
6	As an owner/manager I want to remove another user from a company so that he no longer has access to the company.	B	2

8	As an owner/manager I want to add a geofence to a company so that I could track if a rental car went to a forbidden area.	B	5
9	As an owner/manager I want to remove a geofence from a company so that I am no longer tracking if a rental car went outside the geofence.	**B**	2
10	As an owner/manager I want to create a group of rental cars so I can easily filter the cars in the group from the others.	**B**	3
12	As an owner/manager/employee I want to filter the mapview by group so that I only see the cars in the group.	B	3
19	As an owner/manager/employee I want to assign a car plate number to a rental car so that I can identify it better.	B	2
20	As an owner/manager/employee I want to see a car's information online and when I select it so that I can see more information about it.	B	5
11	As an owner/manager I want to remove a group so that I no longer see it as an available filter.	C	2
13	As an owner/manager/employee I want to see if a car is online/offline so that I can see which rental cars are in use.	C	3
14	As an owner I want to filter the mapview by company so that I can see only the rental cars in that company.	C	3

15	As an owner/manager/employee I want to see popular stops so that I can better analyse my customers' behaviour.	C	13
16	As an owner/manager/employee I want to see popular routes so that I can better analyse my customers' behaviour.	C	13
17	As an owner/manager/employee I want to see the bandwidth usage for each rental car so that I can respond if any of them are not behaving as expected.	C	5
18	As an owner/manager/employee I want to see the total bandwidth usage for the rental cars that are filtered in the map view so that I can keep track of the usage.	C	3
32	As an owner/manager/employee I want to be able to change the language of the website so that I can understand it better	C	2
21	Setup Jenkins continuous integration server for the project.	A	1
22	Integrate a unit test framework into the project so that it is run in the commit stage on Jenkins.	A	2
23	Integrate a code coverage framework into the project that runs in the commit stage on Jenkins and fails the build if a specific amount of coverage is not met.	A	2
24	Integrate a style checker framework into the project that runs in the commit stage on Jenkins and fails the build if it does not follow the coding rules.	A	3

25	Setup a process of creating acceptance tests that will run on the acceptance test stage on Jenkins Server.	A	3
26	Setup a process of creating capacity tests that will run on the capacity test stage on Jenkins Server.	A	3
27	Create a deployment script that deploys the app to a development server.	A	3
28	Integrate MongoDB into the project.	A	1
29	Integrate Here API into the project.	A	2
30	Integrate React into the project.	A	1

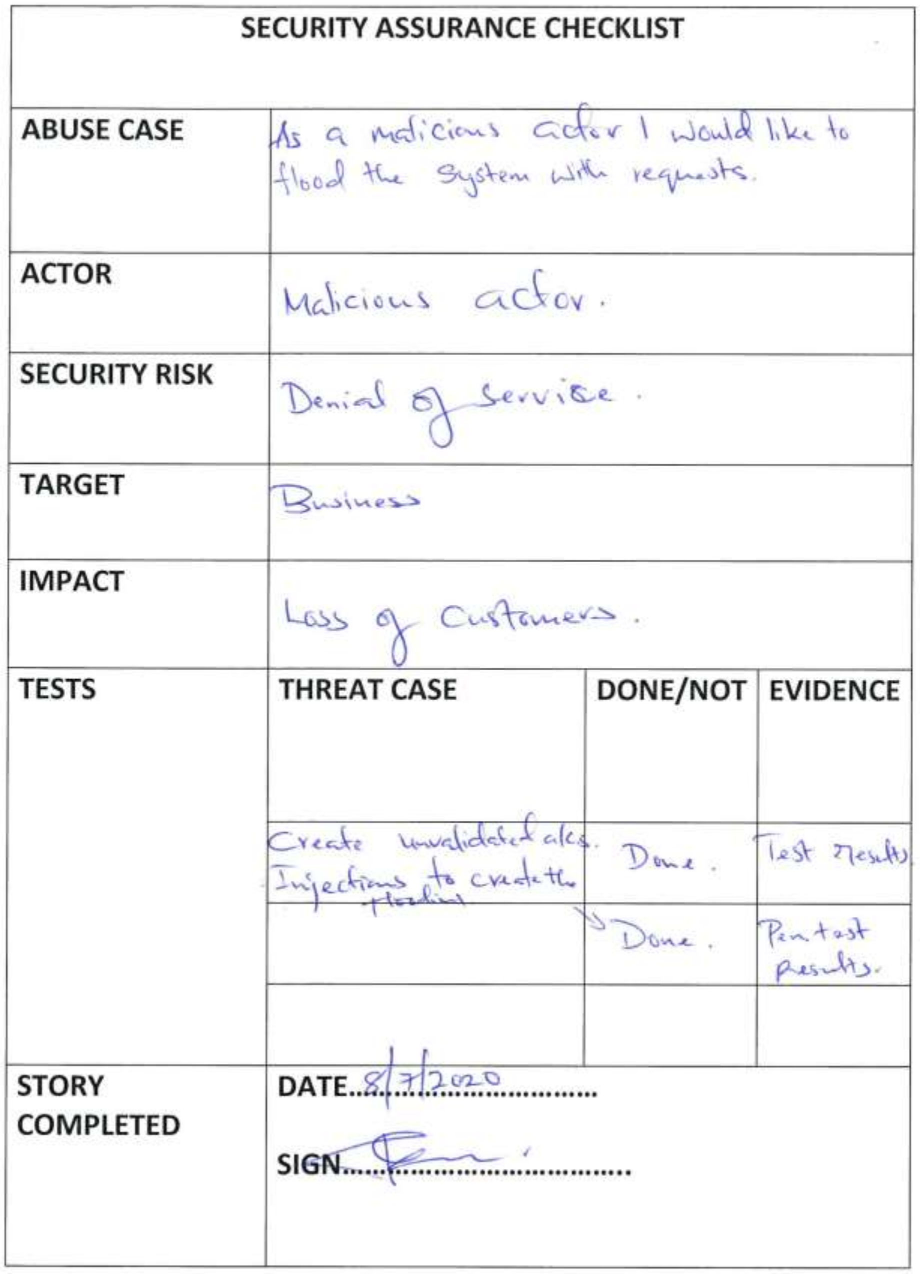
Appendix III: OWASP Top 10



T10 OWASP API Security Top 10 - 2019

API1:2019 - Broken Object Level Authorization	APIs tend to expose endpoints that handle object identifiers, creating a wide attack surface Level Access Control issue. Object level authorization checks should be considered in every function that accesses a data source using an input from the user.
API2:2019 - Broken User Authentication	Authentication mechanisms are often implemented incorrectly, allowing attackers to compromise authentication tokens or to exploit implementation flaws to assume other user's identities temporarily or permanently. Compromising system's ability to identify the client/user, compromises API security overall.
API3:2019 - Excessive Data Exposure	Looking forward to generic implementations, developers tend to expose all object properties without considering their individual sensitivity, relying on clients to perform the data filtering before displaying it to the user.
API4:2019 - Lack of Resources & Rate Limiting	Quite often, APIs do not impose any restrictions on the size or number of resources that can be requested by the client/user. Not only can this impact the API server performance, leading to Denial of Service (DoS), but also leaves the door open to authentication flaws such as brute force.
API5:2019 - Broken Function Level Authorization	Complex access control policies with different hierarchies, groups, and roles, and an unclear separation between administrative and regular functions, tend to lead to authorization flaws. By exploiting these issues, attackers gain access to other users' resources and/or administrative functions.
API6:2019 - Mass Assignment	Binding client provided data (e.g., JSON) to data models, without proper properties filtering based on a whitelist, usually lead to Mass Assignment. Either guessing objects properties, exploring other API endpoints, reading the documentation, or providing additional object properties in request payloads, allows attackers to modify object properties they are not supposed to.
API7:2019 - Security Misconfiguration	Security misconfiguration is commonly a result of unsecure default configurations, incomplete or ad-hoc configurations, open cloud storage, misconfigured HTTP headers, unnecessary HTTP methods, permissive Cross-Origin resource sharing (CORS), and verbose error messages containing sensitive information.
API8:2019 - Injection	Injection flaws, such as SQL, NoSQL, Command Injection, etc., occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's malicious data can trick the interpreter into executing unintended commands or accessing data without proper authorization.
API9:2019 - Improper Assets Management	APIs tend to expose more endpoints than traditional web applications, making proper and updated documentation highly important. Proper hosts and deployed API versions inventory also play an important role to mitigate issues such as deprecated API versions and exposed debug endpoints.
API10:2019 - Insufficient Logging & Monitoring	Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response, allows attackers to further attack systems, maintain persistence, pivot to more systems to tamper with, extract, or destroy data. Most breach studies demonstrate the time to detect a breach is over 200 days, typically detected by external parties rather than internal processes or monitoring.

Appendix IV: Security Assurance Checklist Signed for an abuse case.



SECURITY ASSURANCE CHECKLIST			
ABUSE CASE	As a malicious actor I would like to flood the system with requests.		
ACTOR	Malicious actor.		
SECURITY RISK	Denial of service.		
TARGET	Business		
IMPACT	Loss of customers.		
TESTS	THREAT CASE	DONE/NOT	EVIDENCE
	Create unvalidated alcs. Injections to create the flooding	Done.	Test results
		Done.	Pen test results.
STORY COMPLETED	DATE 8/7/2020 SIGN [illegible]		