UNIVERSITY OF SOUTHAMPTON

Faculty of Physical Sciences and Engineering

Electronics and Computer Science

The Investigation of Security Issues in Agile Methodologies

by

Ahmed Alnatheer

Thesis for the degree of Doctor of Philosophy in Computer Science

January 2014

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF PHYSICAL SCIENCES AND ENGINEERING
ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

THE INVESTIGATION OF SECURITY ISSUES IN AGILE METHODOLOGIES

by Ahmed Alnatheer

This thesis is about an empirical study on the effects of using predominant security mechanisms for integration into Agile methodologies. Claims uncovered throughout our review of literature and research are presented along with our findings, analysis, and interpretation of the qualitative and quantitative phases which underscore the gap in the literature in the past few years. In this thesis the researcher uses the issues raised in the literature and incorporates empirical findings from practitioners working in the field to form a cohesive and complete investigation into the predominant security practices that are suitable to be included into Agile. Current security issues related to and applicable to popular Agile methodologies such as Scrum and eXtreme Programming (XP) are examined along with their effects on the process and the final product are researched, quantified, analyzed, interpreted, and summarized. This is done to gain a more practical and in-depth understanding of the security issues and effectiveness of methods proposed for use in the Agile software development field today. The research considered their potential for inclusion (and possible integration) into Agile methods from multiple perspectives utilizing a mixed method approach of in-depth empirical interviews, empirical surveys, and an academic experiment to test those findings. In this manuscript we present the research along with the findings obtained with our conclusions and the future direction of the research. The contribution of this work is to identify and empirically classify outstanding issues that were agreed upon by practitioners and experts in the field. The most popular of these turned out to be the addition of the security engineer or experienced developers to the Agile team to bolster the resulting software's security assurance argument. Others aimed at modifying aspects of Agile that were deemed necessary for security include documentation, risk analysis, or the need for better tools. Building software with security in mind and the use of software security controls were also important findings from our qualitative phase of the study. This along with our own findings formed the basis of the comprehensive survey of practitioners to gauge the suitability and feasibility of those issues and solutions for possible inclusion into Agile. The significant findings from our survey suggested that the most suitable mechanisms are the addition of a dedicated Security Engineer and the use of more experienced developers to the Agile team, and the use of software security controls. Based on these results we put together an experimental trial to test the effect of more experienced developers on the Agile team on the process, the final product (which is the software produced), and the people involved (which are stakeholders in Agile projects). The statistically significant result of the experiment was in the affirmation of the hypothesis which stated that the inclusion of more experienced developer(s) to the Agile team increased the team's overall awareness of security compared to the less experienced team(s).

Table of Contents

List of figures

List of tables

Declaration of Authorship

I, *Ahmed Alnatheer*

declare that the thesis entitled:

The Investigation of Security Issues in Agile Methodologies

and the work presented in it are my own, and have been generated by me as the result of my own original research. I confirm that:

- this work was done wholly or mainly while in candidature for a research degree at this University;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed;
- where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- parts of this work have been published as:
    - Alnatheer, A., Gravell, A., Argles, D. and Gilbert, L. (2014) Agile Security Methods: an Empirical Investigation. In: *The 13th IASTED International Conference on Software Engineering*, 17 – 19 February 2014, Innsbruck, Austria.
    - Alnatheer, A., Gravell, A. and Argles, D. (2010) Major Security Issues in Agile Software Development Methodologies. In: *XPDay 2010*, London, United Kingdom.
    - Alnatheer, A., Gravell, A. and Argles, D. (2010) Agile Security Issues: an Empirical Study. At *ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 16-17 September 2010, Bolzano - Bozen, Italy.
    - Alnatheer, A., Gravell, A. and Argles, D. (2010) Agile Security Issues: a Research Study. In: *5th International Doctoral Symposium on Empirical Software Engineering (IDoESE)*, 15 September 2010, Bolzano - Bozen, Italy.

Signed: ……..

Date:……..

Acknowledgements

First and Foremost, I would like to thank both of my supervisors Dr. Andrew Gravell and Mr. Lester Gilbert for their continuous support, guidance, and feedback throughout this research. I would also like to thank Dr. David Argles for his advice, guidance, and support during the early stages of my research and wish him well after his retirement.

A special thanks goes to all practitioners in both software engineering and security communities around the globe for their participation in our empirical semi-structured interviews and questionnaire. I also would like to thank students of University of Southampton for taking the time to participate in the experimental trial.

I would like to express my thanks to my friends and colleagues in the University of Southampton, and other parts of the world for their support and being there when I needed them. In addition, I would like to thank my sponsor King Abdulaziz City for Science and Technology (KACST) for financial support.

Last but not least, I wish to thank my wife for her endless support, care, and patience despite being PhD student as well which allowed me to complete this thesis. I also wish to extend my special thanks to my mother for her support and prayers and to my family and friends for all their support and encouragement. My last grateful words are to my little son, Abdullah, for giving me love, hope, and happiness during my research work.

To my mother, my wife, my son, and
in the memory of my father

Definitions and abbreviations

ANOVA: Analysis of Variance

CASE: Computer Aided Software Engineering

CC: Common Criteria

CLASP: Comprehensive, Lightweight Application Security Process

COTS: Commercial Off-The-Shelf

DHS: Department of Homeland Security, United States Government

DoD: Department of Defense, United States Government

DSDM: Dynamics Systems Development Methods

DV: Dependent Variable

EAST: Extended Agile Security Testing Methodology

EFA: Exploratory Factor Analysis

FDD: Feature-Driven Development

IV: Independent Variable

NFR: Non-Functional Requirements

OVAL: Open Vulnerability Assessment Language

OWASP: Open Web Application Security Project

PCA: Principal Component Analysis

PFA: Principal Factor Analysis

QA: Quality Assurance

RQ: Research Question

SDL: Security Development Lifecycle

SQA: Software Quality Assurance

SSA SOAR: Software Security Assurance State Of The Art Report

SSE-CMM: System Security Engineering Capability Maturity Model

TDD: Test Driven Development

XP: eXtreme Programming

XSE: eXtreme Security Engineering

Chapter 1 Introduction

1.1 Overview of Research

There are multiple views and perspectives when it comes to security issues in software development most notably in Agile development methodologies. We present an empirical study on the effects of using predominant security issues for integration into Agile methodologies. As part of this research, current security issues related to and applicable to popular Agile methodologies such as Scrum, eXtreme Programming (XP), and Feature Driven Development (FDD) are researched, quantified, analyzed, and interpreted in order to gain a more practical and deeper understanding of the effectiveness of the security issues and methods proposed for use in Agile software development today. This research underscores the gap in the literature that has been persisting for the past few years on the addition of security into Agile and incorporates the findings from practitioners working in the field into the existing body of academic and theoretical work. Specifically, in terms of empirical evaluation, the contribution of this work is to fill the gap that was identified through the evaluation of the security issues and contributions in the past few years by attempting to identify and empirically classify outstanding issues that were agreed upon by practitioners and experts in the field. We have as part of this research developed hypotheses based upon the results of our in-depth semi-structured interviews and survey which correspond to our identified research questions that were partly derived from the literature. We have further attempted to first rank and then test the top issues based on the expert opinions of the people currently in practice and to further substantiate these findings through a wider range quantitative study aimed at gaining more widespread opinions from more practitioners in order to be able validate and generalize our hypotheses. As a result, we have come up with a way to answer significant questions in order to find out what are the most important issues and solutions that can be used in practice with a high degree of consensus and agreement from the industry. For this research we have adopted a hybrid approach whereby we take into account both the literature and the empirical data gathered though our interactions with the practitioners.

1.2 Research Objectives

The aim of this thesis is to present the research findings obtained in both qualitative and quantitative aspects of the research, to further discuss the quantitative design and the hypotheses that seek to test the principles that we uncovered, and conduct experiments that seek to test these hypotheses and finally to present the contributions and future direction of the study. The identified issues which are important factors in our research are presented along with our analysis and results of the qualitative stage.

Security experts often criticize Agile for having a fundamental lack of an inherent security mechanism to produce more secure software (Viega and McGraw 2002; Goertzel, Winograd et al. 2007) but fail to acknowledge that their approach is heavily focused on documentation and practices which hinder the Agile process in many ways if applied without modification. The current proponents of extending Agile methodologies have not yet been able to prove why the current process (XP, FDD, Scrum, etc.) cannot accommodate security practices as part of the already established methods and why an amendment or extension is required in order for this to happen. There are a number of solutions proposed by both sides of the argument which suggest a set of compromises in order to bring the two methodologies closer to integration (Beznosov and Kruchten 2004; Chivers, Paige et al. 2005; Kongsli 2006). From among this body of work, a set of ideas, issues, and questions emerged that became part of the motivation of this research to extract, analyze, and provide evidence for. Table 1-1 shows the link between the topics and the relevant papers from the literature.

Focus	Source(s)	
General Papers		
Agile and/or Security	(Abbas, Gravell et al. 2008) (Abbas, Gravell et al. 2010) (Alexander 2002) (Amoroso 1994) (Bishop 2003) (Boehm and Turner 2003a) (Boehm and Turner 2003b) (CC 1999) (Chao and Atli 2006) (Chong 2009) (Chung and Drummond 2009) (Cockburn 2002) (Dingsøyr, Dybå et al. 2008) (Doshi and Doshi 2009) (DSB 1999) (Erdogan and Baadshaug 2008)	(Gregoire, Buyens et al. 2007) (Hearn 2004) (Huo, Verner et al. 2004) (Keith 2010) (McGraw 2006) (Meneely and Williams 2010) (OWASP 2010) (OWASP 2011) (Rico 2008) (Rostaher and Hericko 2002) (Ryan Jr and Scudiere 2008) (Sindre and Opdahl 2001) (Siponen 2001) (West and Grant 2010) (Wichers 2009) (Williams, Kessler et al. 2000)
Key Theme 2: Security		
Security in Agile	(Abrams 1998) (Amey and Chapman 2003) (Baca 2010) (Berg and Ambler 2006) (Coleman 2009) (Di Lucca, Fasolino et al. 2002) (Di Lucca and Fasolino 2006) (Goertzel, Winograd et al. 2007) (Goodin 2010) (Heckman 2005)	(Hieatt and Mee 2002) (Lane 2010) (OWASP 2008) (Poppendieck and Morsicato 2002) (Sars 2010) (Pfleeger and Pfleeger 2007) (Singh 2009) (Theunissen, Kourie et al. 2003) (Vaha-sipila 2010) (Viega and McGraw 2002)
Key Theme 1: Software Engineering (Agile centered)		
Agile Security Integration/Extension *** Empirical Study**	(Aydal, Paige et al. 2006) ***** (Baskerville 2004) (Beznosov 2003) (Beznosov and Kruchten 2004) (Boström, Wäyrynen et al. 2006) ***** (Chivers, Paige et al. 2005) ***** (Erdogan, Meland et al. 2010) ***** (Ge, Paige et al. 2006) ***** (Ge, Paige et al. 2007)	(Keramati and Mirian-Hosseinabadi 2008) (Kongsli 2006) (Kongsli 2007) (Paige, Cakic et al. 2004) (Siponen, Baskerville et al. 2005) (Wäyrynen, Bodén et al. 2004) (Wichers 2008) (Williams, Meneely et al. 2010) *****

Table 1-1: Relevant papers for each topic of investigation

As part of this effort, we aim to identify outstanding security issues that we will outline in chapters 2 and 3 and discuss the methods of data gathering, analysis and

experimentation that we have developed in order to assess how useful and feasible they could be in practice. We identified a gap in the literature on the lack of empirical works on the topic which underscores the importance of having a continued discussion on this important topic which is gaining more widespread attention than in previous years by the industry in addition to the academic field. As result of our comprehensive and systematic review of literature and surrounding issues, 68 papers were looked at on the topics of Agile and/or security, 36 presented issues that were relevant to the topics of security in agile, 17 of which had issues related to integration and/or extension mechanisms but only 6 were empirical which shows the gap in the literature on this important topic. This gap is similarly illustrated in Figure 1-1 which shows the progression from the most general topics to the more specific categories leading to the gap in the literature. Dingsøyr also described this as urgently needed to be investigated empirically (Dingsøyr, Dybå et al. 2008).

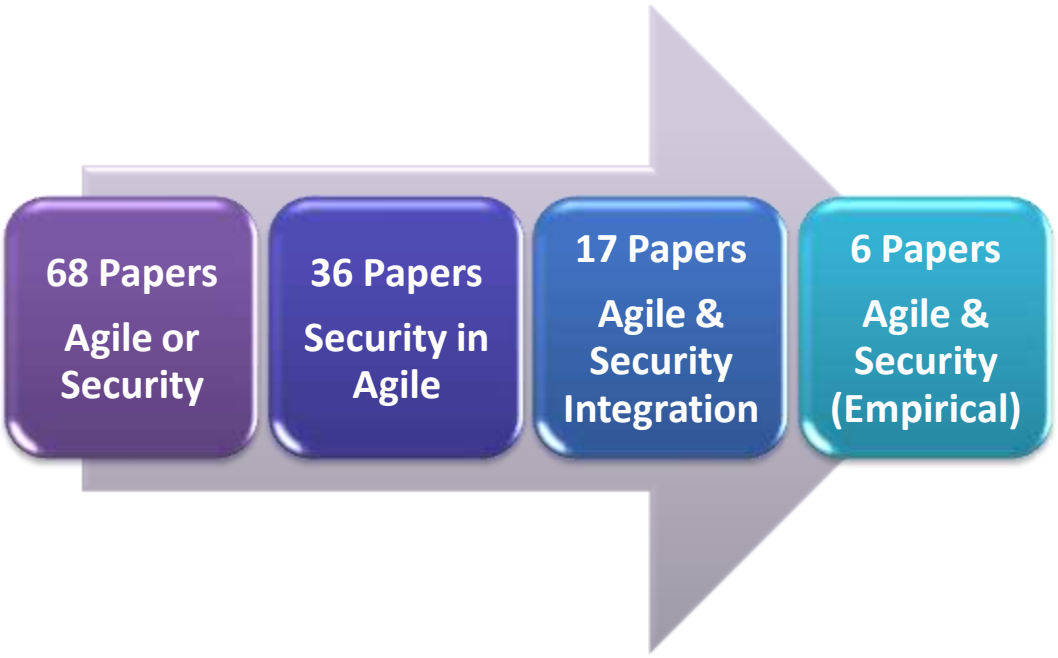


Figure 1-1 : Number of Papers identified in Literature Review

This chapter serves as an introduction to this undertaking, presents the outstanding issues that have been discovered through systematic literature review, and presents the qualitative phase of the research on the topic(s) of interest. It describes the research direction and methodologies used, provides a set of hypotheses that were tested through a comprehensive questionnaire to find broad consensus on the top two research questions that were identified through the analysis of the qualitative data. The

quantitative phase would include the analysis and results of the questionnaire followed by an experimental trial that is based on the results of the questionnaire to validate some of the hypotheses.

1.3 Research Methodologies

This research consists of a multi-phase mixed method approach of combining qualitative and quantitative techniques in order to investigate the topic of security issues in Agile methodologies from multiple perspectives and points of view. According to Kaplan and Duchon (1988) this type of study provides a more rich and rigorous approach which provides the basis for interpretation and validation of the results (Kaplan and Duchon 1988).

The mixed methods were originally introduced in the field of psychology through the work of Campbell and Fiske (1959) (Campbell and Fiske 1959). After that the interest in mixing qualitative and quantitative methods grew (Jick 1979) and finally became a distinct form of inquiry for research (Tashakkori and Teddlie 1998; Creswell and Clark 2007). There have been many different terms associated with this approach such as Integration, Synthesis, Multi-Method, Mixed methodology, and the most recent has been Mixed Methods. Multi-methods is when the researcher uses either one of Qualitative or Quantitative Methods to answer the research question while conducting empirical research. Multiple Methods, on the other hand, is being advocated more and more as the choice research methodology for empirical research (Curran and Blackburn 2001). One can use this method if they feel it can provide a more practical way to answer the research questions and be able to better evaluate the extent they can be used (Tashakkori and Teddlie 2003).

Mixed Methods consist of both Qualitative and Quantitative aspects which are done in parallel (at the same time) and/or sequentially (one after another) to answer research questions while conducting empirical research. For quantitative data collection, a number of analysis methods may be used which are different from the analysis methods used by the qualitative aspect of the research. One main point here is that the same analysis method cannot be used for Qualitative and Quantitative mechanisms at the same time and usually the use of one method predominates the other which is used as a

basis for the more elaborate phase. Using a Multi Method approach allows for greater confidence to be placed in the results obtained by the research and increases the strength of the conclusions drawn in finding answers to the research questions (Saunders, Lewis et al. 2007). Linking these two methodologies enables us to confirm and corroborate our research findings and at the same time explore the topic under discussion in much better depth (prior research plus data driven) than would otherwise be possible. Others also contend that new insights and lines of thinking could result from following this approach in investigation of a given phenomena (Rossman and Wilson 1985; Miles and Huberman 1994). The following figure illustrates where our research methodology fits within the other methodologies which is adapted from Saunders et al. (2007):

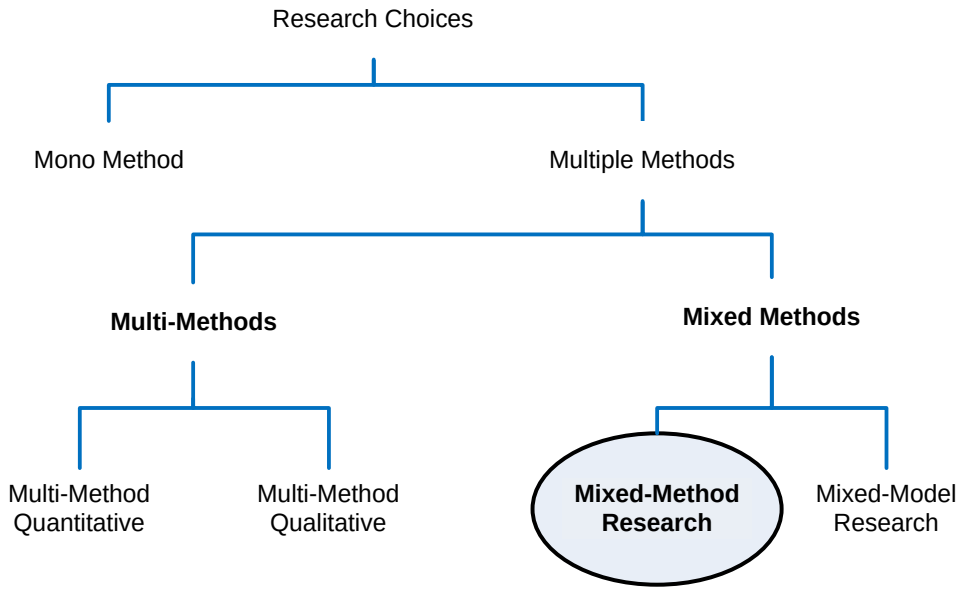


Figure 1-2 : Our choice from available research methods (Saunders, Lewis et al. 2007)

1.3.1 Qualitative Phase

According to Saunders et al. (2007), Qualitative means "Any data collection technique or data analysis procedure that generates and/or uses non-numeric data (words, pictures, video, sound)" (Saunders, Lewis et al. 2007). Data gathering methods for example could be in the form of interviews. An example of data analysis procedure includes Categorizing Data and Thematic analysis. As part of our qualitative phase, we elected to conduct semi-structured interviews followed by a number of analysis methods such as thematic analysis and frequency analysis which formed the basis for the next phase of the study.

1.3.1.1 Semi-Structured Interviews

In-depth Semi-Structured Interviews with Developers, Managers, Consultants, Security Experts, and Quality Assurance analyzed qualitatively in order to narrow down the issues into those where consensus existed between stakeholders on their effectiveness and usefulness in practice. This would influence the content of the Survey for the next phase and lead to the development of the emerging hypotheses which based on the research questions and the themes that were discovered from the semi-structured interview data. See §4.2 for design details.

1.3.2 Quantitative Phase

According to Saunders et al. (2007), Quantitative methods are defined as "Any data collection technique or data analysis procedure that generates and/or uses numeric data" (Saunders, Lewis et al. 2007). Data gathering methods for example could be in the form of a questionnaire. An example of data analysis procedure includes Factor Analysis or ANOVA that will be used depending on the characteristic of the results obtained through the quantitative phase of the research.

1.3.2.1 Questionnaire

A survey in the form of questionnaire has been distributed to a pool of 100,000 stakeholders and practitioners in the field in order to narrow down the issues further and get a more comprehensive response to issues on a larger scale. This will provide quantitative data that when analyzed using statistical methods would provide further insights into which issues would be more effective and important to consider. From the results, the top issue would be chosen for further study and experimentation which was particularly important to practitioners in the field to know what level of consensus exists on these issues. See §5.2 for design details.

1.3.2.2 Experiments

Experimental trial conducted on one or more of the top issues in order to gain more objective evidence on the use and effectiveness of the approach which was validated both through Semi-Structured Interviews and the Questionnaire. This would allow for the real world results in order to demonstrate the best ways to move forward in integrating Security and Agility. See §6.2 for design details.

1.3.1 Mixing Qualitative & Quantitative Methods

Mixing of the two phases means that at some point the data from qualitative and quantitative phases could be merged, separated from each other, or combined in some way (Creswell 2009). We are planning to keep the data separate but connected. Connected means within the qualitative and quantitative phases information are connected between the data analysis of the first method and the design and data collection of the second phase (Creswell 2009).

In our study, we have elected to use the results of the qualitative phase of the research as the basis for the questionnaire and for developing the hypotheses. We do this by doing data gathering and analysis in 2 phases. First the interview information was gathered and a thematic analysis as well as a frequency analysis was conducted and then using the results of the above analyses the questions for the quantitative study (which will be in the form of a questionnaire) designed in the quantitative phase to test our emerging hypotheses. Mixing the data might occur in several stages: Data Collection, Data analysis, and Data Interpretation. We also chose to merge the qualitative data with the quantitative information through converting themes into counts. The following figure 1-3 illustrates the overall direction of the research for the thesis.

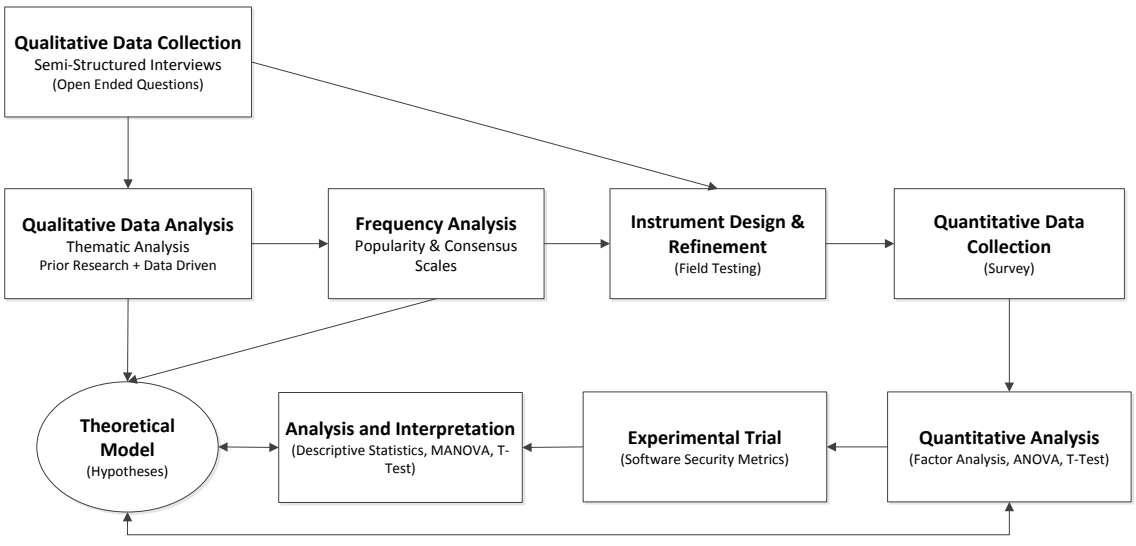


Figure 1-3: Overview of Research Methodology

1.4 The Thesis Structure

We present the issues and their related research questions in a structured fashion in chapter two aimed at uncovering useful information that provides insight into the usefulness and feasibility of each approach provided by proponents of security and agility from their point of view. This chapter discusses the background for the research and outlines efforts that have been undertaken in order to bridge the gap between security and Agile approaches.

Chapter three discusses the integration methods in terms of the rationale behind each one and discusses implications of each integration mechanism/method along with our opinions on how each are presented and could be beneficial overall.

Chapter four presents the qualitative phase of the research: design, data collection, analysis, and interpretation of the empirical data gathered through semi-structured interviews and analyzed through thematic and frequency analyses. The results of the qualitative phase led to a number of emerging hypotheses that were a result of a hybrid approach of combining the data driven or empirical results with the prior research information obtained through our comprehensive and systematic literature review.

Chapter five illustrates the quantitative stage of the research and includes the design of the questionnaire on the hypotheses derived from the result of the qualitative aspect of the research which attempts to gain a more widespread consensus on the top two research questions that were elected by practitioners as the most important to follow up and investigate further. The discussion includes the background and design of the questionnaire and also presents analysis methods and other issues surrounding the quantitative phase and ends with the top issues that would be used for the experimental trial.

Chapter six presents the experimental trial involving the hypotheses generated as a result of prior research and findings of the qualitative and quantitative methods and follows with descriptive statistics and analysis and interpretation of the findings.

Chapter seven presents the summary of the research and the future work for those wishing to continue this work.

Chapter 2 Agile Vs. Security Perspectives

2.1 Introduction

According to the "Chaos" report from the Standish Group (Standish 1995), the failure rate of software development projects is more than 50% partly because of management issues. This is because of the rigid and change resistant nature of traditional software development mechanisms such as waterfall. To make projects more flexible and successful, Agile Manifesto was created in 2001 and was later the basis for many iterative software development methodologies aimed at accomplishing what waterfall was unable to accomplish. In the past years new software engineering methods have emerged under the umbrella of Agile. They include Extreme Programming, Scrum, Test Driven Development, Lean Software Development, and Feature Driven Development amongst other methodologies. According to Forrester research (West and Grant 2010), more than 55% of the projects are utilizing Agile principles and methodologies as part of their software development efforts today. According to Beznosov and Kruchten (2004) every aspect of Agile software development methodology is affected by security (Beznosov and Kruchten 2004). They include iterations, emergence, communication patterns, and documentation practices. In this chapter we will attempt to comprehensively review the various perspectives and points of view in regards to software development methodologies. We will investigate traditional software development methodologies (such as waterfall) and security methods (such as SSE-CMM and CC) which collectively form the security focused perspective, and then present modern Agile methods to uncover all the major issues that were discussed in the literature which collectively form the Agile focused perspective.

2.2 Background

There have been numerous attempts at integrating security aspects into other methodologies that in some form or fashion employ similar ideas adopted by the Agile model. According to the Software Security Assurance State Of The Art Report (SSA SOAR), it was "the first known effort to provide a truly comprehensive snapshot of the activities of the software security assurance community, the security-enhanced software life cycle processes and methodologies they have described" (Goertzel, Winograd et al.

2007). There was another attempt at incorporating security into an evolutionary acquisition (Abrams 1998) process before but not as comprehensively as the SOAR.

The conclusion and solution to the issue of from which perspective the security should be looked at is in a compromise between agile practices and security practices (Wäyrynen, Bodén et al. 2004) in which more documentation is required and adding more steps to the process in the form of additional iteration(s) becomes warranted. In order to keep the agile benefits intact we must strictly safeguard the process and instead introduce activities that fit within the framework of agile specifically dedicated to security in order to bolster the project to provide a better security assurance argument. This includes introduction of additional iteration(s) with the specific focus on security and security assurance with the addition of a security engineer to the team that abide by all agile principles and at the same time creates security related use cases for the system.

Chivers et al. (Chivers, Paige et al. 2005) provided a supporting argument for why traditional security is not suitable for agile. They claimed the traditional security engineering processes had been built with traditional software design methodologies such as waterfall in mind and in order to bring agile to use security there should be new security engineering methods developed from the ground up to match with agile principles. Their paper claimed that agile could be integrated with security through the use of an independent Security Architecture which seemingly allows iterative security processes to be applied effectively to the project. The authors claimed that agile projects generally lacked an overall vision and that vision becomes more clear as the project progresses through iterations which results in creating an architecture but they fail to mention that in order to create effective solutions, they must already have a good enough vision on where they are and where they need to be in terms of design and development. The security architecture provides guidelines on how to judge relative security of a system through looking at the practices and the way things are done system-wide and partitioning a system into security relevant and non-security relevant parts helps with more effective design. One of the possible uses of the architecture is through summarizing parts relative to security; people can identify parts of the system that can be retrofitted possibly for security purposes through discussions and meetings. One drawback of this approach to security as a functional unit or model was in its approach to create a functional view of software, brand it as an architecture, and attribute it to a concern and an idea of security. Since security is a relative concept, it is

not tangible in and of itself although if one writes code to make the software more robust and less vulnerable, it will definitely have positive effects on the resulting solution but security implemented as a functional requirement that demands refactoring and process level changes to the code is not accurate.

2.3 Traditional Security Methods

Since long established security practices such as Common Criteria (CC) require extensive documentation to be effective (Boström, Wäyrynen et al. 2006) and since it was originally created with traditional software development methodologies such as waterfall in mind, there is a need for it to be updated and adapted to new methodologies such as Agile to be effective in today's environment(s). The reason for this is because the industry has not adopted current CC extensively due to issues regarding documentation and cost (Hearn 2004). Some have attempted to come up with alternatives to adapt CC to more areas in order to increase flexibility such as Fast Track Assessment Methodology from Information Assurance and Certification Services. According to Baskerville (2004) current security mechanisms for information and organizational security are highly resistant to change and rigid to be able to respond to rapidly changing and advancing environment of continually changing and evolving vulnerabilities and there is a need for a more Agile method of supporting security mechanisms in order to cope with the new threats and vulnerabilities (Baskerville 2004). These suggest that the traditionally established security mechanisms are no longer effective in combination with new and improved software development methodologies such as agile and need to be revised and adapted to suit new purposes.

At the same time, there have been numerous other approaches (Beznosov 2003; Paige, Cakic et al. 2004; Chivers, Paige et al. 2005) towards integrating security engineering methods with Agile practices which shows a trend towards using Agile in web applications and more (Ge, Paige et al. 2006). For example, Agile has been considered for use in Security Engineering applications (Abrams 1998; Beznosov 2003) and safety engineering applications (Poppendieck and Morsicato 2002; Amey and Chapman 2003). Baskerville (2004) described an iterative approach to secure against new and changing conditions: The OODA cycle (Observe, Orient, Decide, Act) becomes the measure of the responsiveness to sensing a change and behaving accordingly in response to that change. Over time the cycle times must be reduced to allow for more rapid change

response for a given organization to be effective against new and sophisticated adversaries (Baskerville 2004).

2.4 SSE-CMM/CC Mechanisms

Viega and McGraw argue that accelerated schedules of agile hinder security (Viega and McGraw 2002) but an alternate perspective from agile point of view is that since time has not specifically been set aside for security issues traditionally in projects, it does not mean they cannot be incorporated into agile in an effective manner. One example would be to dedicate an iteration entirely to security issues while leaving other iterations as normal functional iterations. In the security iteration, every member of the development team tries to break the functionality (already put in place so far in the project) and based upon the results of the security iteration, other iterations can follow that aim to mitigate risks discovered through the security iteration or associated with vulnerabilities found as part of testing. SSE-CMM specifies broad organizational steps that need to be taken to make the organization implementing security at multiple steps, to increase their security process's maturity.

The following table 2-1 shows how various Process Areas (PA) of SSE-CMM can be implemented at the project level by assigning a number of Agile specific practices in order to achieve better maturity for software development projects in terms of security but keep in mind that not all of these practices are required to be included, only those parts that reflect the desired level of SSE-CMM compliance.

Process Areas	Equivalent Potential Agile Practice(s)
PA01 – Administer Security Controls	Standard Security Controls
PA02 – Assess Impact	Upfront Design
PA03 – Assess Security Risk	Upfront Design (Security Risk Assessment)
PA04 – Assess Threat	Upfront Design (Misuse/Abuse Cases)
PA05 – Assess Vulnerability	Vulnerability Assessment Tools, Static/Dynamic Analysis Tools
PA06 – Build Assurance Argument	Security Engineer, More Experienced Developer(s), Awareness of Security, Static Code reviews
PA07 – Co-ordinate Security	Training, Pair Programming
PA08 – Monitor Security Posture	Security Metrics (tracking security bugs and vulnerabilities)
PA09 – Provide Security Input	Security Focused Testing
PA10 – Specify Security Needs	Reflection Meeting(s), Retrospectives Modeling Security Issues (attack trees, etc.)
PA11 – Verify and Validate Security	Security Focused Testing, Pair Programming, Security Engineer, Vulnerability Assessment Tools, Static/Dynamic Analysis Tools

Table 2-1 : SSE-CMM's Process Areas and their Equivalent Potential Agile Practices

There are not many specific PAs that directly apply to agile methodologies for software development for example more secure coding practices for internal and external deployments. However, the Common Criteria outlines steps that are more relevant to Agile from the software engineering perspective which should be taken into account more proactively. The perspective of some security engineering experts is that security cannot be added as an afterthought but needs to be built-in and specified from the start (Siponen 2001; Viega and McGraw 2002; Bishop 2003). From the Software Engineering perspective however, security is only secondary to functionality that is needed to be provided in order for the software to work. To make further enhancements in quality and security assurance, more steps could be taken to enhance the software. Some argue that solutions such as motherhood stories that some in the agile community proposed does not go far enough in making the requirements more clear but CC succeeds further in this aspect (Wäyrynen, Bodén et al. 2004). It should be noted however that even CC is not considered to be the best way to move forward. On why CC is not still adequate for software security, the SSA SOAR states: “CC STs are considered by those who are developing standards for software security assurance cases

to provide only a limited amount of meaningful security evidence for supporting security claims made in the context of software security assurance cases" (Goertzel, Winograd et al. 2007).

2.5 SSA SOAR Implications on Agile

The proponents of agile techniques and methodologies even went as far as claiming that some of the basic founding agile principles had a negative impact on security as it was presented in the SSA SOAR (Goertzel, Winograd et al. 2007). While some of the reasons given why those principles seemingly clashed with apparent security requirements, there were a certain number of assumptions used in their analysis which does not accurately represent all agile projects that are in use today which have been successful in achieving their intended purpose(s). At the same time, we acknowledge that the agile principles do need some help in achieving better security assurance by having a security engineer onboard and we will present our detailed analysis on how this could be accomplished without requiring a change in any agile practice or methodology.

We took from the SSA SOAR each principle included in the agile manifesto that was shown to have neutral or negative security implication. We chose not to go over those principles that were not shown to have negative security impact because they do not affect the security of the resulting software. Table 2-2 contains the principles and implication that appeared on the table in appendix F of the SSA SOAR (Goertzel, Winograd et al. 2007).

Principle	Implication
"The highest priority of agile developers is to satisfy the customer. This is to be achieved through early and continuous delivery of valuable software".	*"Negative, unless customer is highly security-aware. There is a particular risk that security testing will be inadequate or excluded because of 'early delivery' imperatives".*
"Agile developers welcome changing requirements, even late in the development process. Indeed, agile processes are designed to leverage change to the customer's competitive advantage".	*"Negative, unless customer is careful to assess the security impact of all new or changing requirements and include related requirements for new risk mitigations when necessary".*
"Agile projects produce frequent working software deliveries. Ideally, there will be a new delivery every few weeks or, at most, every few months. Preference is given to the shortest delivery timescale possible".	*"Negative, unless customer refuses to allow schedule imperatives to take precedence over security".*
"The project will be built around the commitment and participation of motivated individual contributors".	*"Neutral. Could be Negative when the individual contributors are either unaware of or resistant to security priorities".*
"Customers, managers, and developers must collaborate daily, throughout the development project".	*"Neutral or Positive when all participants include security stakeholders (e.g., risk managers) have security as a key objective".*
"Agile developers must have the development environment and support they need".	*"Neutral. Could be Positive when that environment is expressly intended to enhance security".*
"Developers will be trusted by both management and customers to get the job done".	*"Negative, unless developers are strongly committed and prepared to ensure security is incorporated into their process and products".*
"The most efficient and effective method of conveying information to and within a development team is through face-to-face communication".	*"Negative, as the assurance process for software is predicated on documented evidence that can be independently assessed by experts outside of the software project team".*
"The production of working software is the primary measure of success".	*"Negative, unless 'working software' is defined to mean 'software that always functions correctly and securely'".*

Table 2-2 : Core Principles of Agile with Security Implications

2.6 Modeling Security Issues

While the idea of modeling security issues is promising, so far the investigation shows attempts at introducing security into agile because of apparent lack of security assurance that is needed in some projects. Attack Trees were used as a method of implementing security in agile (Amoroso 1994). Throughout the literature, terms have

been used to refer to "abuse cases" (McGraw 2006) as a way to model security issues which is more indicative of the desired functionality required which has to do with assessing vulnerabilities. A blanket term to accomplish security tasks has also been called "modeling misuse" (Sindre and Opdahl 2001; Alexander 2002) as well.

This goes on to show there is no formalized or standardized way to refer to this set of proposals in implementing security. This further shows security practices have been making headway into research areas of agile methodologies but using different terminology and this is indicative of the fact that there is no consensus on which method or terminology is preferred and flexible enough to be included in all agile projects. This is in part the aim of our research to find the relevant and applicable security issues that can be practically used in agile and find to what degree consensus exists on different issues that are uncovered as part of the process of systematic literature review.

Even though many think that the traditional waterfall mechanisms are best at accomplishing security, according to SSA SOAR "Given that a risk-centric approach is needed for security engineering, a spiral systems development model, which is a risk-driven approach for development of products or systems, is ideal" (Goertzel, Winograd et al. 2007) which shows that Agile might be more appropriate than waterfall for these purposes. On the same note, one way for the security engineer to accomplish vulnerability assessment is to model misuse and conduct analysis on the model:

> *"Fault tree analysis for security (sometimes referred to as threat tree or attack tree analysis) is a top-down approach to identifying vulnerabilities. In a fault tree, the attacker's goal is placed at the top of the tree. Then, the analyst documents possible alternatives for achieving that attacker goal. For each alternative, the analyst may recursively add precursor alternatives for achieving the subgoals that compose the main attacker goal. This process is repeated for each attacker goal. By examining the lowest level nodes of the resulting attack tree, the analyst can then identify all possible techniques for violating the system's security; preventions for these techniques could then be specified as security requirements for the system"* (Goertzel, Winograd et al. 2007).

2.7 Lack of Agile Security

Historically, Agile has been cited as lacking proper security facilities (Viega and McGraw 2002) because of an apparent mismatch between how the processes are accomplished in security mechanisms versus Agile. So far our investigation shows that attempts at introducing security into Agile were initiated because of the apparent lack of security that might be needed in a number of more critical and sensitive projects. As an example to integrate security into agile methodologies Kongsli presented activities to take security into account as an initial step to introduce security mechanisms into agile practices (Kongsli 2006). The main issue here however is that there is a fundamental problem in comparing traditional security techniques with their inclusion into the modern Agile practices. Since the security methodologies were created with longer schedules and more formality in mind, Agile opted for less formal activities and accelerated schedules. Therefore it is not fair to directly compare their inclusion into the Agile model without modifications to make them more Agile friendly. Once modifications are in place comparison between the new combined practice to the old Waterfall model plus security can be attempted to assess the true potential of secure agile practice.

2.7.1 Documentation for Security

The security engineering community has voiced concern over apparent lack of documentation generation activities in Agile for security critical projects as it relates to design documentation and detailed interface specifications (Viega and McGraw 2002). According to Kongsli (2006), Agile methodologies hold opposing views to the traditional security practices such as the process of security reviews. In order to implement the same level of security as in the traditional methods, the Agile process needs to extend the documentation facilities beyond what was called for in the manifesto (Appendix A) (Kongsli 2006). The community also claimed that there is no "proof of compliance" with accepted software engineering standards and security engineering standards such as Security System Engineering-Capability Maturity Model (SSE-CMM) and CC (Theunissen, Kourie et al. 2003). Of course, the contribution of the research was to better understand Agile methodology's suitability for security focused software development but they failed to acknowledge that the SSE-CMM and CC were never meant to be used for accelerated timeframes and schedules which Agile

is created for. A better approach would be to explore security issues from the agile perspective rather than the other way around which is biased towards documentation and early planning to accomplish security.

Security assurance increases confidence in the solution at best but does not ensure or guarantee that security issues will not occur in the software. But what it does mean is that it will reduce the chance of certain types of security issues to cause damage and hinder operations for the company or organization (Wäyrynen, Bodén et al. 2004). Furthermore, this analysis was only undertaken for Extreme Programming (XP), and left other popular methods such as Scrum and Feature Driven Development (FDD) untouched. As part of this study, we are going to use Scrum as well as XP and FDD as the representative agile models because their adoption rate in the industry is over 50% of all Agile projects (West and Grant 2010).

2.7.2 Verification Tools for Secure Code

Another major issue is whether or not the testing and verification tools used by agile teams provide adequate security assurance for the software in addition to quality assurance. In research conducted by Kongsli (2006), the tools were found to lack certain security level assurances to be addressed when it came to testing for security issues because sometimes the tests went beyond application level vulnerabilities and had to do with the deployment environment rather than the specific software that was being written. This was an attempt at bringing more security awareness and accountability to the team that employed agile as their development methodology. They found while the practices were worthwhile, they were not complete or satisfactory in achieving good enough security. This underscores the need for better tools which provide additional flexibility in the face of security related requirements in order to make a stronger security assurance argument. On the same note, we will create specific questions that attempt to understand this from different points of view and also ask opinions on alternative approaches.

2.8 Agile Security Requirements

In Agile, user stories are often indexed by the customer to get the development started. In order to form a complete vision of the project, it would be helpful to prioritize features or user stories ahead of time and then choose to implement the top X number of higher-priority features. Developers should have equal say in the prioritization along with the customer to make sure some features that need to be completed early on are put in place. This establishes a high-level structure for the project and allows for people to know where they are and understand the big picture which agile projects do not currently emphasize (Chivers, Paige et al. 2005). The following figure 2-1 illustrates how the traditional practices of waterfall are being accomplished through different practices with similar outcomes but in a more Agile fashion. Here it is acknowledged that having Misuse or Abuser stories plus some upfront Design and release planning does help increase security of the resulting software. The additional steps for Agile include the addition of Abuser Stories, Upfront Design, and Security Unit tests along with Regression Testing in order to provide a similar process than how things are done in waterfall while at the same time keeping the core Agile practices largely intact.

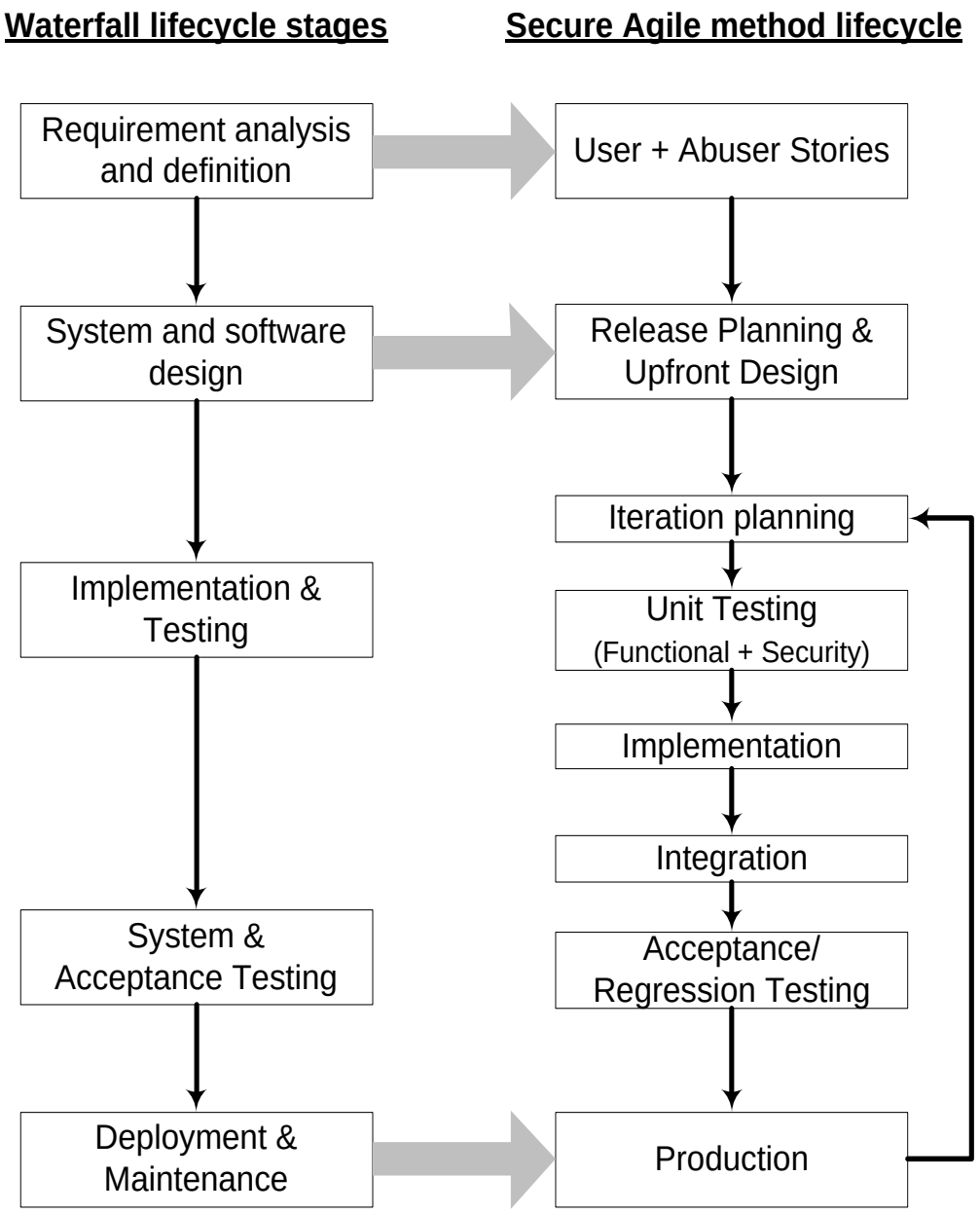


Figure 2-1 : Parallels between Traditional Waterfall and Secure Agile

2.9 Abuser/Misuse Stories

Siponen et al. (2005) described a way to introduce security into agile seamlessly through an application of security focused activities in a project (Siponen, Baskerville et al. 2005). They claim that traditional security methods need to be modified and become more Agile in order to be able to fit into the Agile process. They cite as a requirement

retrofitting security methods and methodologies to fit the new model of Agile in order to be able to integrate those mechanisms into the new Agile methods in practice. Their approach calls for creating one or more abuse cases that may or may not correspond with a feature to be implemented by the system. According to Bostrom, Abuser stories are hailed as the means to go about creating security related requirements for the project (Boström, Wäyrynen et al. 2006). Abuse cases are created and organized based on apparent measure of loss in N USD and the risk scale from 1-10. The idea was that the riskier abuse cases should be handled and tested first. Figure 2-2 shows a sample of the abuse case for a given feature of the FDD process. Even though abuser stories are considered as the means to go about creating security related requirements for the project, and since the abuser stories - though described specifically rather than in generic or general terms - might not be directly translatable to code, they may not be suitable to be added to the project or turned into functional requirements. This is because the purpose of use cases or user stories is to understand the process that one needs to simulate that is directly translatable to code. Also, how can tests be written for stories that may not even be in the same level or domain as the domain of problems that are being solved as part of the project. The above research assumes that the abuse cases are easily translatable to code but this needs to be looked into further empirically before it could be substantiated.

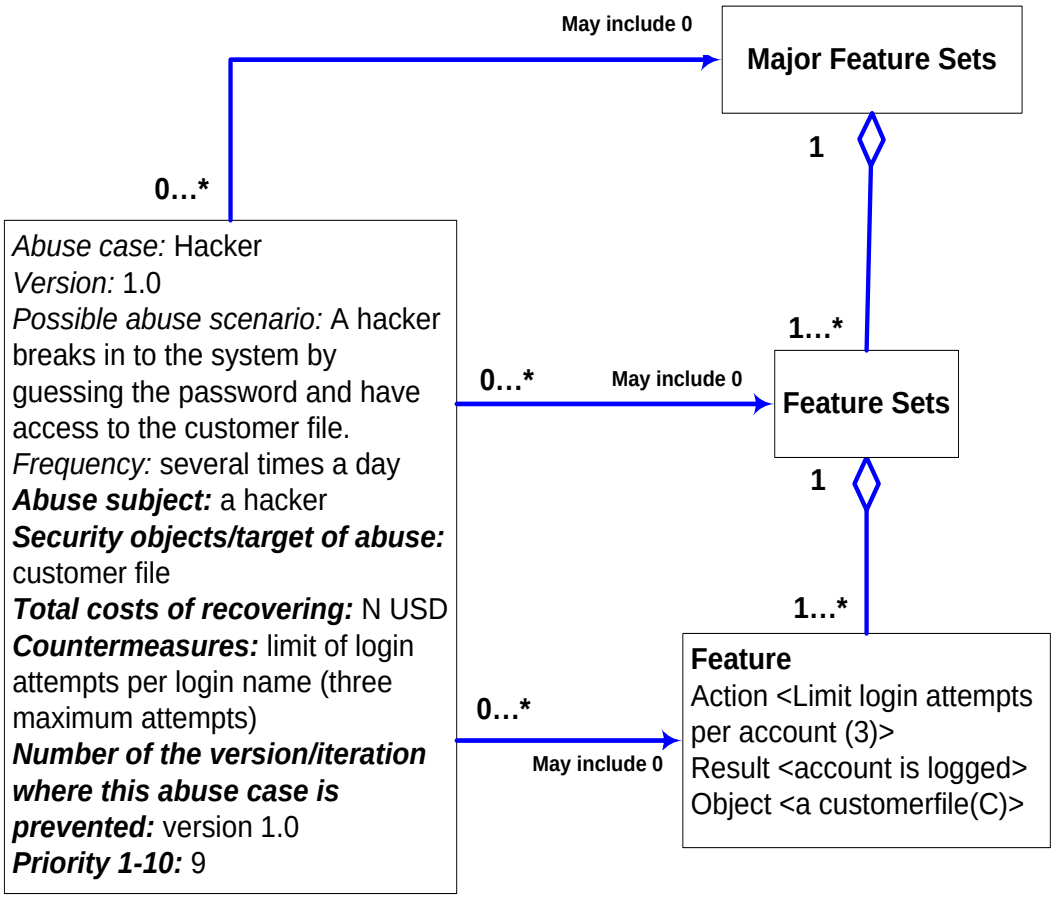


Figure 2-2 : Abuser Stories Integration into FDD (Siponen, Baskerville et al. 2005)

There are some pieces of functionality that are expected from every application today and they are many times security related. Therefore, deferring their implementation to the end of the project in the name of security could add significant development and refactoring time when all of the functionality could have been incorporated from the beginning. While we acknowledge that there are overriding security concerns that need to be addressed within the framework of the project, we maintain, they need to be carefully examined to make sure they are not already considered as required and we intend to find evidence for or against this. Another solution proposed the use of Misuse or "Abuser Stories" in Extreme Programming to try to put emphasis on security related features and/or functionality of the system being developed and to be able to test for those specific threats and vulnerabilities in order to be able to perform security regression testing (Siponen, Baskerville et al. 2005; Boström, Wäyrynen et al. 2006; Kongsli 2006). A method was introduced to integrate security into Agile methodology and presented activities to take security into account as

an initial step to introduce security mechanisms into Agile practices. According to Kongsli, writing security test cases to address vulnerabilities allows for more formalized security requirements to be taken care of within the project and add to the overall security of the software (Kongsli 2006). The term "misuse stories" is used to refer to undesired functionalities of the system that needs to be taken care of as part of the security mechanism. They used misuse stories as part of the requirement analysis phase of the iteration in order to add a layer of security to the process. An important fact to understand that many times the security requirements modify and/or restrict access to certain areas of the software and this kind of behavior is not considered normal development, therefore security is said to be a collection mostly Non-Functional Requirements (NFR).

It turns out that sometimes the misuse stories may or may not have a large threat level to the system and treating them as an acceptance scenario might prove to be too much for the project to handle in terms of time and cost (Kongsli 2006). In other words, the security issues need to be prioritized in terms of threat level and risk and voted on by stakeholders to implement or not in order to keep costs to a reasonable level. This is a major issue that needs further analysis and discussion to gain a more comprehensive understanding of all the variables involved in the process. Another paper by Kongsli (2007) focuses on an automated testing tool named Selenium where it was seemingly adapted to make it more suitable to create and run automated security tests (security focused unit tests run by the testing framework automatically for each release) for web applications (Kongsli 2007). This was done by using misuse stories to exploit vulnerabilities through applying the undesired inputs on the web application. Using this tool, the author contends that security testing provides basis for the final acceptance test suite and helps to make a security assurance argument for the application. Kongsli claimed that this tool could possibly find certain common vulnerabilities in web based applications for example improper error handling, cross site scripting (XSS), information leakage, broken authentication, and access control. He also claimed that such a testing tool would reduce the gap between regular developers and security engineers because the same tool used for functional testing can be also used to create security related tests. It also could enhance collective ownership over the project. Depending on only one testing tool, such as Selenium might create an inflated level of confidence to developers towards their application. Therefore, having a number of tools

and/or other different methods to assess and achieve better security is preferable. No matter what, software may very well be vulnerable to all unknown security threats, and there seems to be no consensus on how to best provide an approximation for the level of security provided by the addition of testing methodologies that are aimed at uncovering flaws. Selenium is claimed to be able to create tests for such cases but Kongsli could not show it was even possible to find with certainty, most known vulnerabilities. This underscores the need for good metrics to be able to measure the relative security of the software that is being written.

The countermeasures described for handling abuse cases clearly shows how the abuse cases could be handled very well by other means than from within the software such as tools and 3rd party utilities. Therefore spending extra time and effort into misuse and abuse cases and going through all the extra analysis and documentation within the Agile framework may not benefit the project in a meaningful way and might end up adding more documentation and other activities to the project which have been deemed unnecessary in the first place by the Agile manifesto. A number of authors claimed to have found an Agile security mechanism but in their description they still did not provide an agile method to handle security related issues. For example, they presented an access control mechanism that could be handled in many different ways and made it part of every Agile iteration to have to discuss and think of abuse cases even though there might not be a representative and specific threat at the project level that needs software written for it in order to mitigate the issue.

2.10 The Security Engineer

According to Pfleeger et al. “Unfortunately, many developers do not have the opportunity to become sensitive to security issues, which probably accounts for many of the unintentional security faults in today's programs” (Pfleeger and Pfleeger 2007).

2.10.1 Education and Awareness

Ge et al. introduced the idea of security training for all stakeholders in 2007 to make people more aware of security related issues. Amongst things that could come up during the course of the project, having an understanding of known security threats and vulnerabilities helps in achieving better quality software that can be assured through

security testing (Ge, Paige et al. 2007). Coincidentally, the inclusion of security engineer accomplishes the same task among other things. Ge et al. introduced the notion of a fundamental security architecture as a set of best practices and principles that allow a system architect or security engineer to apply security related practices across projects. They mentioned that a common result of lack of security awareness is increased risk and vulnerability of the resulting software. They introduced risk management into software development methodologies to improve security functionality and code quality and claimed that the notion of security architecture is in line with XP's established values or requirements for the given software development project. They also recommended the use of a security expert within the team that acts as the coach for other members to teach them about security issues and the security engineer fits this task nicely. They also cite the use of practical security manuals and best practices as a basis for risk management within and across projects to achieve better overall security. Table 2-3 shows various levels of developer capabilities which Boehm revised from the original Cockburn rating (Boehm and Turner 2003a; Boehm and Turner 2003b).

Level	Characteristics
3	*"Able to revise a method, breaking its rules to fit an unprecedented new situation".*
2	*"Able to tailor a method to fit a precedented new situation".*
1A	*"With training, able to perform discretionary method steps such as sizing stories to fit increments, composing patterns, compound refactoring, or complex COTS integration. With experience, can become Level 2".*
1B	*"With training, able to perform procedural method steps such as coding a simple method, simple refactoring, following coding standards and CM procedures, or running tests. With experience, can master some Level 1A skills".*
-1	*"May have technical skills, but unable or unwilling to collaborate or follow shared methods".*

Table 2-3 : Revised Cockburn's Developer's Capabilities (Boehm and Turner 2003a; Boehm and Turner 2003b)

Other approaches also been around for a while that are more commercial in nature such as Development Lifecycle (SDL) that focuses on Tractability, and Comprehensive, Lightweight Application Security Process (CLASP) which provides a set of activities to establish and use metrics throughout the software development lifecycle. Both methodologies are focused on Waterfall methods but they have been adapted to include Agile as well. SDL also provides a set of tools to help with the management and support tasks that an organization needs to go through but CLASP focuses on establishing and implementing a set of activities to accomplish the same. Both methods have support for education and awareness for security purposes; SDL puts emphasis on developers while CLASP recommends that training should include all project roles (Gregoire, Buyens et al. 2007).

From the above discussions from the literature and other areas of practice that indeed regular developers may not be suitable for handling security requirements. However, more experienced level 1A and above developers could be trained for security to be able to handle security requirements. Such mix of experience and security knowledge will result in an effective role of being a "Security Engineer" who can not only develop software, but also recognize and mitigate risks and solve security issues.

2.11 Providing Security Assurance

Agile specific activities such as automatic testing and pair programming (especially if one of the team members is a dedicated Security Engineer or expert) could be used to test for security specific cases to provide security assurance argument for the project and satisfy this aspect of the overall security requirement (Wäyrynen, Bodén et al. 2004). Furthermore, the existence of the Security Engineer could help spread the tacit knowledge of security rather than having to do the same through documentation with the same outcome. According to "Common Criteria for Information Technology Security Evaluation" (CC 1999), the assessment of vulnerabilities could be counted towards a security related activity within the agile methodology. Therefore, it could be a basis for an overall security assurance argument. To go one step further, a three iteration security solution could be used which involves dedicated iterations for vulnerability assessment (to find all vulnerabilities), risk assessment (to decide which vulnerabilities

need to be fixed), and risk mitigation (to fix the vulnerabilities) to act as a multi-step approach towards overall project level security assurance.

One measure of security capabilities of developers could be the number of vulnerabilities that they introduce into the system and to mitigate this risk the team member with the lowest security score would be paired with the Security Engineer to help him improve security of the software. This is an improvement to overall security of the Agile in utilizing pair programming to provide an additional security argument for better security assurance. On the same note, an answer to the assertion that Agile is not good for security because of lack of formal documentation would be to say that with the advent of Computer Aided Software Engineering (CASE) tools (that aid in better documenting the software and its functionality) the argument against agile is weakened more and more everyday as the tools and technologies become more sophisticated and mature.

2.12 Risk Based Approaches

There were a number of approaches that utilized risk assessment as part of their security mitigation strategy but none have put a singular emphasis on risk alone to accomplish security especially in Agile methodologies. According to Boehm, the COTS utilization also adds a level of uncertainty that has its associated risks that it imposes on the software being written. To mitigate this, there must be a watchful eye in assessing emerging technological risks (Boehm and Turner 2003a). Using a COTS package increases inherent risk especially in Agile projects because of the less time spent on investigation and planning to ensure the COTS package provides the necessary security and quality assurance for use in the project.

2.13 Summary

In this chapter we attempted to describe various perspectives and points of view of researchers and practitioners from the literature on how to best provide security assurance in Agile. First we presented issues more relevant to the security perspective and then followed up with more Agile focused approaches. According to SOAR (2007) **"**It could be very interesting to take a similar snapshot 10 years from now, when

software security assurance as a discipline reaches the level of maturity that software quality and safety assurance are at today" (Goertzel, Winograd et al. 2007). The previous quote from SSA SOAR sums up this chapter quite appropriately in a sense that the solutions and mitigation strategies currently advanced as part of the literature are not mature enough to address the security assurance of the software produced by Agile. In the next chapter we will expand our investigation into areas where security and Agility could be integrated.

Chapter 3 Security & Agility: Integration Methods

3.1 Introduction

It has been proposed to amend the agile process to include security related practices (Aydal, Paige et al. 2006; Boström, Wäyrynen et al. 2006; Kongsli 2006) and discussion at each iteration that extends the planning process (in XP) by a number of stages and more. Given the fact that security is an overriding issue which in most cases may not directly impact functionality, which is the purpose of the agile project to begin with, the focus of the planning should not be skewed in favor of security. Rather, if security is required to be implemented, one or more dedicated iterations with their own security related topics (or themes in the case of Scrum) and use cases can be introduced to take care of some of the issues without involving too much time and resources. The implication of the above issue is that there needs to be more concrete and objective evidence in support of the claim that the addition of security should not hinder or lengthen the agile process which makes sense given the fact that some projects do not need security and simply adding a security step at each iteration would not be beneficial in those cases. There is no proposal at present that attempts to produce a structured approach to analyze XP from a strict security engineering perspective and Wayrynen (2004) called for more investigation into the issue (Wäyrynen, Bodén et al. 2004).

3.2 Security Issues

3.2.1 Security and Resource Utilization

There is little to no evidence that adding security to agile projects contributes significantly to finished projects and furthermore the effort involved is costly (Di Lucca, Fasolino et al. 2002; Hieatt and Mee 2002; Di Lucca and Fasolino 2006). Implementing acceptance tests or unit tests or security testing in general sometimes requires more resources than is available to developers as part of the Agile project. In other words, the testing requirements go above and beyond normal development practices because they may involve the system as a whole on multiple levels and often cannot be accomplished by automatic testing at the project level alone. This is a known hurdle in implementing

security mechanisms in general which shows Agile alone cannot be responsible for solving security issues for a given piece of software in development. The following figure 3-1 shows the scope of applicability of Agile compared to Security which clearly shows how security issues go above and beyond what Agile is capable of.

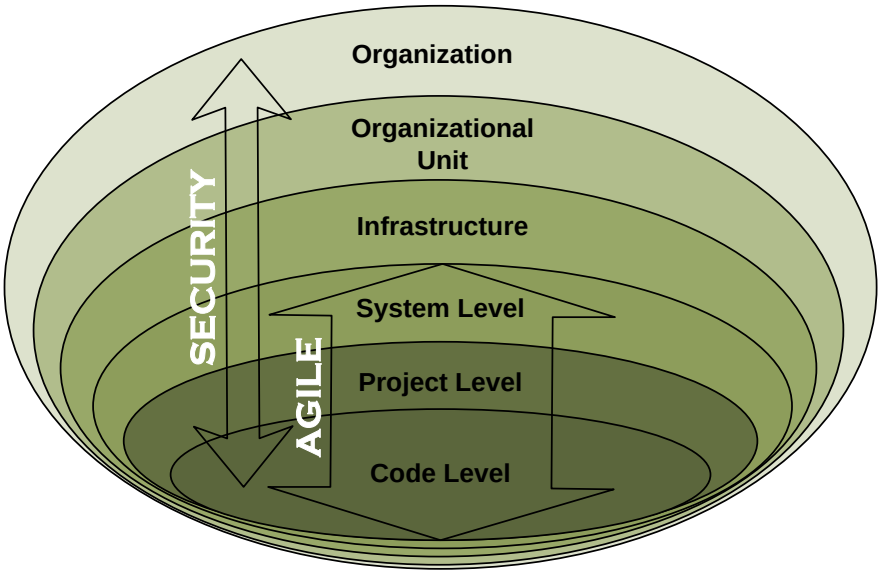


Figure 3-1 : The Scope of Agile vs. Security at each layer within the organization

This has a number of implications that needs to be considered while designing and developing secure software using Agile and also since security goes above and beyond the functional requirements, the customer needs to have a say in whether or not security is desired for the project. We aim to investigate to what degree this is needed. Even though total and absolute security in software is highly desirable and attractive to stakeholders involved, the additional increase in costs associated with such undertaking is the biggest barrier to its realization. For example, a study conducted for DoD in 1999 (DSB 1999) concluded that even though the utilization of COTS contributed to increased risk to the department operations and software systems, the department was limited in funds to be able to create their own comparable, more secure solution which clearly shows the additional costs that the additional security assurance incurs might very well be prohibitive in creating the desired functionality in the first place. With this in mind, the best that can be feasibly achieved in terms of security is to take steps to

safeguard against the most popular and widely used attacks which represent significant risks in order to decrease the software's likelihood of exploitation.

3.3 Agile Issues

3.3.1 Security and Quality Assurance

A recurring theme within the software security landscape is that of increasing security through reducing vulnerabilities which is directly related to software quality. Realization of this fact leads us to conclude that software quality and security are inevitably related and any increased quality would have a measurable impact on the software's security as well. Therefore Agile techniques that increase software quality through various techniques also contribute to a higher security assurance by producing less defects and better quality code. In Agile training for NFRs is key to understanding security issues. Having an independent testing individual for every team for QA is indicative of a mature Agile process implementation that needs to be an expert in NFRs as well as security issues so they can help the teams with these overriding issues.

In Agile in order to be able to figure out what security requirements should be, the developers must have at-least basic knowledge and understanding (from level 1A and above from table 2-3) of security issues to be able to come up with useful misuse/abuse cases or threat scenarios. While customers understand the business risk associated with the overall functionality of the software product, they are not well suited to discuss any specific instance within the program or its functionality on how to avoid or mitigate such risks. After this it is solely upon the Agile team and their associated security structure to provide adequate security assurance to the customer and present such a scenario for customer approval on its implementation. In order for the customer not to form a negative view of the resulting software, great care must be taken to present security issues in a positive way to promote a strong security assurance rather than scaring them into canceling the project because added cost to secure seemingly high-risk issues might be prohibitive (OWASP 2008; Wichers 2008).

Huo et al. found that the lack of documentation in Agile teams contributed to a growing sense of confusion for third party QA that often conducted static QA techniques (Huo, Verner et al. 2004). Their research specified how QA techniques were used within Agile and how they produced the same or better quality than waterfall or traditional techniques with the added benefit of shorter iterations and changing

requirements. The authors contend that some of the work that was traditionally recognized as distinct QA processes are now assigned to developers. It was found that instead of the elaborate and document heavy architectural specification in the waterfall method, Agile took the overall system as a kind of a vision (Chivers, Paige et al. 2005) and metaphor in describing an idea of the architecture since it is not yet matured into a full clear picture. In this sense Agile projects are always works in progress while waterfall presents a blueprint for a completed system. Refactoring in Agile helps QA by keeping and ensuring clear and clean code is maintained throughout the development process and any and all variables, values, and patterns that are used could be changed and refactored accordingly. Huo et al. claimed that the effect of pair programming is the same as having a code review session all the time which increases software quality, security, and decreases defect rates but this claim needs to corroborated by more researchers in order to be substantiated further. It was also found that continuous integration and acceptance testing acts as dynamic QA does later on in the waterfall model and helps with finding integration problems and compatibility problems earlier than before. The following figure 3-2 (adapted from Huo) compares a typical Agile lifecycle with its traditional counterpart in terms of QA techniques employed at each step:

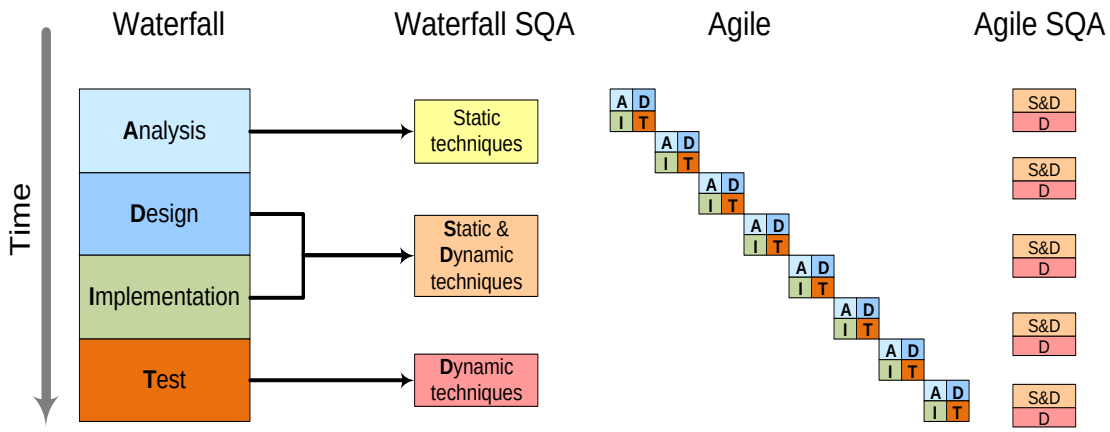


Figure 3-2 : Waterfall vs. Agile in terms of Quality Assurance (Huo, Verner et al. 2004)

The conclusion is that agile QA techniques have fewer static but more frequent dynamic QA methods than the waterfall model but at the same time provide better

quality and security assurance but to what extent they provide additional security assurance is an open question which we intend to find answers for.

3.4 Clashes between traditional security and Agile

From the security engineering perspective, research suggests that there is a mismatch or disparity (Viega and McGraw 2002) between established security methodologies and the ideas and methodologies proposed by the agile advocates (Beznosov and Kruchten 2004). Beznosov and Kruchten attempted to see whether or not agile methodologies fit well with security assurance practices. They found that over 50% of the processes can comfortably coexist with existing agile methodologies and found the other 50% could be tailored to work alongside Agile practices to a large degree (Beznosov and Kruchten 2004). As part of their research, they classified a number of security assurance practices from various stages of the software development lifecycle and determined which ones could be adapted relatively easily and which practices were openly clashing with Agile. The following table is adapted from their classification on these aspects (Beznosov and Kruchten 2004):

Security assurance method or technique		semi-Automated	Mismatch
Requirements	Specification analysis		X
	Review		X
Design	Formal validation		X
	Internal validation		X
	External review		X
Implementation	Informal correspondence analysis		X
	Requirements testing	X	
	Informal validation		X
	Formal validation		X
	Security testing	X	
	Vulnerability and penetration testing	X	
	Test depth analysis		X
	Security static analysis	X	
	Change authorization		X
	External review		X
	Security evalution		X

Table 3-1: Security Assurance Practices Classification (Beznosov and Kruchten 2004)

Agile methodology created a barrier between the development team and the 3rd party security assurance in how knowledge is relayed to the 3rd party team. Since the Agile methodology relies on tacit knowledge to communicate requirements and other related issues the lack of formal documentation contributes to a greater gap between the understanding of the 3rd party security and the in-house development team. Since the developers and security assurance people do not meet eye to eye, there could be shortcomings from the lack of security awareness on the software. Perhaps an in-house security engineer can be present so the developers understand all the related issues and details as they occur to make the process more effective. On the other hand, according to Wichers there's no clear clash between Agile and security because aspects of Agile can be used towards improving security of software. It's the speed of development that causes some to think the software is not adequately tested or created with security built-in (OWASP 2008; Wichers 2008). The conclusion drawn on the conflicting areas of

security assurance with Agile methodology mitigated through the use of more automated testing and adaptation of unit tests for security purposes and educating the developers on specific security issues and security-oriented testing philosophy. At the same time, this incurs a lot of cost to train staff specifically for security when one security expert can be hired for the team to take care of security issues as part of the team rather than act as a 3rd party auditing agent.

3.5 Security Centered Approaches

There are a number of solutions aimed at extending Agile and introducing new practices only for the sake of security in order to boost its assurance levels. Since Agile software development methodology is practiced in iterations, security reviews should be more appropriately conducted after each iteration rather than after the software is almost ready for production. This would ensure that after each iteration new functionalities are audited appropriately. Since security assurance provided by a 3rd party adds to the cost and to the development time, it goes against one of the agile principles which emphasizes more accelerated schedules and iterations. Concepts such as refactoring and testing might interfere with normal security assurance techniques in practice. For example refactoring functionality from one module to another can lead to new security vulnerabilities that was previously not a problem. Security testing introduces new levels of documentation requirements that might not be desirable to the agile methodology in the sense that they increase upfront design time associated with requirement gathering and analysis. Perhaps there is an expectation for agile to be all things to all types of projects but the reality is that agile only provides tangible benefits to projects that benefit from lightweight iterations and shorter lifecycles and therefore may not be a direct replacement for monolithic undertakings and very critical tasks that require meticulous security and documentation, and formality. At the end, the goal should be to not change or “extend” agile for security compliance but change security for inclusion into Agile because the fundamental principles behind Agile need to be kept intact in order for the process to work as it was originally intended (Ryan Jr and Scudiere 2008; Chung and Drummond 2009). In the following subsections we will discuss a number of proposed solutions that claim to accomplish security through adding and/or modifying certain Agile practices for better security assurance.

3.5.1 Security in Scrum

Erdogan in 2010 presented a case study consisting of the comparison between two testing methodologies commonly used for web applications (OWASP Testing Framework and Penetration Testing) with the Extended Agile Security Testing (EAST) methodology (which also needs the presence of a Security Engineer). EAST is an extension of the HTTP Unit testing tool which has been especially adapted to be used for projects utilizing Scrum (Erdogan, Meland et al. 2010). The aim was to be able to create an iterative process to be able to translate security requirements into test cases which is also compatible with the ideas proposed through TDD (similar to making security test cases from abuser stories). This study serves as an example of how the Agile is extended to include security steps but the authors claim it is much more suitable for Agile than comparable methodologies such as Penetration Testing Framework (PTF) and the OWASP Framework (OWASP 2011). The results of the case studies performed by authors indicate that even though the security assessment tools are effective at finding vulnerabilities, there is still a necessity for human intervention (i.e. Security Engineer) in order to mitigate the tools' limited capabilities. In conclusion, The EAST methodology introduced many steps that would reduce the resulting agility of the process and it was not tested compared to more mature traditional security mechanisms to assess its effectiveness and efficiency. It was only tested compared to a relatively ad-hoc process which did not include any steps similar to Penetration Testing and OWASP Frameworks.

3.5.2 Security in XP

According to Bostrom (2006) changing the Agile methodology for the sake of security alone and adding steps and documentation artifacts to XP in particular to have it conform to security requirements is a naïve approach towards bringing together security practices into Agile (Boström, Wäyrynen et al. 2006). Aydal et al. attempted to achieve good security within a web application (Aydal, Paige et al. 2006) by deferring the security checking until after all functionality was done and then dedicated a few iterations to take care of some security aspects towards the end of the project. Their solution involved outlining system-wide security issues through the use of Extreme Programming to prioritize and classify security mechanisms and then to implement

them using their own iteration at the end of the functional development iterations. This was done through the use of refactoring that aimed to provide a complete working solution at the end of each iteration even for security purposes. While this approach does provide some benefits, sometimes it is not a good idea to apply security mechanisms only at the end of the project because they may require substantial refactoring to accomplish and if this is done earlier while functionality is being built changes could be much more rapidly incorporated into the design and development process which can be assessed further through empirical measurements. We also intend to empirically ascertain whether or not the use of refactoring would indeed be considered beneficial for security from the standpoint of the experts and practitioners in the field.

3.5.3 Introducing Secure FDD

The following figure shows a typical FDD process:

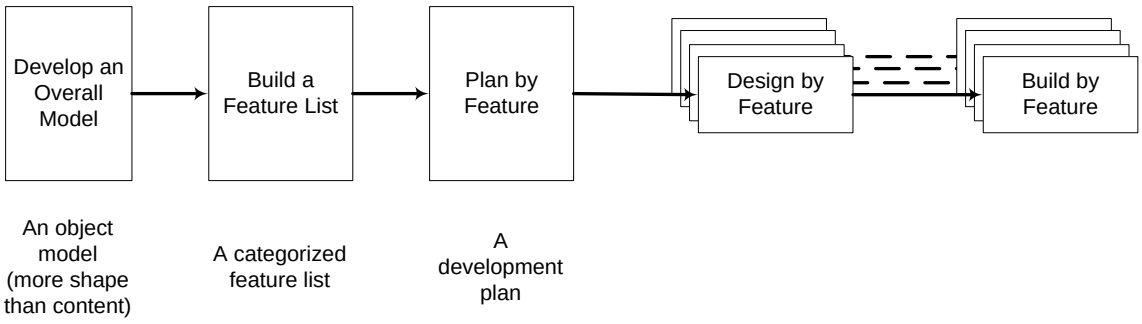


Figure 3-3 : FDD Process Overview (Ge, Paige et al. 2006)

One approach discussed in the literature claimed it modified the FDD process (Ge, Paige et al. 2006) to include security but in practice they introduced security related sub-tasks that still followed the same FDD process except the emphasis had shifted from functional features to security related features. Ge et al. modified the general FDD process to include additional steps that focuses on certain security specific issues at the Modeling and Design by Feature stages. The following figure 3-4 adapted from Ge et al. (2006) depicts a typical set of practices that anyone following the FDD process would normally go through with the new security practices shown in red boxes:

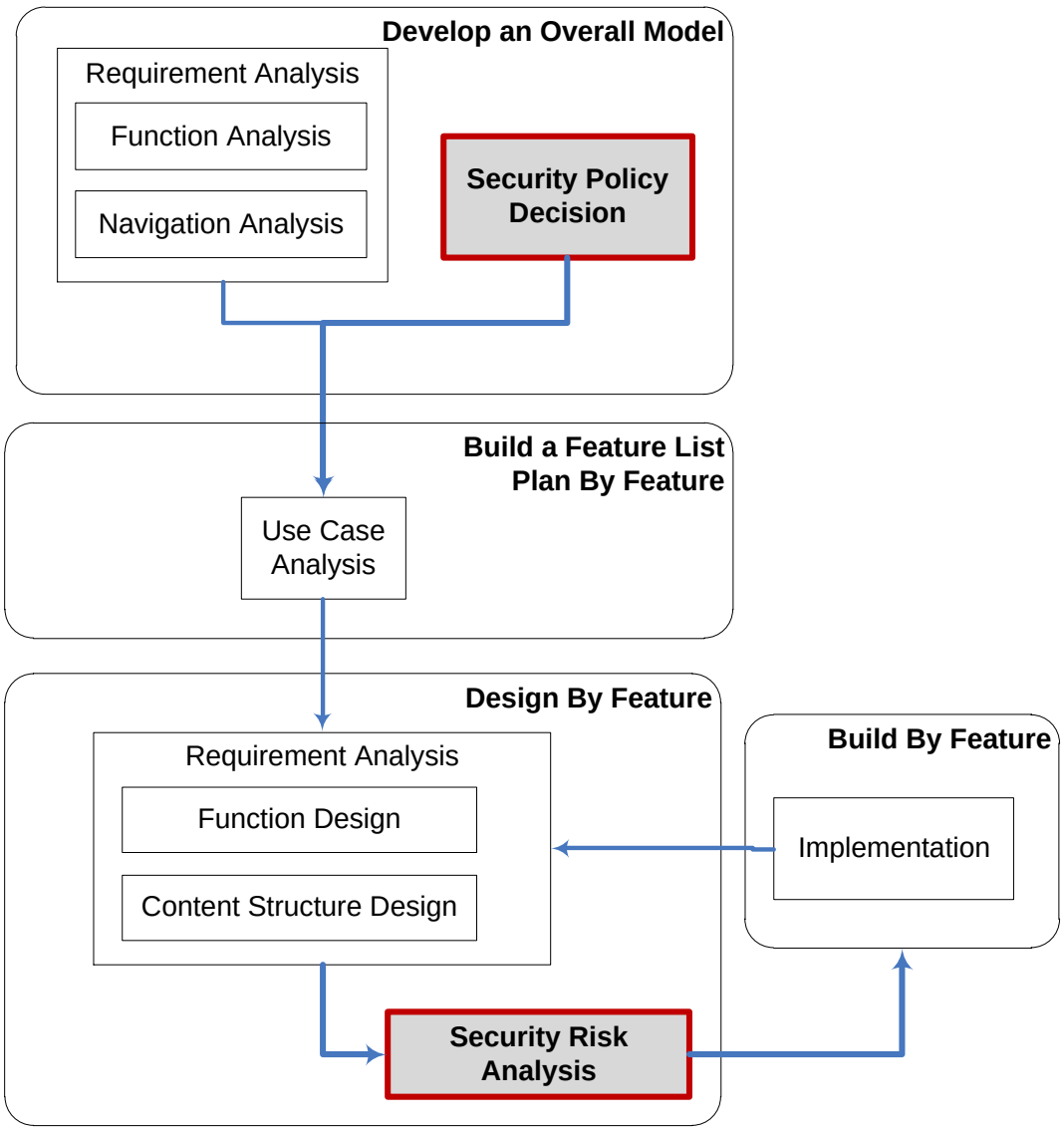


Figure 3-4 : Proposed Secure Web Application Process by (Ge, Paige et al. 2006)

The results of this research goes on to show how security related aspects such as risk analysis could be adapted to become part of iterative and incremental processes (Boehm and Turner 2003a; Boehm and Turner 2003b; Keramati and Mirian-Hosseinabadi 2008) that are used as part of Agile. Ge et al. claimed (Ge, Paige et al. 2006) that through the introduction of security risk analysis during the feature design iteration, the system could be made more secure through the use of "design with security in mind" but the case study failed to produce any design or implementation changes to the software being developed. The process itself was not changed entirely because only a part of the process was expanded to include the creation of security

policy document and security risk assessment documentation for every iteration and the rest of the process remained intact. This presents the question which is whether or not extending Agile processes for the sake of security really necessary (Chung and Drummond 2009; Doshi and Doshi 2009).

3.5.4 Extreme Security Engineering

According to Ge et al. (2006), "the growing trend towards the use of Agile techniques for building web applications means that it is essential that security engineering methods are integrated with Agile processes. This is, in itself, a substantial challenge". Beznosov introduced eXtreme Security Engineering (XSE) which was an application of XP practices for security applications that introduced the idea of "Good Enough Security" as a form of security assurance (Beznosov 2003). In this model, "Good Enough Security" allowed the customer to provide security related requirements and to be able adjust those requirements as much as they wanted. Industrial experience indicates security requirements are required up-front but this is a problem because people in charge of projects do not have a clear idea what kind of security they really need until close to the conclusion of the project. Just by the virtue of introducing this Agile technique the author claimed "Good Enough Security" was provided to security related projects and applications. On the same note, Beznosov makes the case that writing security requirements in the form of user stories help in bringing the security aspects into the Agile realm. This allows the customer to much better assess priorities and balance the need for security as well as functionality at every iteration if they deem it necessary. Since the customer can see the process move ahead frequently, they would be much more content that what is being done is worthwhile and they can also provide valuable feedback for better productivity and security of the final delivery. In XSE, customers must be in charge of creating functional test cases, while Security Engineers develop quality related unit tests to control the developed parts but this could be potentially problematic and needs to be empirically tested to see how it works in the real world.

3.5.5 Planning for Security

As part of implementing security in Agile, the iteration planning meetings were extended (Kongsli 2006) to include time for security issues. This was done to establish

collective ownership over security matters for the team. As the development progressed, the security issues were mitigated and tested for earlier in development than traditional models and this had the added benefit of earlier risk mitigation for the project in terms of security compared to traditional methods. A key benefit of integrating security into the Agile methodology is the relative increase in security awareness in designers and developers, and managers of the project because in the normal Agile process developers and architects are not usually concerned with security issues (Viega and McGraw 2002). Expanding on the latter, it would be advantageous for an iteration to be dedicated to security. In such iteration, the team members could think of ways to try and break the security barriers, in order to gain understanding of how vulnerable their software could be and to gain insight into how they can write better, more secure code. This is touched on through earlier research through the use of misuse or abuser stories but in our opinion they do not go far enough in achieving good enough security assurance. A dedicated security iteration would make it clear what the theme and the focus of the iteration is and makes sure all stakeholders are aware and on board to make it happen. It is further apparent that the collective (team based) ownership of secure coding is a benefit to the project (Kongsli 2006).

3.6 Agile Centered Approaches

The results of industry's two players on their Agile adoption (Yahoo and VeriSign) indicates that that in order to have successful Agile implementations, organizations need to strictly adhere to the rules and guidelines established by the principles and guidelines of the Agile manifesto (Ryan Jr and Scudiere 2008; Chung and Drummond 2009). They said any deviation from these rules and regulations resulted in some kind of shortcoming at different levels of the project and within teams. The takeaway point of these undertakings is to recognize the long term benefits of Agile that resulted in large parts of Yahoo and VeriSign adopting Agile and releasing better products faster (Ryan Jr and Scudiere 2008; Chung and Drummond 2009). The above empirical references make a strong point for why we need to have an Agile friendly approach as opposed to approaches that attempt to extend and modify Agile processes and/or practices in order to accomplish security specific tasks.

3.6.1 Security Engineer

According to Erdogan (2008), there is a knowledge gap between security experts and software developers (Erdogan and Baadshaug 2008). At the same time it is important to note that the best Security Engineer would be an experienced developer who has added training and knowledge of security (Level 1A or above). According to Cockburn (2002), "In an ideal situation – the sweet spot – the team consists of only experienced developers. Teams like this that I know report much different, and better, results compared with the average, mixed team. Because good, experienced developers may be two to ten times as effective as their colleagues, it is possible to shrink the number of developers drastically if the team consists entirely of experienced developers" (Cockburn 2002).

According to DHS in Security in the Software Life Cycle, one of the major threats to security in the software development lifecycle comes primarily from the insiders, a rogue developer who wishes to further their own malicious aims, by tampering with the code or corrupting one or more of the following (Goertzel, Winograd et al. 2007):

- Changing the design, architecture, or requirements such that there is an exploitable flaw in the software
- Changing or refactoring source code to introduce new vulnerabilities and/or defects that could lead to exploitation
- Tests and test cases designed to overlook certain defects or malicious code and pass even though some visible vulnerabilities could exist
- Documentation could be forged and/or omitted to mask certain undesirable functionalities
- The tools and/or processes used to develop the software could be compromised to share information on builds and/or other aspects of the software to third parties without the knowledge of the management

The presence of the security engineer and the periodic auditing and code reviews conducted either in person or automatically acts as a deterrent against such exploits by rogue insiders in addition to providing increased security assurance to the team and the

software. This not only benefits Agile projects, but also benefits other types of methodologies as well in terms of security.

Additionally, the periodic code reviews conducted by the Security Engineer could detect and subvert malicious code from being introduced into the system through special release. This is especially important to Agile methodologies because the rapid release schedule and integration of software makes it especially easy for a rogue insider to inject malicious code into the production environment.

In order for the costs to be kept to a reasonable level, it would be best to have the Security Engineer only be present during the security focused iteration(s) and for other normal iterations the development team can focus on functionality and the Security Engineer could work on other projects and come after a number of normal iterations have been completed. This would reduce operating expenses and promotes company-wide security adoption and also promotes all projects to be aware of security issues rather than one team with security emphasis. Tacit knowledge is increased throughout the teams since the Security Engineer would rotate between projects and makes developers aware of security issues. Therefore, the Security Engineer must have coding abilities and have a good security related training and/or experience to fulfill the role of Security Engineer and he or she must help create security requirements that are directly related to the proposed functionality of the system and avoid as much as possible sweeping overriding issues that goes beyond the scope of the project and what it needs to accomplish.

Additionally, every organization should have a security engineer that gathers prominent threats and figures out proper mitigation strategies at different levels and recommends features to be added by various Agile teams if he/she feels those security threats are better handled at that level. The security engineer should more appropriately become part of the team if he she has recommended the security feature to be implemented through software and act as a stakeholder and give direction and guidance as well as education to the developers on how to accomplish the given requirement.

This alleviates the team members and developers from having to think about overriding issues such as security so they can focus on building functionality while a dedicated expert will analyze and create requirements and assigns them to different teams to handle which reduces the overhead caused by having to hire a Security

Engineer per project or per team and also adds a layer between managers, the customer, and the development teams when it comes to security related issues. The Security Engineer further ensures the requirements are handled sufficiently with respect to the risks involved from both organizational aspects as well as specific team related aspects. The Security Engineer should ideally be a level 2 and above developer with specific security knowledge and training to be able to handle the assigned tasks.

3.6.2 Security Focused Testing

In the same way that testing provides quality assurance, security testing by contrast, should bring security assurance to the customer. Wichers advocated for the addition of security without fundamental change in the Agile methodology in order to create secure, high-quality software. He claimed that in Agile, test driven development establishes a strong argument for functional verification of code. This is a significant factor in trying to improve software quality in Agile software development. From the perspective of security however, the goal is to get a high-level of confidence in the software in terms of functionality and its resistance to vulnerabilities and threats. A problem in this respect is that customers are not usually aware of security issues and they should be informed about the risks and their impact on the software. The inclusion of security could have a negative impact on the time and the budget and may therefore increase budget and time constraints. As a result the issues should be carefully examined to find a balance between security and functionality of the resulting software (OWASP 2008; Wichers 2008).

According to SSA SOAR "Security testing techniques for software are still immature and collectively represent an incomplete patchwork of coverage of all security issues that need to be tested for." (Goertzel, Winograd et al. 2007). Kongsli (2006) introduced an approach towards addressing security in Agile through adding abuse cases to user stories and transforming them to unit tests for automated security testing. He also introduced security regression testing. Early on, there have been usage scenarios that had been created in order to increase overall security of the software (securing deployment, regression testing, system hardening, and penetration testing). According to SSA SOAR, "Tests to discover design defects are difficult to develop. Like the systems engineers developing security designs, the testing group (whether independent

or not), will be able to construct test cases based on understanding the psychology of the attackers and knowledge of typical software, hardware, and other system fault types" (Goertzel, Winograd et al. 2007).

Integrating security early on helps with integration because continuous integration is part of the standard model of behavior in Agile development. Beznosov claimed that having two Security Engineers working as one on a given security problem produces more high-quality results but again these are untested opinions at this point which require further study. It is evident that this was an attempt to bring together security engineering practices into Agile software development. However, the way the author went about implementing each step seemed to suggest all the Agile principles in some way or another could be adapted to security even though this came at the cost of overburdening the customer with security related testing and other responsibilities and the addition of two security engineers acting as one to accomplish the same task seems cost prohibitive (Beznosov 2003). At the end all of the proposed changes were not supported empirically which underscores the need for more quantitative analysis and review.

3.6.3 Dedicated Security Iteration

According to Kongsli (2006) traditional security methodologies have traditionally been sequential and it counters the Agile principles of iterative design. Then why not make an entire iteration be a security iteration so the process becomes sequential just like a traditional security method. To go one step further, a three iteration security solution could be used which involves dedicated iterations for vulnerability assessment (to find all vulnerabilities), risk assessment (to decide which vulnerabilities need to be fixed), and risk mitigation (to fix the vulnerabilities) to act as a multi-step approach towards overall project level security assurance.

According to Wichers, one can create functionality for a few iterations but sooner or later security issues should be discussed before deployment and as an ongoing practice to make the resulting application more robust and secure. It might be beneficial to dedicate an entire iteration to security issues because security issues are inherently interrelated so it makes sense to try and take care of them all at once because it'll make the process more effective and efficient and at the same time provides a clear separation from the normal Agile practice of creating functionality which is the prime focus of the

project. Whether or not to dedicate an entire iteration to security related issues or to just implement security as part of each iteration, it should ideally depend on priorities established as part of the project and therefore it is not to be made strictly one way or another in order to achieve a better security assurance argument (OWASP 2008; Wichers 2008).

Another proposed idea is to create security related requirements and once there's enough requirement(s) that warrants a whole iteration, conducting security iteration is appropriate to create tests related to how protected against vulnerability the system is, and then run those tests without specifically having written code to protect against it and see the results. Based on the results of the testing, vulnerabilities could be investigated and mitigated and after every potential fix the security tests should be repeated. Given the number of security scenarios that could come about, the customer might agree to an iteration that involves testing the system to see how secure it is against those possibilities.

3.6.4 Standard Security Components

Building security components is an involved and complicated task and therefore it might not be even be possible to try and accomplish that as part of a standard iteration where people are responsible for creating functionality as well as security. Security is an overriding issue that goes beyond the scope of a typical software development project. According to Wichers, standardized tests and best practices, as software components can be created in order to make the overall software written across projects more secure and effective. Therefore, trying to implement security into each iteration of the process is not the best way to handle security in general and for Agile specifically. There are cases where adequate security has already been implemented as part of the infrastructure, therefore it might not even be of any value to want to implement yet another layer of security into application as well. The overriding question is how security can be reused from project to project in order to become effective for Agile otherwise the benefits are not realized if the security implementation only benefits one singular project and efforts should be repeated for the other projects (OWASP 2008; Wichers 2008).

3.7 Risk based security model

One important measure of how critical a security issue is lies in assessing its risk to the business or the organization in question. It remains to be seen whether or not the risk should be mitigated or eliminated through the application or if there are other places that more effectively address those vulnerabilities. Whether or not to integrate a security mechanism into working software or not is a big issue in Agile software development methodology. First of all, it adds additional costs to take care of security and related issues and also it extends the project timeline by a few iterations. According to OWASP, major security vulnerabilities should be ranked according to their level of risk that they represent to the company. Application Security Verification Standard should be used to assess whether or not the application is protected against the most significant security threats and vulnerabilities to provide a sound security assurance argument. Normally in Agile teams the security assurance argument is made through security related testing which are based on abuser stories and threat modeling. This method of security verification and mitigation strategy is superior to the proposed methods of abuser stories and threat modeling because of community involvement and the standard nature of the effort in assessing the most common, dangerous, and risky threats employed by attackers today (Wichers 2009; OWASP 2010).

In 2010, Williams presented the "Protection Poker" as a practice that helps to conduct security risk estimation during planning meetings. It relied on the security knowledge of the meeting's participants to be able to correctly assess the risk and create useful misuse or abuse cases. The claimed benefits were to increase security of the software but in reality the only tangible benefit of this scheme was in attaching an estimated risk value to each piece of functionality at every meeting. This method assumes that there exists some kind of security related resources specific to a given Agile project which is highly unlikely in many real world projects and organizations. One of the outcomes of the case study involving the "Protection Poker" uncovered that there was a need for more security specific training in order to reduce vulnerabilities with the application under review (Williams, Meneely et al. 2010).

It is known that developers are not trained for security specifically, so unless they have been trained beforehand to know security related issues and risks, they will not be able to function effectively as creators of good abuse cases. The practice requires that

"the entire extended development team" to be present at the meeting which is highly unlikely and not practical in the real world. Therefore, the benefits of the "Protection Poker" are similar to a regular risk assessment analysis and it is not fit to handle the broader set of requirements needed to establish overall security at the project level. Williams et al. attempted to present primarily a risk assessment tool as a full-fledged security extension to Agile and iterative types of projects but only tested it in a limited case study involving system maintenance activities. The conclusion at this point is that the results are only narrowly tested and it remains to see the effects of having achieved a good enough security for the system or not (Williams, Meneely et al. 2010). It remains to be seen how effective these tools are in promoting the increased security of the resulting software but at the very least the relative security assurance compared to traditional methods is higher.

3.8 Issues and relevant literature

Table 3-4 contains a list of summarized issues discussed as part of this literature review on major security issues in Agile development along with our formulated position on each issue based upon our review and experience as software engineers and developers. The opinions and positions reflect the literature, both empirical and non-empirical sources on the topics.

#	Software Engineering Perspective	Supported Source(s)
1	Addition of a Security Engineer as a permanent member to the Agile team **Position: Yes, sounds feasible and effective**	(Wäyrynen, Bodén et al. 2004)
2	Adding documentation to address vulnerabilities and threats **Position: Not good because documentation reduces agility, maybe automated tools could be used to achieve the same outcome**	(Viega and McGraw 2002; Kongsli 2006)
3	Adding risk analysis to be able to see which methodology, plan driven or Agile to use for a given project **Position: May be good before project starts. Not really part of Agile**	(Boehm and Turner 2003a; Boehm and Turner 2003b; Keramati and Mirian-Hosseinabadi 2008)

4	Using testing to make security assurance argument **Position: Good but how to express security issues as functional test cases**	(Kongsli 2006; Kongsli 2007)
5	Making the programmers create misuse/abuser stories **Position: Requires security training and expertise which developers usually do not have**	(Siponen, Baskerville et al. 2005; Boström, Wäyrynen et al. 2006; Kongsli 2006)
6	Whether or not to use COTS in solutions and their security assurance levels **Position: Not good because hard to evaluate from the security standpoint based on SOAR**	(Boehm and Turner 2003a; Goertzel, Winograd et al. 2007)
7	Is modifying agile to include security practices really necessary? **Position: Not necessary because of time/cost issues and lack of customer motivation for security**	(Chivers, Paige et al. 2005; Boström, Wäyrynen et al. 2006)
8	Education and Awareness of security Issues **Position: Great, but how to unobtrusively integrate into Agile.**	(Wäyrynen, Bodén et al. 2004; Kongsli 2006; Ge, Paige et al. 2007)
9	Assessing security of the software through a security iteration at the end of the project. **Position: Good but not good enough because doing security at the end of the project is challenging in terms of refactoring and efficiency**	(Siponen 2001; Viega and McGraw 2002; Bishop 2003; Aydal, Paige et al. 2006)
10	The need for a standardized security mechanism for Agile **Position: Highly desirable IF it does not extend the methodology itself which tends to reduce agility**	(Baskerville 2004; Chivers, Paige et al. 2005)
11	The role the customer should play in decision making on security issues **Position: Yes on decision making, no on assessment because they are not aware of certain security aspects and only an expert can assess issues properly**	(Goertzel, Winograd et al. 2007; OWASP 2008; Wichers 2008)
12	The need for better tools which provide additional flexibility in the face of security related requirements **Position: Great but good tools not available, highly proprietary and costly**	(Kongsli 2006; Goertzel, Winograd et al. 2007; Kongsli 2007; Erdogan, Meland et al. 2010; Williams, Meneely et al. 2010)
13	To what extent do testing and QA tools help Agile projects produce higher quality software? **Position: They generally help a great deal functional-wise but in terms of security unknown**	(Huo, Verner et al. 2004; Kongsli 2006; Kongsli 2007)

14	Do the basic founding principles of Agile themselves contribute to a lack of security assurance and awareness? **Position: Yes and no depending on the type of people used in Agile teams, also touched on in SOAR report**	(Goertzel, Winograd et al. 2007)
15	The need for good metrics to establish a relative scale for security assurance. **Position: Great idea, OVAL provides a baseline standard schema for representing vulnerabilities which could be used as basis for establishing a security assurance argument**	(Goertzel, Winograd et al. 2007)
	Security Engineering Perspective	
16	SSE-CMM and CC (security) integration into Agile **Position: Very difficult unless modified and reduced to suit Agile specifically**	(Wäyrynen, Bodén et al. 2004; Boström, Wäyrynen et al. 2006)
17	Agile Security Mechanism (as opposed to Traditional Security) **Position: Desirable but suffers from timing issues (interferes with Agile)**	(Beznosov 2003; Kongsli 2006)
18	Relationship between security requirements and organization size **Position: This is not really Agile related, but possibly has impact on Agile**	
	OTHER	
19	Agile makes problems visible earlier in development **Position: Potentially good for security**	(Huo, Verner et al. 2004)

Table 3-2 : Major security issues in Agile

3.9 Research Questions

Given the fact that the aim of the research was to investigate the predominant security issues in Agile methods we set out to compile and critically review papers throughout the literature that touched on security and Agile. The result of this review provided us with 19 major issues from various perspectives and points of view from academics and practitioners around the world. The topics were mainly Agile focused and also involved security as a focus area. From this point and from the above table 3-2 it was clear that we needed to reduce the list of major security issues found in our literature review into a manageable list which still had elements from other aspects represented within the questions. Therefore, we decided to extract the most general and common topics that were touched on by these issues that also happen to have some commonalities between the found issues and our general research direction which guided us to identify a set of research questions we wanted to investigate

empirically. The following table provides the link between our research questions and topic(s) commonly referred to between the previously identified issues derived from the literature and our general research questions intended to add an investigative aspect to the research.

#	General Topic	Touched on Issue(s)
Q1	Combining Security & Agility	1, 2, 5, 8, 9, 10, 16 and 17
Q2	Change Agile Practices to address Security	2, 3, 5, 7, 8, 10, 14, 16, 17 and 19
Q3	Security Issues and Software Vulnerabilities	6
Q4	Customer Control over Security	11
Q5	Knowledge Dissemination and Diffusion	1 and 8
Q6	Dedicated Security Iteration	9
Q7	Testing and Verification Tools for Security	4, 12 and 13
Q8	Documentation and Security	2, 5 and 14

Table 3-3 : List of commonalities between Major Security Issues in Agile Methods

The following is our list of research questions that we derived from the literature with adherence to our research topic which was to investigate and to find answers for the most predominant security issues in Agile:

RQ1. How to seamlessly combine security and agility, and evaluate the suitability of security mechanisms for use in agile?

RQ2. Is changing one or more agile practice(s) for better security really necessary and what kind of projects would require security more than others?

RQ3. To what extent do security issues stem from software vulnerabilities and/or defects?

RQ4. Should the customer be able to dictate the level of required security for a given project and how much control should they be given?

RQ5. What are the effects of knowledge dissemination and diffusion on a given agile project in terms of security awareness of its members, especially developers?

RQ6. How beneficial is dedicated security iteration and under what circumstances should they be initiated?

RQ7. To what extent do the testing and verification tools provide additional security assurance in addition to quality assurance?

RQ8. Does the reduced documentation in agile projects lead to less secure software even with the advent of CASE tools?

3.10 Summary

This review touched on all the security related issues and their impact and implications on the Agile projects from multiple perspectives (also included in table 3-2). This allows for better understanding of the outstanding issues which provides insights into trends and apparent requirements for security in various Agile projects. While there are many specific questions that could be asked to shed light on a small aspect of this investigation, we opted to ask more general, pointed questions that aim to answer, more broadly, questions that have important implications on a number of specific issues and their underlying details. Once the answers for these questions are known, the answers to the more specific questions become much clearer.

Chapter 4 The State of Security in Agile: Semi-Structured Interviews

4.1 Introduction

4.1.1 Basic characteristics of the qualitative study

Qualitative procedures are a form of inquiry that relies on forms of data such as text, video, sound, etc. in order to gain a deeper more insightful understanding of the issues involved than what purely quantitative methods are able to offer. There are multiple qualitative procedures and strategies that can be followed in a qualitative study such as Social Justice Thinking (Denzin and Lincoln 2005), Ideological Perspective (Lather and Lather 1991), Philosophical stances (Schwandt 2000), and Systematic Procedural Guidelines (Corbin and Strauss 2007; Creswell 2007). Although each of these is useful in certain circumstances, our strategy is to combine these strategies in order to be able to accomplish our intended tasks.

The literature review on the predominant security issues in Agile methodologies gave us an idea about our target audience and their relative level of understanding of the issues involved. It also made us aware of the qualitative characteristics that we could and could not pursue. The qualitative characteristics mentioned here are taken from (Eisner 1991; Bogdan and Biklen 1992; LeCompte and Schensul 1999; Hatch 2002; Marshall and Rossman 2006; Creswell 2007) and represent both traditional and newer perspectives that apply to our research:

- **Researcher as key instrument:** Qualitative researcher gathers data by him/herself through interviewing participants. A protocol is used but the researcher is the one that actually developed the procedure and gathered the information.
- **Comprehensive source of data:** Qualitative researcher gathers a large amount of data from interviews with participants, and takes steps to review the information, make sense of it, and organize it into themes that will be useful to the area of study.
- **Inductive Data Analysis:** Qualitative researcher builds categories, patterns, and themes from the ground up through codifying the information into more abstract data units. The inductive process depicts the process of going between the

themes and the data source until the investigator has found a complete set of emerging themes and abstractions from the data gathered.

- **Participants' meanings:** In qualitative research, we focus our attention on learning the underlying meanings and issues that the participants expressed about the problem or the issue, not the meaning that other researchers expressed in the literature.
- **Emergent Design:** Qualitative research is emergent which means that there could not be a tightly prescribed initial plan in place. This is because all aspects of the process may change or shift after we begin to collect data. The key here is to try and learn as much about the issues as possible from the participants and to modify the research plan accordingly.
- **Interpretive:** Qualitative studies are interpretive inquiries where the researcher interprets what they observe. Of course these interpretations are intermingled with the researcher's history, background, prior knowledge, and context of the field. Since the readers as well as participants can read and offer their own interpretations of the study, this brings forth multiple interpretations and allows multiple views of the issues to emerge.
- **Holistic Account:** The qualitative researcher tries to build a complex, big picture account of the issue under review. The larger picture emerges through the identification and reporting of multiple perspectives and factors involved. They may be able to provide a sketch to illustrate this big picture to aid the reader in better understanding of the process of a phenomenon (Creswell and Brown 1992).

4.1.2 Qualitative strategy of inquiry

There are many specific strategies and approaches that researchers presented over the years that focus on data collection and analysis. Tesch (1990) identified 28 approaches (Tesch 1990), Wolcott (2001) identified 19 (Wolcott 2001), and Creswell (2007) identified 5 approaches (Creswell 2007) to qualitative inquiry. While many possibilities for qualitative inquiry exists (phenomenology, narrative, case study, ethnography, or grounded theory) exist, a method different from, but similar to grounded theory was chosen by us because it represented the closest match to our goal of finding consensus on major security issues in agile methodologies which

produces a number of hypotheses for further study to be tested later on quantitatively. Another distinctive difference from our strategy and the established grounded theory method is that even though we did employ a data driven approach to inquiry we also used the prior research to identify major security issues which makes it distinctly different from a grounded theory perspective but similar in later stages of arriving at an emerging theory or a set of hypotheses.

This method is most appropriate because we needed to incorporate many issues that had come up in our process of systematic literature review on predominant security issues and given the fact that many practitioners are not familiar with security aspects and issues in general, we felt our study would have been incomplete had we only elected to generate hypotheses based upon only the perspective of the practitioners.

4.1.3 The researcher's role in the study

In order to be able to interview the practitioners in-depth, the first and most productive approach would have been to conduct a case study on premises and perform a combination of observation, on-site interviews and follow-up interviews or questionnaires over time. However due to the sensitive nature of the topic being security issues in software development methodologies virtually every credible organization we contacted refused to participate often citing reasons given to them by their legal department on why it couldn't be done, even though we made assurances of anonymity and confidentiality for both the organization and the participants. This left us to follow through with the next best approach which was to directly interview practitioners outside of their professional work environment in order to gain an insight into their practices and knowledge surrounding the topic.

To be able to carry out such a plan we contacted the Electronic and Computer Science School Ethics Committee at the University of Southampton and applied for the necessary approval to carry out interviews with subjects involving details about the subjects with the following selection criteria:

> "The subjects will be chosen based upon their level of experience and domain knowledge on the issues related to agile software development and secure software development. The subjects would be chosen from a pool of authors, practitioners, managers, and stakeholders that have related their experience to these topics and are interested to participate in the interview. The selection criteria would include being from a background of software engineering and development and related fields that directly impact the agility and security of the final product. Contact will be made in person, through the internet, through the supervisors, workshops, and conferences, and after that through various forms of communication such as Phone, Email, etc."

4.1.4 Steps in gaining entry

After the ethic approval process was completed (reference number E10/09/11) we began our search for suitable candidates and after an extensive survey of the practitioners in the industry both in UK and abroad we were able to secure a number of interviews ranging anywhere from 23-110minutes. The geographical distribution of the participants was carefully taken into consideration for us to have a representative sample from around the world.

4.1.5 Sensitive ethical issues

Complete anonymity was provided for participants and they were assured by the university and the researcher that their information was not going to be released in any way to the general public without their express consent due to the extremely sensitive nature of the topics under study and the relative importance of their roles in their respective position in often very reputable organizations.

4.2 Research Design

4.2.1 Sampling strategy for sites and/or individuals

We wanted as part of the interviews to have a geographically diverse number of individuals that practiced Agile and Security from various roles and organizations. The setting for the interviews was partly chosen to be professional groups that meet routinely in order to share information and update individuals with the latest tools,

training, and technologies. The following table shows the results of the background information for each individual interviewed that lists their professional attributes that could be shared without exposing any personally identifiable information.

No	Exp (yrs)	Location	Interview Length	Method	Iteration Length (weeks)	Iterations per Release	Role	Type of Organization
1	9	Taiwan	75 min	Scrum + XP	4-5	4-5	Developer	Software/ Systems Development
2	2	Japan	110 min	Other Agile	2-3	3-4	Developer	Electronic Company
3	4	U.K.	35 min	XP + other Agile	2	1	Developer/QA	Financial organization
4	1	China	95 min	XP	2-4	3-6	Developer	Software Development Company
5	3	U.S.	77 min	Scrum	2	24	Developer	Electronic Company
6	20	U.S.	38 min	XP	per story	Varies	Developer/Manager/QA	Software Developing Company
7	15	U.K.	30 min	TDD	2	1	Developer/QA	Financial organization
8	15	U.K.	35 min	Other Agile	2-6	Varies	Consultant	Consultation Firm
9	5	U.K.	52 min	Other Agile	2	Varies	Security/Software Engineer	Software/ Systems Development
10	16	U.K.	23 min	Other Agile	Varies	Varies	Manager	Financial organization
11	12	U.K.	29 min	Scrum	2	8-12	Agile Coach/Manager	Dealer Company
12	14	U.K.	66 min	Scrum	Varies	Varies	Consultant/Scrum Master	Consultation Firm
	10	U.K.		XP	2	Varies	Lead Developer/ Manager	Financial organization
13	28	U.K.	37 min	N/A	N/A	N/A	Consultant/Manager/ Developer	Consultation Firm
14	21	U.S.	57 min	N/A	N/A	N/A	Security/ Consultant	Security Firm
15	7	Saudi Arabia	94 min	Waterfall	N/A	N/A	Manager/ Security Auditor/ Developer	Telecom Company

Table 4-1: Professional Interviewee Background Information

4.2.2 Forms of data collection

The following discussion of participants considers the 4 aspects that might be involved that were proposed by Miles and Huberman: The setting, the Actors, the Events, and the Process (Miles and Huberman 1994). The form of data collection was chosen to be semi-structured interviews and participants were asked a number of open ended questions. Based upon answers given to those questions, a number of

more specific follow-up questions were asked from participants in order to gain a much deeper insight and understanding of the issues under investigation. As part of the questioning, the interviewees were gauged in their manner of response as well as the quality of their answers given the sensitivity of the issues. If an interviewee for example expressed or cited concern with answering a question then we immediately moved to the next question to maximize our chances of getting relevant information if not directly receiving answers to our main question. After a number of interviews we had a good idea which questions were more sensitive than others and we optimized the questioning based upon our previous experiences interviewing previous participants. This is why the method of semi-structured interviews proved to be most useful and appropriate in this situation.

As part of the semi-structured interviews some participants were interviewed in a face-to-face, one-on-one, in-person manner and others who could not be reached in person by interviewers were interviewed by phone. The advantages of these two options were dependent upon the situation of each interviewee and the relative amount of time and convenience that they needed in order to feel comfortable responding freely to our questions. This was also important in order to be able have control over the questions and line of questioning in order to maximize the usefulness of the responses.

4.2.3 Rationale for choosing these data collection methods

We acknowledge that this method is not as desirable as a direct observation technique since the information that we get is filtered through the views of interviewees but given the extremely sensitive nature of the research that deals with security issues and organization's lack of cooperation in sharing such information directly we maintain that the data collection method used was the best possible for the research at this time.

4.2.4 Recording information protocol

For the semi-structured interviews, we recorded the audio of the majority of the interviews and proceeded to transcribe and summarize the information provided. The steps are as follows:

1. Company information (related to the type of the organization and not the specific name of the company) and their role in the company will be asked (their job title and not their name)
2. Start with a general question relevant to the topic of the interview.
3. Based on the answer from the first question, have specific follow-up questions available to ask (see Appendix B) that may not be necessarily chosen beforehand.
4. Framework for the interview in the form of an interview guide.

The gathering of email addresses and phone numbers are used only for the purposes of contacting the individual interviewees. This information was kept separately in a secure place aside from the actual data gathered and was promptly destroyed completely after the interviews were conducted.

4.2.5 Process of Generating Interview Questions

Since we were interviewing the practitioners working in real projects, we elected to ask general, open ended questions that were worded specifically with a given practitioner's working environment in mind. For example, instead of asking them a specific question regarding an issue found from the literature asking "Is modifying agile to include security practices really necessary?" we elected to ask them about their "project or code level security practices" that they may or may not be employing in their every day work environment. This was done with 2 specific purposes in mind:

1. Open ended question allowed for more data to be captured beyond the scope of what the specific issue from the literature would have.
2. The question took into account the various differences between teams of professionals working in the field.

Asking questions in this manner is what was required by a qualitative study that aims to capture data beyond what was raised by the prior research on the topic(s) under investigation. The following table lists 8 general questions from the interviews:

Interview General Questions	
#	Description
1	What are your project or code level security practices that are in use today in your organization?
2	Is security as important to agile projects as many would like to have us believe?
3	Which specific practice in your project contributed the most to increased security of the resulting software?
4	Which factor(s) are most responsible for causing vulnerabilities and weaknesses in software in terms of security?
5	Is there an observed or apparent trend towards increased security of software in agile projects?
6	How can security related aspects such as risk analysis be effectively made to become part of iterative and incremental processes?
7	Does every project inevitably end up with some security vulnerabilities and what can be done to mitigate this risk?
8	What is the best way to model Security issues as part of agile projects?

Table 4-2: Sample of Semi-Structured Interview Questions

4.3 Analysis Steps

4.3.1 Identify data analysis steps

A typical qualitative analysis involved the collection of data by the researcher who then proceeds to identify and collect themes or perspectives, and presents them in a concise and human understandable form (Creswell 2008).

As part of the analysis we followed a similar approach to Grounded Theory (Strauss and Corbin 1990; Strauss and Corbin 1998; Corbin and Strauss 2007) with a key difference of using prior research data to generate interview questions for use in the analysis and interpretation stages as well as using data gathered qualitatively through semi-structured interviews. The type of analysis used follows the Hybrid Approach mentioned in a book titled “Transforming Qualitative Information” by Boyatzis (1998) that outlined steps to follow (Boyatzis 1998) as part of the analysis and interpretation of qualitative data:

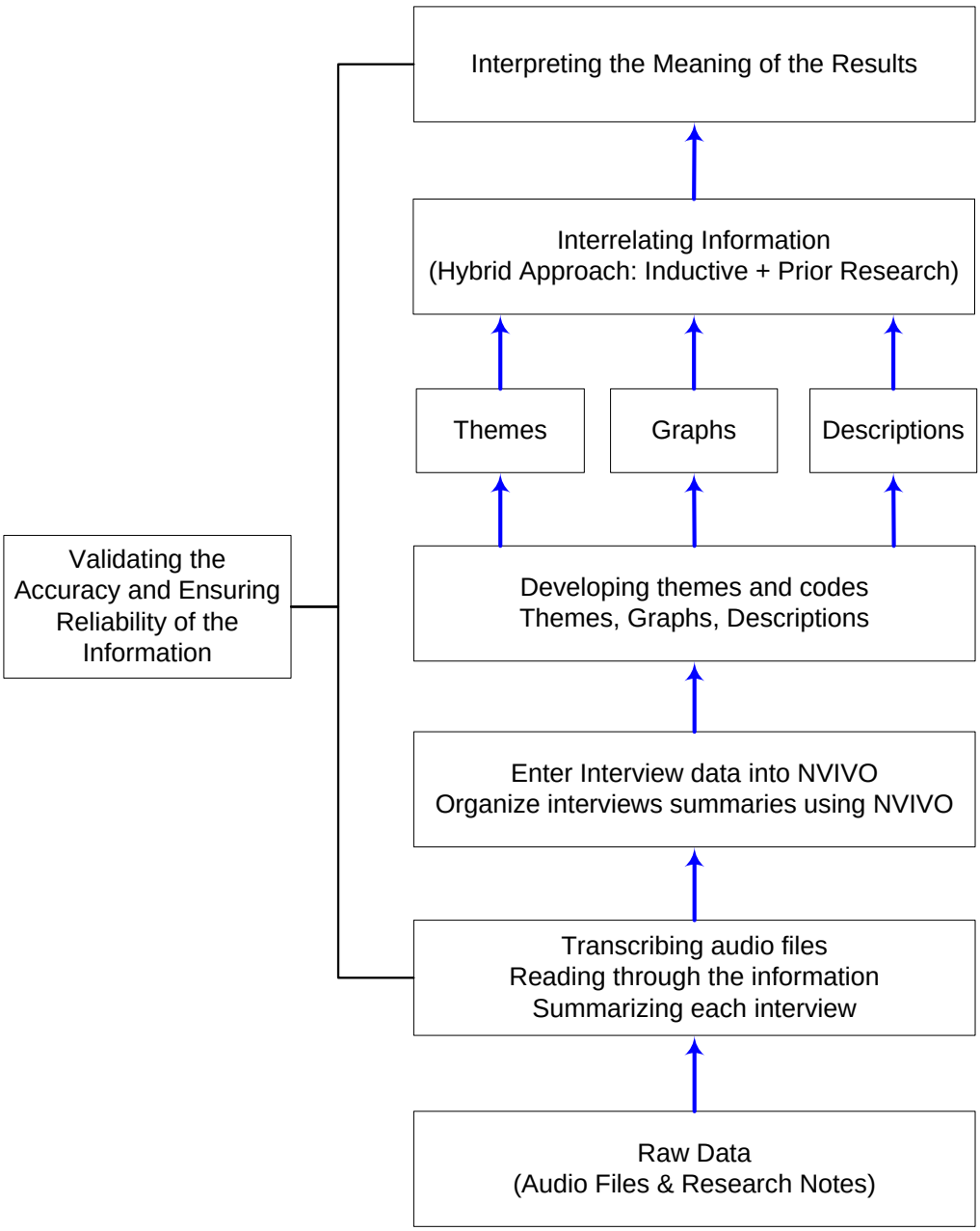


Figure 4-1 : Data Analysis Steps in Qualitative Research

4.3.2 Evidence for preparing data for analysis

There was over 15 hours of raw data captured in the form of audio that was later transcribed into over 140 pages of written textual material. The transcriptions were read over and summarized into 120 pages of relevant useful information and were subsequently entered into NVIVO by the researcher. From that point the information was analyzed and codified into over 665 general codes which were later reduced to 419 highly relevant and specific instances useful and directly pertaining to the topics

of interest. The basis for developing the codes consists of a combination of codes that emerge from the data as well as codes that were pre-determined from the literature.

The 419 codes were then used to identify 34 major themes that were conceptually gathered into 8 groups based on the research questions. The following figure shows the major research questions along with the distribution of codes for each question:

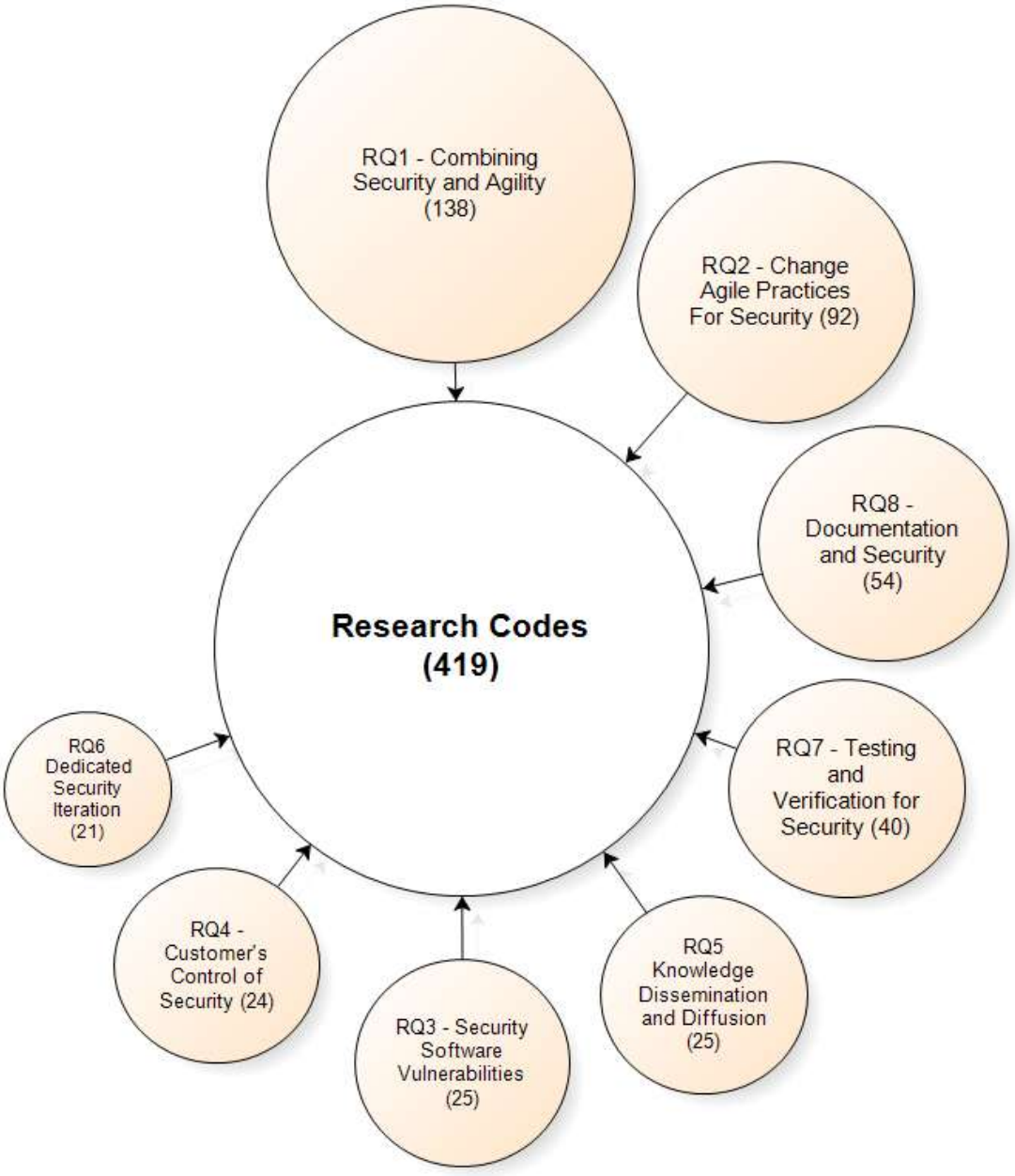


Figure 4-2 : Research Questions and Codes

4.3.3 Developing Themes and Codes

As we mentioned earlier most of the themes are emerging (Inductive or Data Driven) while some are based on the literature. Data driven codes were built inductively from the transcribed and later summarized interview data. According to Boyatzis the researcher is responsible for analyzing the data, interpreting the meaning of finding and to build hypotheses after obtaining results (Boyatzis 1998). Since the closeness of the code to raw information is high, the likelihood that people will code the raw data similarly is increased and therefore this approach has a relatively higher interrater reliability than other approaches. With a complete view of the information the researcher can recognize "gross (i.e. easily evident) and intricate (difficult-to-discern) aspects of the information." (Boyatzis 1998). Each theme includes a number of perspectives expressed by individuals that answered a particular question related to the theme and is supported by a number of specific quotes (as evidence). The themes were then organized conceptually based on their relationship to each of our specific research questions and taken collectively formed an interconnected and complex picture that should be evident from the responses. The following are some examples of how the transcript was coded and classified according to the nature of the answer(s) to given questions under investigation:

Excerpt from an actual interviewee response to question #4: "Which factor(s) are most responsible for causing vulnerabilities and weaknesses in software in terms of security?"

Transcribed response given by Interviewee #1:

> "If you are in hurry then you will have more chances to write a less secure code. This is common in all projects. This is about the maturity of your skills [the developer]. A more knowledgeable and skillful person could avoid making security vulnerabilities in their code. Usually, you are under the pressure of the schedule."

Extracted Code #1:

Codified Version of Transcription	"This is about the maturity of your skills [the developer]. A more knowledgeable and skillful person could avoid making security vulnerabilities in their code."
Code Label	Level of Experience is key to avoid security vulnerabilities in the code
Classified Under	RQ1 – Combining Security and Agility / Experience of Developers

Extracted Code #2:

Codified Version of Transcription	"If you are in hurry then you will have more chances to write a less secure code. This is common in all projects."
Code Label	Schedule Pressure as factor for causing security vulnerabilities and weaknesses
Classified Under	RQ2 – Change Agile Practices for Security / Impact of Accelerated Schedules

4.3.4 The structure of a Useful, Meaningful Code

According to Boyatzis "A good thematic code is one that captures the qualitative richness of the phenomenon" (Boyatzis 1998). He also stated that a good thematic code has 5 elements:

1. **A label (i.e. a name):** "Should be conceptually meaningful to the phenomenon being studied. It should be clear and concise, communicating the essence of the theme in as few words as possible and it should be close to the raw data."
2. **Definition:** "A definition of what a theme concerns (i.e. the characteristic or issue constituting the theme)"
3. **Indicators/Flags:** "A description of how to know when the theme occurs (i.e. indicators on how to "flag" the theme)"
4. **Exclusions:** "A description of any qualifications or exclusions to the identification of the theme"
5. **Examples:** "Both positive and negative, to eliminate possible confusion when looking for the theme"

Based on the above guidelines provided by Boyatzis, the following table is provided that shows the Code, label, and definition for each codified theme which form the basis for later stages of our analysis. For a comprehensive and detailed theme information refer to appendix C with a corresponding theme code from the summary table below:

Research Question	Theme Code	Label
Combining Security & Agility	RQ1A	**Dedicated Security Engineer:** The addition of a highly experienced Developer and/or Software Engineer/Architect to the team to act as the Formal, Dedicated Security Engineer for the project.
	RQ1B	**Software with Security in mind:** The claim that consideration of security and having security specific knowledge will help in creating more secure software.
	RQ1C	**Security Controls:** The use of Standard or 3rd party Security Specific controls within software as an alternative to provide added security to the resulting software.
	RQ1D	**Informal Security Experts:** Sometimes, if the team lacks the formal security expert as a source for guidance, they turn to the most experienced person in the team with known or implied security experience.
	RQ1E	**Integration Risk:** The notion that adding or attempting to add security practices and/or procedures to Agile projects will affect Agile in undesirable ways.
	RQ1F	**Experience of Developers:** The idea that the more knowledgeable and experienced the developer, the more the resulting software will be secure.
	RQ1G	**Security in planning:** Dedication of some time to discuss security issues with the software team during planning stages as part of the process.
	RQ1H	**Static Analysis Tools:** Tools and/or software programs that aid in finding various issues in terms of security of the resulting software.
	RQ1I	**Static Code Reviews:** Techniques and/or practices that allows developers to find weaknesses and vulnerabilities within the code base.
Change Agile Practices for Security	RQ2A	**Agile vs. Traditional Methods:** The opinion that Agile either in part or as a whole produces software that is different in terms of security than more traditional upfront-designed and developed software development mechanisms.
	RQ2B	**Awareness of Security to Agile:** The concern that the stakeholders of the project may not recognize the importance and may not even be aware of the potential impact security has on the software.
	RQ2C	**Impact of Accelerated Schedules:** The notion that doing things faster and being given a shorter timeframe to get tasks done adds an element of risk to various stages of software development lifecycle in terms of security.

	RQ2D	**Reduced Security for Internal Projects:** The recognition that software that is going to be only available internally within a given organization can have less security requirements.
	RQ2E	**No change Necessary:** The opinion that Agile is better off not being changed in favor of security by the addition of certain security specific practices and/or processes.
Security Issues and Software Vulnerabilities	RQ3A	**Security through resolving vulnerabilities:** The recognition that in order to achieve better security of the resulting software all vulnerabilities must be taken care of before release.
	RQ3B	**Relationship between defects and Security:** Based on the notion that there is a relationship between software bugs, defects, and vulnerabilities and the resulting security of the software.
	RQ3C	**Impact of Experienced Developers:** The point that having more experienced and knowledgeable developers on the team will contribute to the resulting security of the software.
	RQ3D	**The role of security training:** The claim that the knowledge of security contributes to creating a more secure software compared to the case where there is no specific security training for the team members.
Customer's Control of Security	RQ4A	**Customer must Dictate all requirements:** The opinion that the customer should be responsible for and provide all needed requirements even though they might not be in the best position to understand certain issues.
	RQ4B	**Customer needs help with security:** The notion that the customer however knowledgeable they may still need more technical help in understanding security issues in order to understand the importance of security to the project.
Knowledge Dissemination and Diffusion	RQ5A	**Developers lack of Security knowledge:** The recognition that developers are not trained on security specific issues and may not be in the best position to make security related decisions.
	RQ5B	**Tacit knowledge Dissemination and Sharing:** The notion that as it relates to agile the most effective form of informal training and making various stakeholders aware of security issues is through tacit knowledge sharing and dissemination.
	RQ5C	**Security through Training:** The obvious choice that in order to learn security issues one must be specifically trained for it.

Dedicated Security Iteration	RQ6A	**Periodic Security:** The idea that for every few normal functional iterations there should be one or more steps focused on security issues taken as part of the iteration.
	RQ6B	**More public needs more Security:** The recognition that the more exposed the software is going to be to the public the more uncertainty in terms of security it would face and therefore more attention should be placed on the security aspect.
Testing and Verification for Security	RQ7A	**Refactoring:** The idea that refactoring helps with security in general.
	RQ7B	**Testing for security:** The notion that QA in addition to developers should be conducting unit testing and other security focused tests to increase the security assurance.
	RQ7C	**Tools provide added Security Assurance:** The recognition that tools in general add to the level of assurance for a given purpose such as for security focused testing and verification, and other security aspects.
Documentation and Security	RQ8A	**Documenting the Agile way:** The idea that the traditional forms of documentation used in waterfall and other traditional methods are no longer effective and could be replaced by more modern tools and/or practices.
	RQ8B	**Agile not less secure than traditional methods:** The idea that just because agile produces less documentation that it somehow results in less secure software produced.
	RQ8C	**Abuse or Misuse Stories:** The idea that the use of security focused user stories such as misuse or abuser stories would increase the security assurance argument.
	RQ8D	**Reducing Documentation Risky in Agile:** The recognition that while reducing documentation is desired, too much reduction of documentation could add more risk to the project.
	RQ8E	**Testing over Documentation:** The idea that forgoing writing of unnecessary documentation could be replaced with more effective practices such as more Testing.
	RQ8F	**Simpler code better than more documentation:** The idea that writing simpler, self documenting code is preferred over documentation that is hard to maintain and update.

Table 4-3 : Summary of Themes from the Semi-Structured Interviews

4.3.5 Units of Analysis and Coding

The unit of analysis according to Boyatzis (1998) is the topic in which the interpretation of the research will focus. In our case, the unit of analysis is the predominant security issues in Agile teams in which the various stakeholders that were interviewed operate. In contrast, the unit of coding is the smallest part of the raw data that could be evaluated in a useful way regarding the unit of analysis which in our case is the opinions of the individuals that are part of the agile team. These opinions are in the form of a response (or a part of the response) to each interview question.

4.4 Results of Interviews

4.4.1 Frequency Scoring, Scaling, and Clustering themes

According to Boyatzis (1998), Scaling means the act of combining two or more coded themes into a single score and Clustering means the organization of two or more coded themes into groups (Boyatzis 1998). Similarly, according to Coffey and Atkinson (1996), clustering is a way of organizing data to assist in the process of analysis and interpretation (Coffey and Atkinson 1996). As such, we are grouping themes discovered through our semi-structured interviews conceptually based on our research questions and each code discovered (empirically) would be rated based on the frequency of occurrence once per interviewee per question (consensus index) and presence or absence for measuring occurrence of the issues. Similar themes could be applied to more than one research question and we discovered this to be the case during the process of coding and analysis of results. We employed frequency scoring because we wanted to be able to describe and analyze the observation using numeric representation.

4.4.2 Occurrence Index

In order to find the significance of each issue among the participants in the interviews, we chose to obtain a frequency score for each participant on the issue(s) they raised during the interview on the various questions that we wanted to discuss that could tell us about their preferences from various perspectives and points of view. The results of the empirical observations gathered from the interview data will

help in determining the most predominant and important security issues in Agile methodologies as reported by practitioners.

Our frequency scoring is calculated based on an nominal scale of the number of mentions by the interviewee at various points throughout the interview transcript on various issues discussed. The scale is relative because in order to find which issues are more popular with practitioners, their opinions could be counted as an affirmative or negative vote on a given issue.

We also calculated the overall occurrence for each research question which was calculated by dividing the number of occurrences (or themes within that research question) by the total number of occurrences for all themes which was 419. For example, for the first research question, the number of occurrences divided by the total was 138/419 which is .329 which means 32.9% of the responses included statements which related to the first research question. The results of this process are as follows:

4.4.2.1 RQ1 Combining Security and Agility

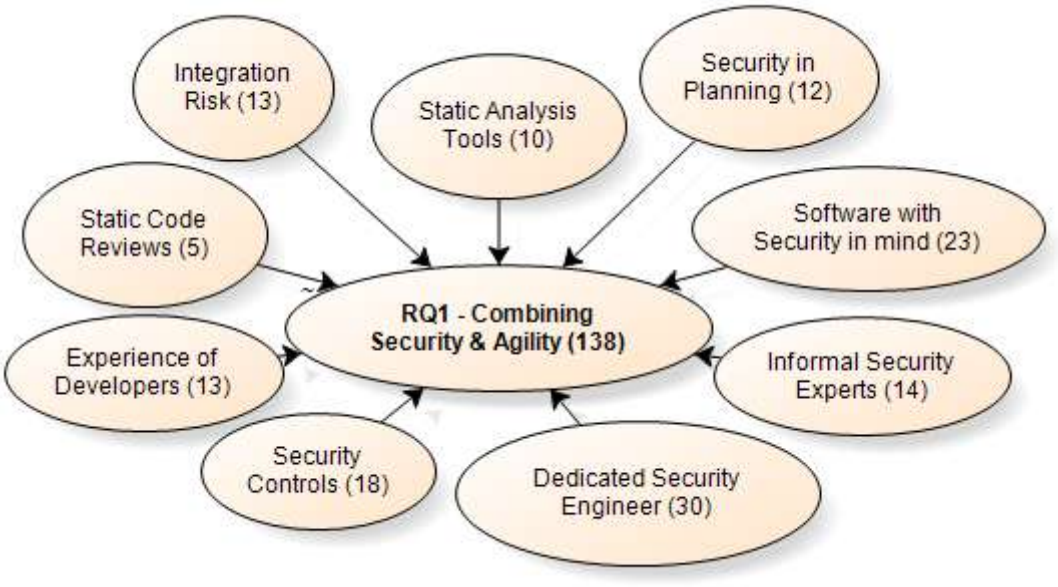


Figure 4-3: Overall Themes for RQ1

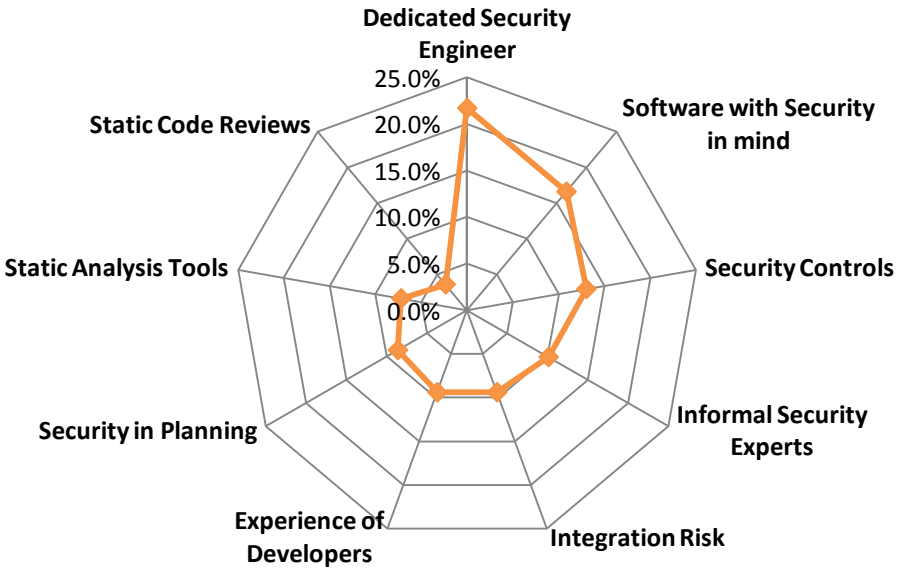


Figure 4-4: Occurrence Index Graph for RQ1

Theme Code	Theme Description	Overall Frequency	Theme Frequency	Interviews Cited	Occurrence Index
RQ1A	Dedicated Security Engineer	138	30	**12**	**21.7%**
RQ1B	Software with Security in mind	138	23	**11**	**16.6%**
RQ1C	Security Controls	138	18	**4**	**13.0%**
RQ1D	Informal Security Experts	138	14	**8**	**10.1%**
RQ1E	Integration Risk	138	13	**7**	**9.4%**
RQ1F	Experience of Developers	138	13	**6**	**9.4%**
RQ1G	Security in Planning	138	12	**8**	**8.6%**
RQ1H	Static Analysis Tools	138	10	**6**	**7.2%**
RQ1I	Static Code Reviews	138	5	**2**	**3.6%**
				Total:	**100%**
				Overall Occurrence:	**32.9%**

Table 4-4: The Frequency Details for RQ1

4.4.2.2 RQ2 Change Agile Practices for Security

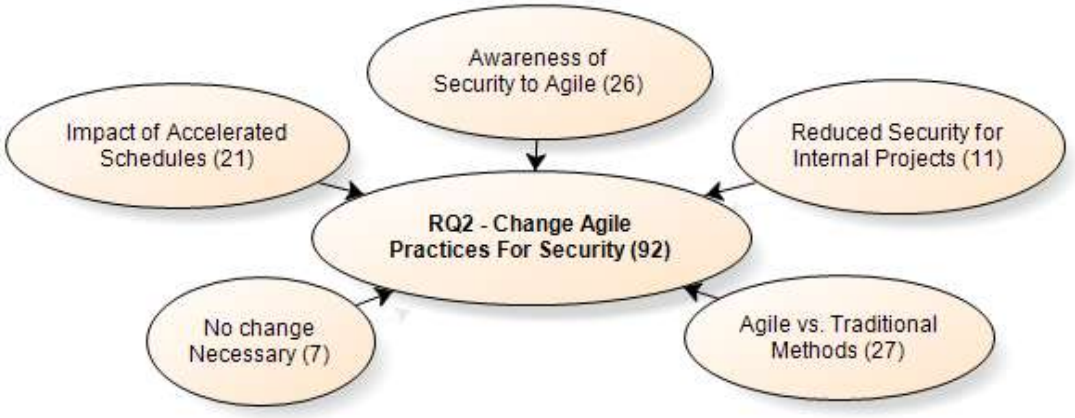


Figure 4-5 : Overall Themes for RQ2

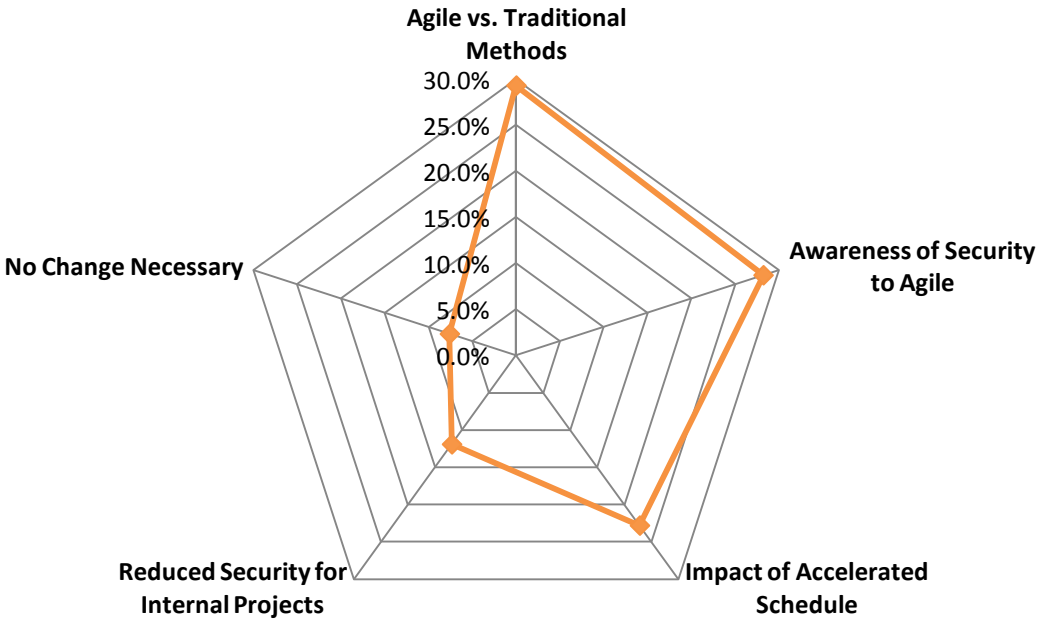


Figure 4-6 : Occurrence Index Graph for RQ2

Theme Code	Research Question Theme	Overall Frequency	Theme Frequency	Interviews Cited	Occurrence Index
RQ2A	Agile vs. Traditional Methods	92	27	**11**	**29.3%**
RQ2B	Awareness of Security to Agile	92	26	**11**	**28.2%**
RQ2C	Impact of Accelerated Schedule	92	21	**10**	**22.8%**
RQ2D	Reduced Security for Internal Projects	92	11	**6**	**11.9%**
RQ2E	No Change Necessary	92	7	**5**	**7.6%**
				Total:	**100%**
				Overall Occurrence:	**21.9%**

Table 4-5 : The Frequency Details for RQ2

4.4.2.3 RQ3 Security Issues and Software Vulnerabilities

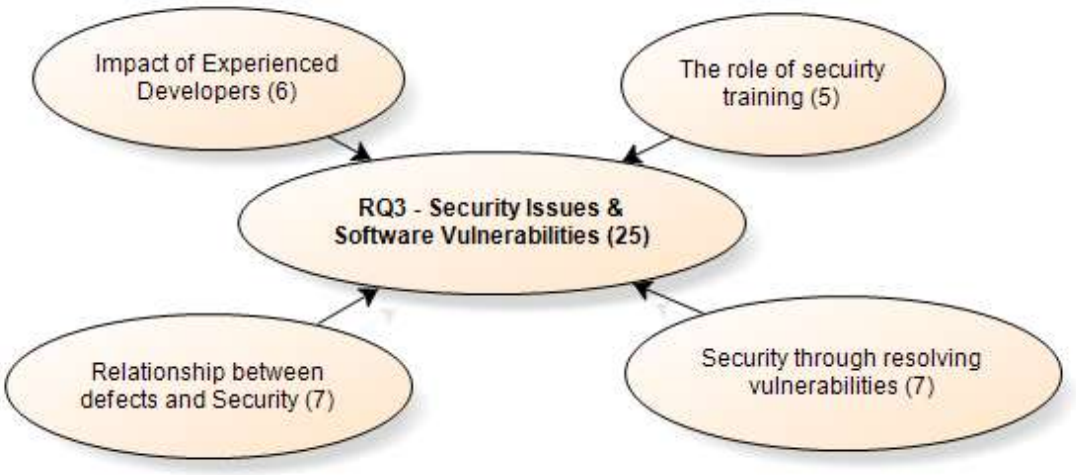


Figure 4-7 : Overall Themes for RQ3

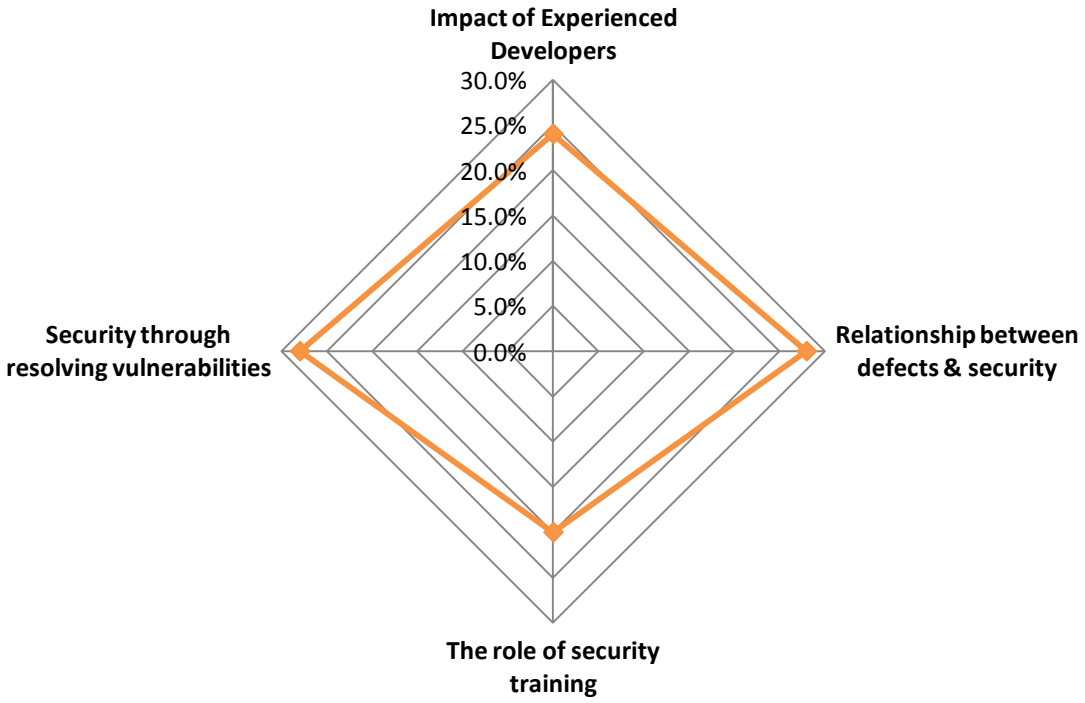


Figure 4-8 : Occurrence Index Graph for RQ3

Theme Code	Research Question Theme	Overall Frequency	Theme Frequency	Interviews Cited	Occurrence Index
RQ3A	Security through resolving vulnerabilities	25	7	**5**	**28.0%**
RQ3B	Relationship between defects & security	25	7	**5**	**28.0%**
RQ3C	Impact of Experienced Developers	25	6	**2**	**24.0%**
RQ3D	The role of security training	25	5	**4**	**20.0%**
				Total:	**100%**
				Overall Occurrence:	**5.9%**

Table 4-6 : The Frequency Details for RQ3

4.4.2.4 RQ4 Customer's Control of Security

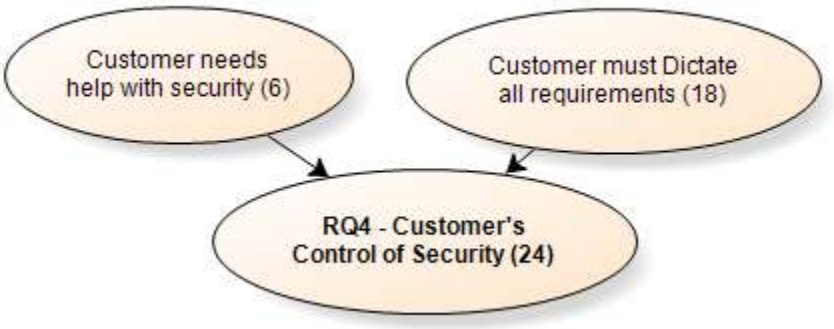


Figure 4-9 : Overall Themes for RQ4

Figure 4-10 : Occurrence Index Graph for RQ4

Theme Code	Research Question Theme	Overall Frequency	Theme Frequency	Interviews Cited	Occurrence Index
RQ4A	Customer must Dictate all requirements	24	18	**12**	**75.0%**
RQ4B	Customer needs help with security	24	6	**5**	**25.0%**
				Total:	**100%**
				Overall Occurrence:	**5.7%**

Table 4-7 : The Frequency Details for RQ4

4.4.2.5 RQ5 Knowledge Dissemination and Diffusion

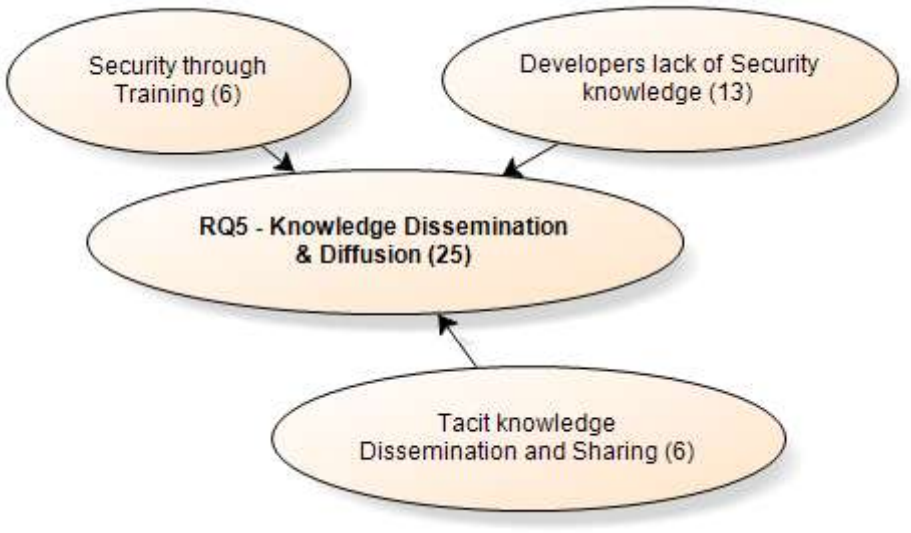


Figure 4-11 : Overall Themes for RQ5

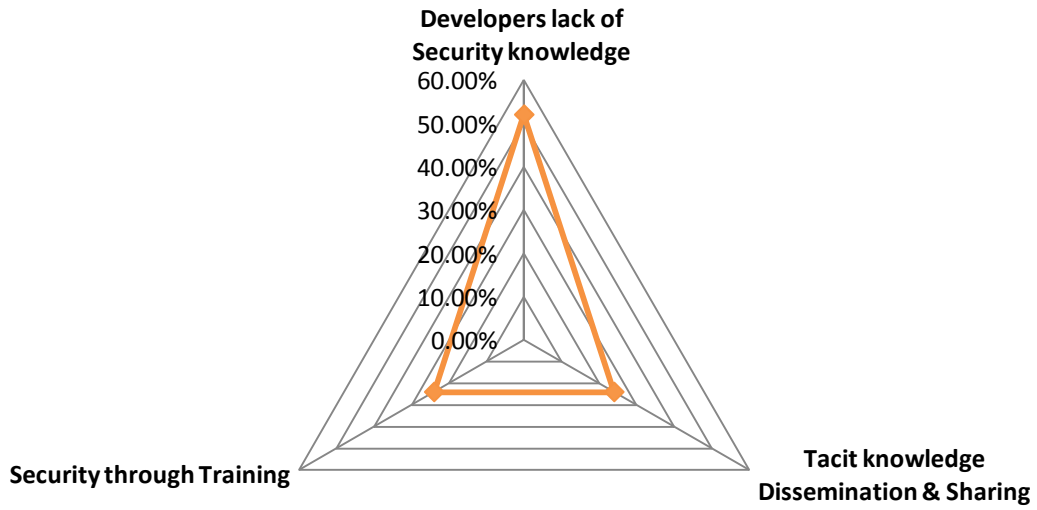


Figure 4-12 : Occurrence Index Graph for RQ5

Theme Code	Research Question Theme	Overall Frequency	Theme Frequency	Interviews Cited	Occurrence Index
RQ5A	Developers lack of Security knowledge	25	13	**9**	**52.0%**
RQ5B	Tacit knowledge Dissemination & Sharing	25	6	**4**	**24.0%**
RQ5C	Security through Training	25	6	**6**	**24.0%**
				Total:	**100%**
				Overall Occurrence:	**5.9%**

Table 4-8 : The Frequency Details for RQ5

4.4.2.6 RQ6 Dedicated Security Iteration

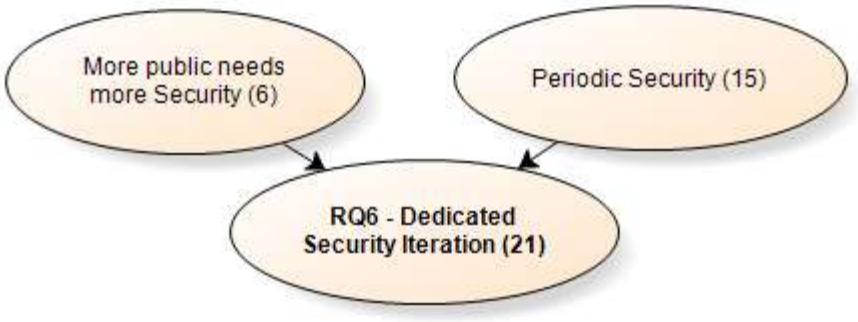


Figure 4-13 : Overall Themes for RQ6

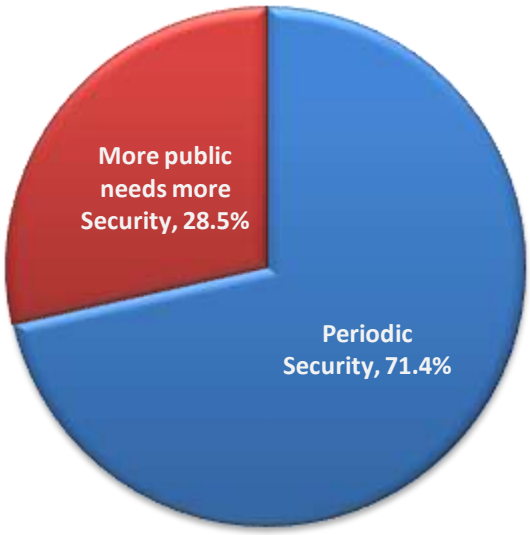


Figure 4-14 : Occurrence Index Graph for RQ6

Theme Code	Research Question Theme	Overall Frequency	Theme Frequency	Interviews Cited	Occurrence Index
RQ6A	Periodic Security	21	15	**9**	**71.5%**
RQ6B	More public needs more Security	21	6	**5**	**28.5%**
				Total:	**100%**
				Overall Occurrence:	**5.0%**

Table 4-9 : The Frequency Details for RQ6

4.4.2.7 RQ7 Testing and Verification for Security

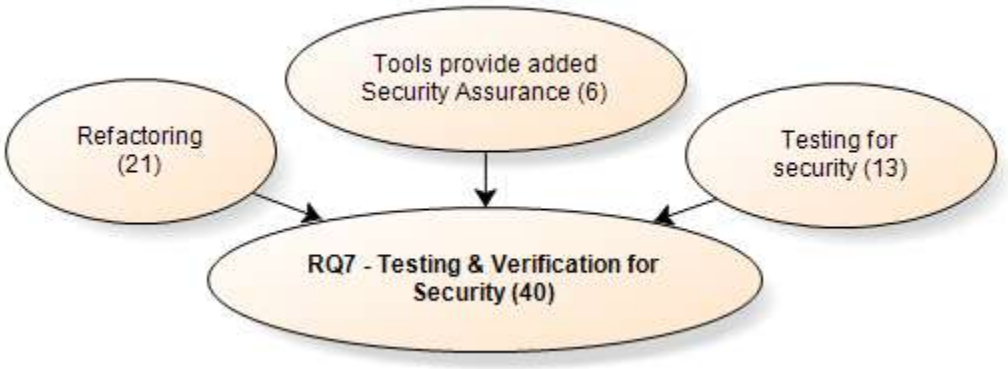


Figure 4-15 : Overall Themes for RQ7

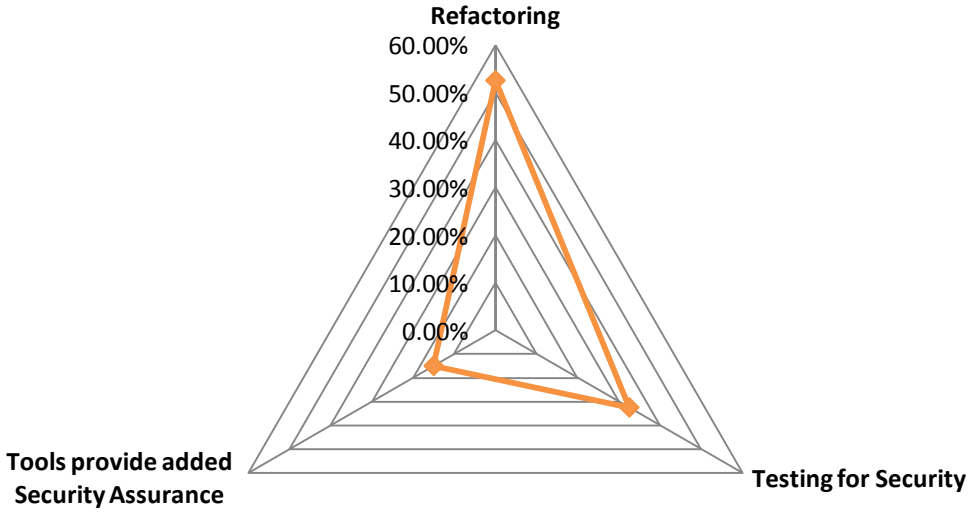


Figure 4-16 : Occurrence Index Graph for RQ7

Theme Code	Research Question Theme	Overall Frequency	Theme Frequency	Interviews Cited	Occurrence Index
RQ7A	Refactoring	40	21	**12**	**52.5%**
RQ7B	Testing for Security	40	13	**8**	**32.5%**
RQ7C	Tools provide added Security Assurance	40	6	**3**	**15.0%**
				Total:	**100%**
				Overall Occurrence:	**9.5%**

Table 4-10 : The Frequency Details for RQ7

4.4.2.8 RQ8 Documentation and Security

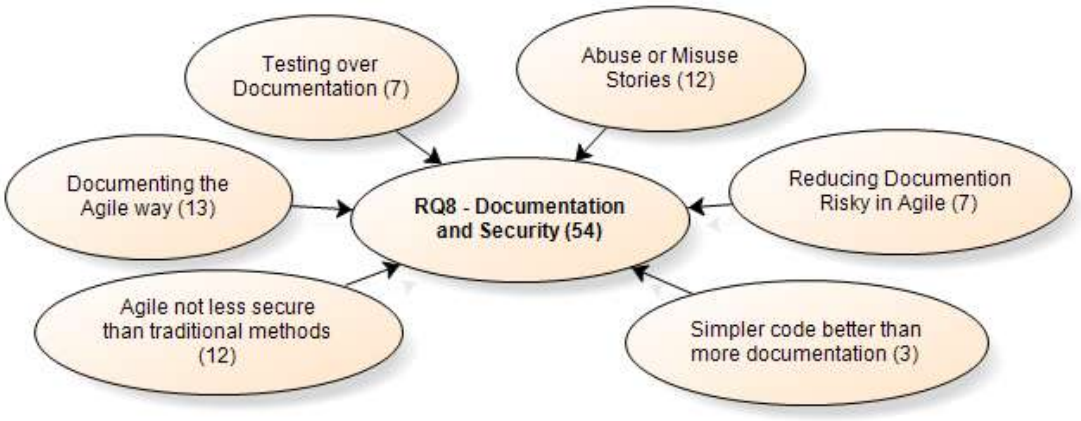


Figure 4-17 : Overall Themes for RQ8

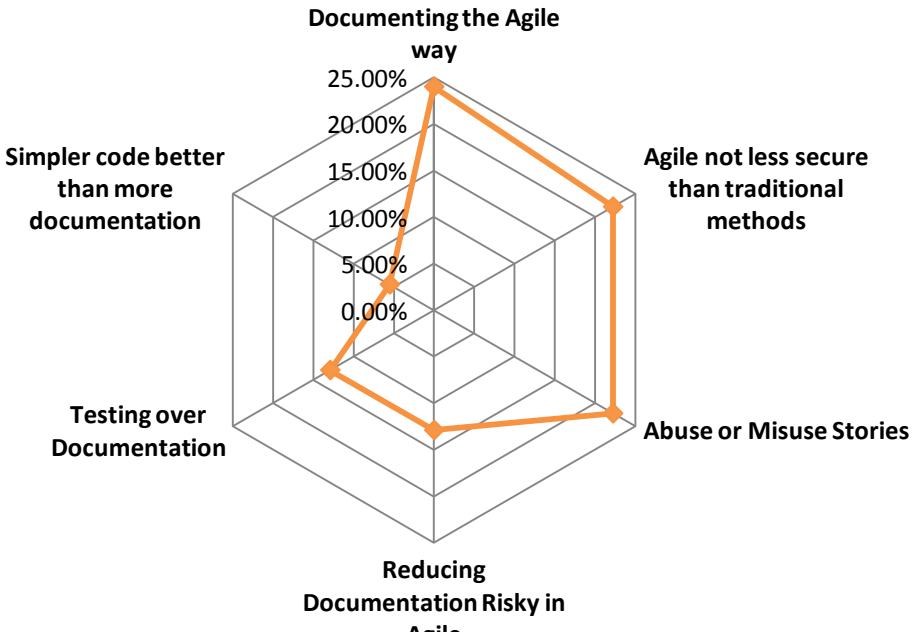


Figure 4-18 : Occurrence Index Graph for RQ8

Theme Code	Research Question Theme	Overall Frequency	Theme Frequency	Interviews Cited	Occurrence Index
RQ8A	Documenting the Agile way	54	13	**9**	**24.0%**
RQ8B	Agile not less secure than traditional methods	54	12	**6**	**22.2%**
RQ8C	Abuse or Misuse Stories	54	12	**5**	**22.2%**
RQ8D	Reducing Documentation Risky in Agile	54	7	**3**	**12.9%**
RQ8E	Testing over Documentation	54	7	**5**	**12.9%**
RQ8F	Simpler code better than more documentation	54	3	**1**	**5.5%**
				Total:	**100%**
				Overall Occurrence:	**12.8%**

Table 4-11 : The Frequency Details for RQ8

4.4.3 Consensus Index

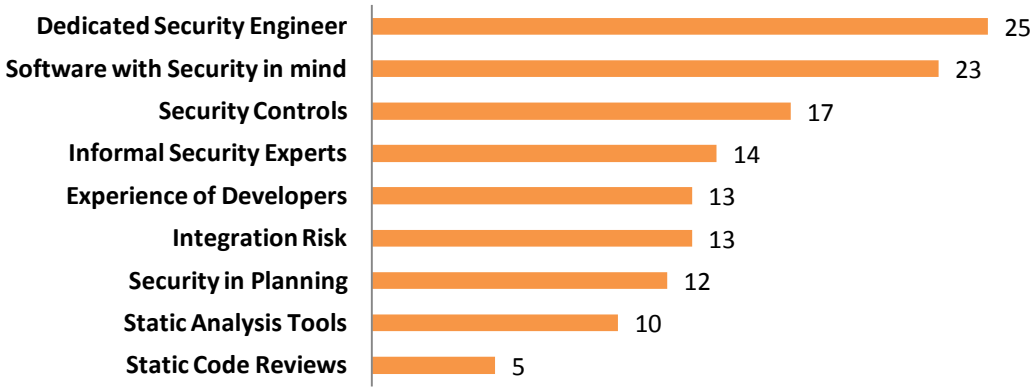
For our research, and in order to find the needed consensus among the participants in the interviews, there was a need to obtain a frequency score for each participant on the issue(s) they raised during the interview on the various issues that we wanted to discuss that could be considered as answers to our research questions from various perspectives and points of view. Although there was variety in the roles of the interviewees and the fact that they might not have directly mentioned some clear points on a given issue, the entirety of the responses and their overall reasoning lead us to infer some answers (inductively) to questions that may not have been immediately obvious from the wording of the response. Here are some examples of how we codified the transcript in Nvivo first a positive one and then a negative example:

Codified Version of Transcription	"So it is trade-off and I think the specialist [security engineer] is a faster solution, it is easier to implement. You recruit someone and interview them and decide that they are good enough bring them into the building and give them a desk and get them involved in the process. Now adding that, it's not process change but it's a faster and lower risk solution than having to educate the whole team or change the methodology [for security]"
Code Label	Positive - Having security engineer will be a faster solution and lower risk
Classified Under	RQ1 – Combining Security and Agility / Dedicated Security Engineer

Codified Version of Transcription	"I have in count of people in my classes that often report that security code specialist is bottleneck [insufficient] within the organization and the ability to do code security constrains the throughput of the velocity and I cannot talk about specific practices because I am not close enough to it."
Code Label	Negative - Security Engineer is considered as bottleneck within some organizations
Classified Under	RQ1 – Combining Security and Agility / Dedicated Security Engineer

Our frequency scoring is calculated based on a scale of the number of consensus points mentioned by the interviewee at various points throughout the interview transcript on various issues discussed. The scale is relative because in order to find which issues are more important, practical and/or popular with practitioners their opinions could be counted as an affirmative or negative vote on a topic. Since we are only concerned with the affirmative votes in the case of consensus, the scale is established to be scored on the basis of the number of affirmative votes on an issue under discussion. The results of this process are as follows:

4.4.3.1 RQ1 Combining Security and Agility



.

Figure 4-19: Overall Consensus for RQ1

Theme Code	Research Question Theme	Overall Frequency	Overall Responses	Positive Responses	Consensus Percentage
RQ1A	Dedicated Security Engineer	138	30	**25**	**83.3%**
RQ1B	Software with Security in mind	138	23	**23**	**100%**
RQ1C	Security Controls	138	18	**17**	**94.4%**
RQ1D	Informal Security Experts	138	14	**14**	**100%**
RQ1E	Integration Risk	138	13	**13**	**100%**
RQ1F	Experience of Developers	138	13	**13**	**100%**
RQ1G	Security in Planning	138	12	**12**	**100%**
RQ1H	Static Analysis Tools	138	10	**10**	**100%**
RQ1I	Static Code Reviews	138	5	**5**	**100%**

Table 4-12: Consensus Details for RQ1

4.4.3.2 RQ2 Change Agile Practices for Security

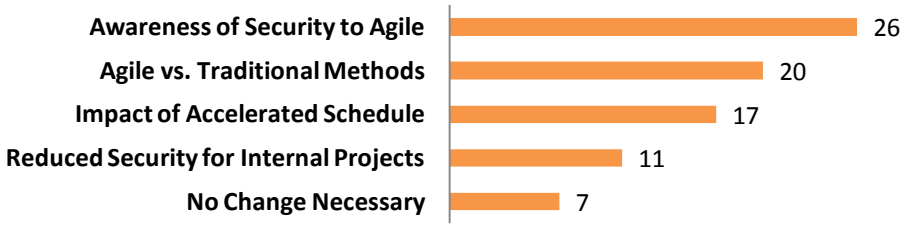


Figure 4-20 : Overall Consensus for RQ2

Theme Code	Research Question Theme	Overall Frequency	Overall Responses	Positive Responses	Consensus Percentage
RQ2A	Agile vs. Traditional Methods	**92**	27	**20**	**74.0%**
RQ2B	Awareness of Security to Agile	**92**	26	**26**	**100%**
RQ2C	Impact of Accelerated Schedule	**92**	21	**17**	**80.9%**
RQ2D	Reduced Security for Internal Projects	**92**	11	**11**	**100%**
RQ2E	No Change Necessary	**92**	7	**7**	**100%**

Table 4-13 : Consensus Details for RQ2

4.4.3.3 RQ3 Security Issues and Software Vulnerabilities

Figure 4-21 : Overall Consensus for RQ3

Theme Code	Research Question Theme	Overall Frequency	Overall Responses	Positive Responses	Consensus Percentage
RQ3A	Security through resolving vulnerabilities	**25**	7	**6**	**85.7%**
RQ3B	Relationship between defects & security	**25**	7	**7**	**100%**
RQ3C	Impact of Experienced Developers	**25**	6	**6**	**100%**
RQ3D	The role of security training	**25**	5	**5**	**100%**

Table 4-14 : Consensus Details for RQ3

4.4.3.4 RQ4 Customer's Control of Security

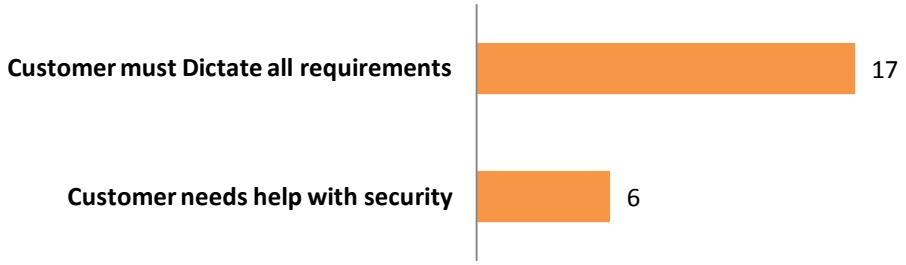


Figure 4-22 : Overall Consensus for RQ4

Theme Code	Research Question Theme	Overall Frequency	Overall Responses	Positive Responses	Consensus Percentage
RQ4A	Customer must Dictate all requirements	**24**	18	**17**	**94.4%**
RQ4B	Customer needs help with security	**24**	6	**6**	**100%**

Table 4-15 : Consensus Details for RQ4

4.4.3.5 RQ5 Knowledge Dissemination and Diffusion

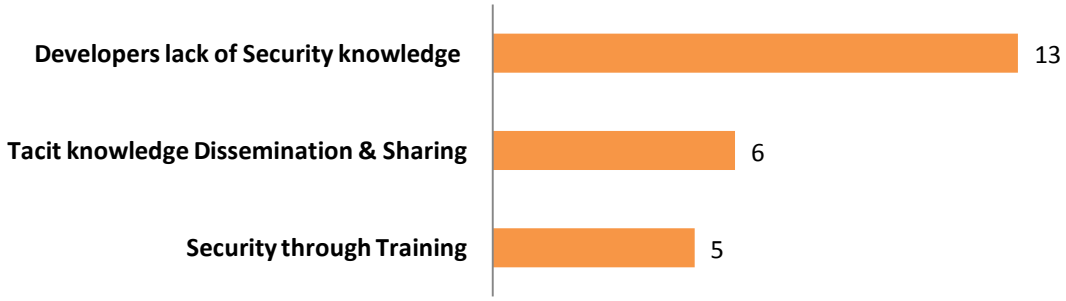


Figure 4-23 : Overall Consensus for RQ5

Theme Code	Research Question Theme	Overall Frequency	Overall Responses	Positive Responses	Consensus Percentage
RQ5A	Developers lack of Security knowledge	**25**	13	**13**	**100%**
RQ5B	Tacit knowledge Dissemination & Sharing	**25**	6	**6**	**100%**
RQ5C	Security through Training	**25**	6	**5**	**83.3%**

Table 4-16 : Consensus Details for RQ5

4.4.3.6 RQ6 Dedicated Security Iteration

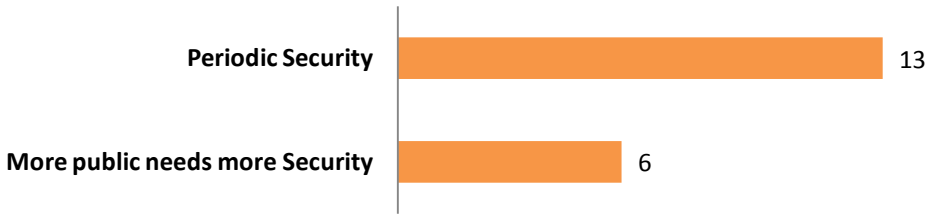


Figure 4-24 : Overall Consensus for RQ6

Theme Code	Research Question Theme	Overall Frequency	Overall Responses	Positive Responses	Consensus Percentage
RQ6A	Periodic Security	**21**	15	**13**	**86.6%**
RQ6B	More public needs more Security	**21**	6	**6**	**100%**

Table 4-17 : Consensus Details for RQ6

4.4.3.7 RQ7 Testing and Verification for Security

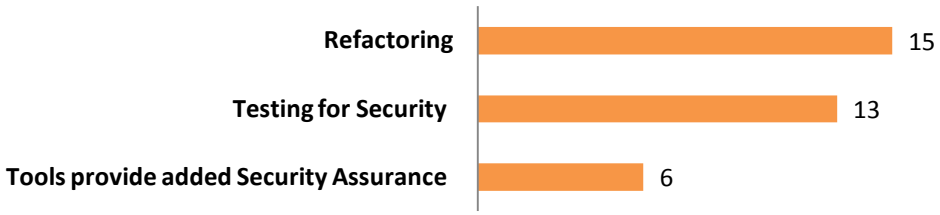


Figure 4-25 : Overall Consensus for RQ7

Theme Code	Research Question Theme	Overall Frequency	Overall Responses	Positive Responses	Consensus Percentage
RQ7A	Refactoring	**40**	21	**15**	**71.4%**
RQ7B	Testing for Security	**40**	13	**13**	**100%**
RQ7C	Tools provide added Security Assurance	**40**	6	**6**	**100%**

Table 4-18 : Consensus Details for RQ7

4.4.3.8 RQ8 Documentation and Security

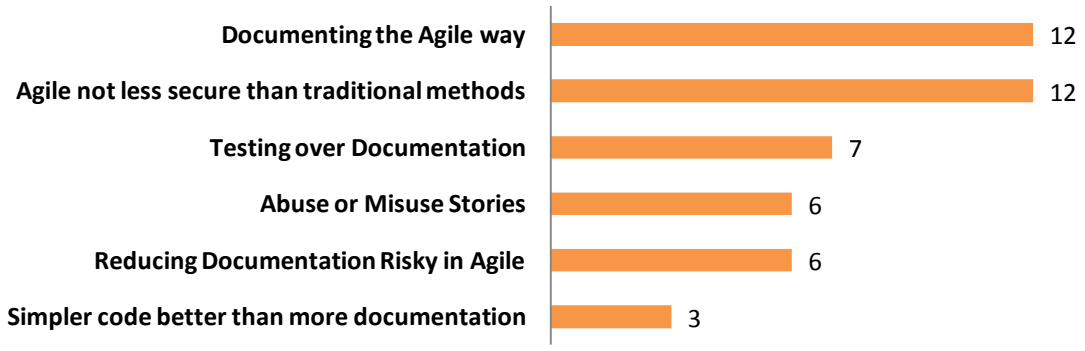


Figure 4-26 : Overall Consensus for RQ8

Theme Code	Research Question Theme	Overall Frequency	Overall Responses	Positive Responses	Consensus Percentage
RQ8A	Documenting the Agile way	**54**	13	**12**	**92.3%**
RQ8B	Agile not less secure than traditional methods	**54**	12	**12**	**100%**
RQ8C	Abuse or Misuse Stories	**54**	12	**6**	**50%**
RQ8D	Reducing Documentation Risky in Agile	**54**	7	**6**	**85.7%**
RQ8E	Testing over Documentation	**54**	7	**7**	**100%**
RQ8F	Simpler code better than more documentation	**54**	3	**3**	**100%**

Table 4-19 : Consensus Details for RQ8

4.5 Interpretation of results

This section of the study attempts to answer the question of "What were the lessons learned?" which try to capture the essence of this idea (Lincoln and Guba 1985). What we are looking for here is "a meaning derived from a comparison of the findings with information gleaned from the literature" (Creswell 2009) and our empirical findings. This way we can either affirm the currently accepted information or we could reject them citing new discoveries that we made at this point. To begin with, we will proceed with a systematic discussion of all the major themes discovered as part of our analysis of the interview data.

4.5.1 RQ1- Combining Security and Agility

In order to be able to seamlessly combine security and agility and evaluate the suitability of security mechanisms for use in Agile, the best and most effective practice turned out to be RQ1A the inclusion of the security engineer (18.1%) even though it came at a relatively higher cost than the second most agreed upon theme which was RQ1B developing software with security in mind (16.6%) which did not include the added cost of having to hire a dedicated security engineer for the team. RQ1C involved the use of security controls in mitigating the need for security but the fact that Standard Security Controls are not yet present it could make it relatively more costly than other more popular ways of adding security to Agile projects. RQ1D underscored the fact that

while there was no formal security engineer, such a skill was still sought after by various team members and more often than not such a person was unofficially picked to help on certain security matters. While this theme is not presenting a solution or a way to combine security and agility, it reinforces the need for RQ1A which is to have a formal dedicated security engineer present. RQ1E came as the only negative set of responses to the question in which the various aspects of the combination was scrutinized. While there are indeed risks and challenges associated with any such undertaking as combining security and agility in a seamless manner, the less than 10% negative response (in terms of risk, not how it could not be done) to this idea shows that there are no major consensus on how and why this method could not be realized. RQ1F shows how the relative experience of the people in the development team has a direct impact on the overall security of the project and how this in and of itself could be used as a way to increase the security of the resulting software for a given project but since relatively few respondents mentioned it as an important factor (9.4%) we can conclude that this pales in comparison to the combined agreement of the benefit of using a security engineer combined with the need for such a person (RQ1A+RQ1D) which is 28.3%.

The addition of the security engineer was simultaneously the most popular and the most agreed upon and contested theme of the interviews. While practitioners recognized and agreed that the addition of the security engineer could benefit the team and the resulting software in significant ways in terms of security they voiced concern in terms of cost as well as the relative impact that the addition of a new member to the team could have on the overall velocity of the project. The security engineer could substantially improve the team's overall knowledge of security over time through sharing their knowledge and experience with the rest of the team. They could bolster the security assurance argument of the agile project and mitigate the risks voiced by some researchers and practitioners regarding the Agile principles in meaningful ways (discussed earlier as a part of the literature review chapters). With everything considered, the addition of the security engineer to the team appears to be the best overall practice that an organization can do in order to increase the security assurance argument for their respective clients as well as for their internally or externally deployed software.

The second most popular theme involved the consideration of security at each step of software development without having to formally be required to follow certain steps. This approach helps agile in a sense that the process of agile is not changed or modified as a result of this increased consideration and it will ultimately result in a much more secure software and an increased security assurance argument for the project. This however required increased levels of knowledge and expertise from developers and such people are relatively hard to find in practice compared to a regular, average developer. However, since agile is known to allow tacit knowledge to disseminate throughout the team all developers do not have to be of such high knowledge level and expertise and therefore could train other less experienced developers as they are doing their normal day-to-day activities. This is a lower cost alternative to having a security engineer present permanently as discussed earlier.

Some interviewees were enthusiastic about the use of security controls within software as an alternative to either a security engineer or any other proposed method for adding security to the resulting software. The addition of these controls alleviates the developer from having to come up with and implement many routine security tasks within software that would otherwise need to be implemented manually and the upside is the lower cost and time that could be used on other tasks. The downside of this approach could be that there are no standard security controls at this point but only some proprietary security controls are provided by software security firms. Surprisingly, this solution was not proposed through the literature and only became apparent after a few interviewees expressed in multiple places that standard controls could be a useful, low cost alternative to other more involved and popular approaches.

While the informal security expert is not cited or proposed as a solution to the problem of adding security into Agile, many interviewees mentioned the existence of an informal security expert who could help them on such issues within their reach in the organization. The experts ranged from Software Engineers to Architects to System Engineers and team leaders who were known for their experience and knowledge on such issues. This represented the bare minimum that they needed in terms of security of the software but in a small way contributed to the overall security assurance for the project. This clearly shows that the existence of a security engineer or expert is indeed required to a certain extent but the actual measure of the degree of such a need and their effectiveness as of this writing is unknown and ambiguous. A take away point from this

theme is that the choice of the informal security expert was guided in part by the relative knowledge and experience of the team member who was designated informally to act as a kind of security consultant.

The theme of Integration risk involves all the relative issues and risks that could come up as security is integrated into Agile. While this was a negative theme where the participants discussed drawbacks and other impacts of combining security and agility, the actual frequency turned out to be less than 10% of all the respondents. What's more, about half of the interviewees raised some kind of issue that could come up as one would combine security and agility for a given project. For example, manager encouragement and support of developers in terms of security is cited as very important in a successful integration. However, since most managers are not concerned much about security for the project their cooperation and setting of the tone for the team in terms of security is a risk that need to be addressed before a successful integration could be accomplished.

RQ1F is another emergent theme that was not mentioned explicitly throughout the literature involves the developers within the team. While the process of software development involves more than just the developers alone, they are the ones who actually write the software and inevitably introduce the vulnerabilities into the code as well. Knowing which developers are more effective and secure has been given as a focus area by many interviewees both directly and indirectly. The overall consensus seems to suggest that the experience and the level of knowledge of developers directly impacts the resulting software as it relates to security in addition to quality and functionality. Not only they could identify more issues earlier but they could also solve problems and security issues easier than any other member of the team when it comes to increasing the security level of the software under development. While it may not be practical to hire only experienced developers, actually how many are needed within the team to make a lasting impact could be investigated further.

While RQ1G was discussed in the literature as a relatively important issue for security in agile, the data gathered from practitioners indicates less emphasis on its importance but more emphasis on the manner in which planning is done in Agile. Some participants argued that the planning process could include steps towards increased security but the exact manner of changes and how long the process should take was not obvious from the discussions. One take away point from this theme is that the majority

of the interviewees view the planning stages of Agile and the lack of upfront design, such as in traditional methodologies, does indeed impact security of the overall software although the exact measure of the impact is as of yet unknown.

While the static analysis tools and similar software programs indeed help in finding threats and vulnerabilities in the code, the actual practical value of using them in the project is relatively unknown since less than 30% of the participants even mentioned static analysis tools. From amongst those who did mention the tools the consensus was positive overall but at the same time they echoed the position that was raised in the literature which was the fact that the static analysis tools are not mature enough to be able to discover many forms of threats and vulnerabilities. Overall static analysis tools do provide added security assurance but not enough to achieve good enough status amongst academic and practitioners alike especially when it comes to using them as a major factor in combining security and agility. A more dedicated discussion of tools is forthcoming in other themes as it relates to other research questions.

RQ1I was only mentioned because some people felt that the practice of code reviews could be used as a way to augment and increase the relative security assurance of the software but by itself this practice is in no way intended to be used as a standalone solution such as the inclusion of the security engineer to the team. Furthermore, in order for this practice to be effective the reviewer must be more experienced and knowledgeable than the other developer who wrote the code. For example, code review during code check-in time, a review by an informal security expert or a more experienced developer and/or software engineer could help in pointing out certain defects and vulnerabilities which will also result in increased awareness and knowledge of security for the less experienced developer as well.

4.5.2 RQ2 Change Agile Practices for Security

To be able to better understand the overall issue of whether or not changing agile for the sake of security really needed and to see what kinds of projects could use security more than others, the responses gave us a relative scale to interpret the findings. Out of all responses to the questions that were related to this research question, 59.7% of the responses agreed on how the changes to Agile were not really necessary from 4 overall points of view (themes). From those who were in favor of changes that agreed with no changes to Agile being necessary the themes involved were RQ2A, RQ2C,

RQ2D, and RQ2E. We also gained insight into what projects could use security more than others and it came to whether or not the project was being used internally within the organization or in a more public setting open to scrutiny by everyone.

RQ2A was one of the most discussed topics when it came to changing Agile for the sake of security. Some participants stressed that the existence of security was independent of either methodology (Traditional or Agile) that should be irrelevant whether security should be included or not. However, the majority of the participants agreed that Agile was not less secure if not better in terms of security and some went as far as claiming that Agile was in fact better in terms of quality and adapting to changes which allowed it to find and resolve more vulnerabilities faster and earlier than traditional methodologies. This coupled with the fact that traditional methodologies by design are not open to change provides the insight which is to say that Agile overall does a better job of developing more secure software than waterfall even without the inclusion of extra security focused steps and therefore does not need to be extended for the sake of security.

Awareness of security in Agile turned out to be almost as important as the discussion of Agile vs. traditional methodologies. However, this was one theme that almost every participant talked about which underscores the need for everyone involved in the project to be at-least aware of security when they design and develop software. If security is important in any way to the project, then stakeholders must think about risk assessment and analysis, elaboration phases, upfront planning and investment, policies and guidelines geared towards increasing every team member's awareness of security and allow them to voice their concerns freely and openly. This suggests that more focus on security is needed that could indeed be accomplished through the addition of certain steps and/or practices to Agile.

RQ2C was an emergent theme which discussed the accelerated timeframes of Agile and their effects in practice especially for security. While some participants characterized scheduling issues as irrelevant to security the majority of the respondents agreed that the accelerated timeframe often employed in Agile projects tended to be too short and this had a cascading effect on all aspects of development and especially affected security because security is considered as a secondary objective to functionality which is a primary concern of stakeholders in Agile. Since security is classified as a cross cutting concern and often results in non-functional requirements there is less

perceived value in its implementation and therefore the added schedule pressure often forces people to cut corners in this area. This is not to say Agile as a methodology is driving this issue but it is the people who choose too short of a timeframe per iteration that results in too much pressure which results in more risk than is warranted. This theme underscores the need for people in charge of creating plans and schedules to be more open and conscious of security related issues and dedicate more time per iteration to address other issues in addition to functionality.

RQ2D was an emergent theme (not mentioned in the literature) that many participants discussed and some even had direct experience on which provides an insight into how internal projects are handled and undertaken in terms of security. Many participants claimed that they did not focus so much on security because of the fact that their software was only deployed internally within a controlled and trusted environment which was either already being protected through the infrastructure or through other means such as various control mechanisms deployed throughout the organization (Access Control). They also mentioned a dedicated team sometimes being present that was handling security issues within the organization and that the software was less prone to threats and vulnerabilities as a result. This further shows that for some projects the security aspect simply is not a concern because either it is being handled at a different layer or the trusted and private nature of the deployment environment minimizes any chances of exploitation and misuse.

RQ2E was a theme directly taken from the literature which was also talked about by some participants. They basically agreed with the premise that no change was indeed necessary to Agile and that the addition of security should be handled as much as possible through unobtrusive means that do not attempt to modify or extend Agile with additional steps and/or practices that may not be needed or required by all projects. This is very important because it clearly shows that there are certain numbers of people in practice who feel they need to keep Agile the way it is in order to keep benefitting from its advantages.

4.5.3 RQ3 Security Issues and Software Vulnerabilities

In order to gain insight into the extent to which security issues stem from vulnerabilities and/or defects, we have found that there were certain factors that led to increased vulnerabilities and weaknesses of software as it related to the methodology and certain practices. Also, on the same note, the relative experience and knowledge of the developers were also discussed that accounted for 44% of the responses to this research question. However, since the number of respondents that discussed these topics accounted for only 5.9% of all the responses, we decided not to further investigate the issues surrounding this research question.

RQ3A was another emergent theme that was brought forth as a result of the link between vulnerabilities and their possible causes. Here there are multiple reasons cited such as complex coding, lack of a security layer, little to no testing, and no clear goals being set for the project which would contribute to increased vulnerabilities and less secure software as a result.

RQ3B was directly related to the research question which inquired about the relationship and the extent of the relationship between bugs and security. While there was many instances were bugs were being classified in terms of various criteria, security was mentioned or considered in only a few places in terms of high-severity or the seemingly critical risk it posed to the software. There was also a mention of risk assessment of bugs but only as a suggestion for improvement not as an established practice. This further goes on to show how security related metrics are not being considered as part of many projects.

RQ3C is similar to RQ1F with a notable difference in cases where the experience of the developer directly affects vulnerabilities and weaknesses of the resulting software and tries to gain further insight into what kind of impact more experienced developers make in this case. The first finding was that lack of experience could be a major cause of vulnerabilities and second finding suggests that more experienced developers do tend to avoid certain security issues and vulnerabilities early on. They tend to be more aware of security issues and more careful when coding and developing software than new developers.

For RQ3D, in cases where the knowledge of the developer directly affects vulnerabilities and weaknesses of the resulting software, the role of training becomes

important and this theme tries to gain further insight into what kind of impact more trained developers could make in this case. It was found that training of developers could be a major factor in reducing vulnerabilities and that more knowledgeable developers tended to avoid certain security issues and vulnerabilities more than others. Even though this theme was deemed important it did not come up as part of the interviews frequently enough to make it a more substantial issue than we were expecting it to be.

4.5.4 RQ4 Customer's Control of Security

From the results and answers to this research question it is clear that the customer should indeed have the authority to dictate all requirements for the project including security, we see no further need to continue our investigation into this issue because less than 6% of the respondents discussed it as an important factor overall and we feel time could be more well spent looking into more important and major themes and research questions. The following paragraphs briefly discuss each theme under this research question.

RQ4A which was also mentioned in the literature as a significant issue found the most positive responses for the research question involving the customer. Most of the participants agreed that the customer should indeed dictate all requirements including security for a given project even though the topic of security might not be an obvious consideration from the beginning of the project. This accounted for 75% of all respondents that had this overall opinion through direct experience of their own. Basically what it comes down to in this case is the fact that if the customer cared enough about security of the software then they would have to demand steps to be taken to focus on that aspect during development which will give them more value in terms of security over the case where they don't explicitly ask for security and have the developers and other project stakeholders worry about that aspect.

RQ4B, while related to the earlier theme with respect to the relative need for the customer to dictate security requirements, does recognize and take into account the fact that the customer might not be in the best position to be able to decide whether or not security should be an important consideration for the project. Therefore the customer needs the help of someone within the project to advise them on the security issues so they can make an informed decision. However, the exact nature of who should be the

advisor and how they should do this is not clear at this point but it does hint at the need for a security expert (formal or informal) which could prove beneficial in this case.

4.5.5 RQ5 Knowledge Dissemination and Diffusion

Towards increasing our understanding of issues and discussions surrounding the question of the effects of knowledge dissemination and diffusion, the apparent lack of developer's security knowledge was most prominent (52%). This means that the number one factor in helping spread the knowledge of security within the team revolves around the relative knowledge and expertise of the developers. The second most contributing factor turned out to be pair programming (24%) which is popular in some agile methods and used as a knowledge dissemination tool to increase the relative knowledge of security and other skills within the team. Almost equally important was achieving security through non-explicit forms of training that could result in increased security for the project. Overall, there were not enough respondents that touched on the subject to warrant more investigation of this research question and therefore we will leave the matter at this point to focus on other more popular topics under investigation. The following is a brief discussion of the themes found for this question.

While a similar issue was raised in some parts of the literature, there was not enough research done on the topic to make RQ5A a significant or substantial issue. However the emerging nature of this theme indicates that practitioners do indeed think that developers in general lack security related knowledge and expertise and therefore may need to be trained and/or mentored in some way to increase their knowledge and awareness of the issues surrounding the security aspects of the software under development. From the multitude of issues that have been raised we can cite the lack of confidence of the developer which adds risk to the project to the lack of motivation and testing for security cases and instances are amongst the responses that were mentioned for this theme.

While not many people touched on RQ5B, a few respondents agreed that pair programming was an effective form of knowledge dissemination within the team which could also be used to spread the knowledge of security around. Pair programming has been recognized as effective in agile and could still be effective for security when there is another more experienced developer and/or security engineer present who is able to advise members on security related tasks and issues.

The general topic of security training (RQ5C) has been discussed sparsely in the literature and the same is true for the interviews. The fact that developers who would like to be able to spread and share their knowledge of security need to have been trained is known and acknowledged by a number of participants but overall not many interviewees mentioned this explicitly although they implied the same in other ways and different contexts which makes the overall theme of training rather important. For example, one participant mentioned that Microsoft had through their security program increased the security of all their software and the gains from this endeavor contributed to less vulnerabilities and zero net cost for the company.

4.5.6 RQ6 Dedicated Security Iteration

In order to be able answer the question of how beneficial are dedicated security iterations and under what circumstances should they be initiated, we discovered that establishing periodic security was beneficial overall regardless of the circumstances (71.5%) and especially important in projects that were being deployed publically and exposed outside of the organization (28.5%). This research question garnered very clear answers in both aspects of whether or not to conduct periodic security iteration as well as the overall circumstances that one should consider when they would like to proceed with a project. While we could delve much more deeply into the topic and investigate further the relatively small number of responses indicates that we have gained as much as we could on the question and we will move to more popular and important questions for further research and analysis.

RQ6A had much support from the literature and was mentioned many times in different settings but many of the proposed solutions were focused on security at the end of the project. Our investigation into the issue resulted in the rebuttal of doing security only at the end of the iteration and/or release and instead focused on achieving security throughout the development lifecycle. While a few interviewees said that indeed doing security at the end of the project could work on certain projects the vast majority sided with the viewpoint that security throughout the project was more effective and better overall. One interesting fact that came up through the interviews was the relative popularity of hiring a security consultant (RQ1A) for short periods to comment and audit the system and project in terms of security. This indicates that the existence of a

security expert and/or engineer is indeed desired and needed for many projects but the relatively high cost of hiring such a person full time made the option of a part time security consultant more attractive to many companies who thought conducting a security iteration every now and then would be beneficial.

RQ6B was a theme that emerged as a result of the fact that projects that are deployed publically and outside of the organization usually require an increased level of security assurance to be effective. Towards achieving this end, the projects may need to employ periodically certain security focused steps and/or practices within an iteration in order to increase the security of the software being written. This theme answers the part of the question which asks under what circumstances increased security for projects is warranted.

4.5.7 RQ7 Testing and Verification for Security

In order to find out to what extent do the testing and verification tools provide additional security assurance in addition to quality assurance, we have found that 52.5% of the respondents agreed that Refactoring could contribute rather significantly towards the increased security of the resulting software if it's done throughout the project. Furthermore the results from testing for security (32.5%) show how strongly testing and verification mechanisms can be used to prove the relative security of the software and contribute to increased security assurance argument for the Agile project. The subject of tools while acknowledged as effective and beneficial (15%), does not go far enough as of yet in terms of providing solid assurances that the resulting software would be secure but they could be used as a good augmenting practice to add to other approaches to further increase the security of the resulting software under development.

Refactoring (RQ7A) turned out to be one the most hotly contested topics which also was mentioned throughout the literature. While few proponents of the use and benefits of refactoring cited the definition of refactoring as irrelevant to security from a purely theoretical perspective, many did acknowledge that in practice it was much harder to cling to any notions of theoretical definitions and indeed refactoring was being effective in simplifying the code base and in making testing and verification easier which resulted in increased quality and security of the resulting software. They went on even further and claimed that refactoring should not be done only at the end of the

project as some throughout the literature had proposed and claimed that it should be done at all times in order for it to be most effective overall.

Security focused testing (RQ7B) was another literature inspired theme which turned out to be also popular amongst practitioners. They cited the amount or numbers of test cases could be used not only prove that certain security focused steps were undertaken, they also claimed such tests could be used as a part of the acceptance testing which is usually done towards the end of the project. The testing mechanism was chosen as a proving ground for security as well as quality since the two are related in some aspects. While when and at what point the actual testing to be conducted was not immediately clear, the overall recognition of security focused testing was recognized in contributing substantially towards increased security assurance argument for the project under development.

While the existence of tools and their relative benefits (RQ7C) have been cited in the literature, the overall consensus on how much benefit tools can bring to a given project have been a point of contention. From amongst the participants, many cited that while tools in general do indeed contribute to added security assurance for the project, they are not enough to be able to stand alone and take care of all issues. This coupled with the fact that there are not many mature tools available for practitioners further solidifies this overall conclusion. Another factor is cost which has also been mentioned by a number of interviewees which affects the tooling adoption throughout the industry.

4.5.8 RQ8 Documentation and Security

To be able to answer the question of does the reduced documentation in agile projects lead to less secure software even with the advent of CASE tools, we have discovered that documentation while it does no longer exist entirely in the form of written manuscripts, there are many other forms of documentation, some automatic, and others replacement practices have begun to fill in the blanks. For example, as a part of documenting the Agile way, practitioners used a various number of tools and practices aimed at both reducing documentation and adding new forms of more efficient and effective approaches that could benefit the project in multiple ways. Documenting the Agile way turned out be the most popular approach that developers and other stakeholders chose to follow (24.0%) and almost as important (also 22.2%) was the fact that they regarded Agile as not less secure than traditional methodologies which gave

them the confidence to continue innovating and finding new and creative ways to develop software faster. While Misuse and Abuser stories were acknowledged as a good way to look at security issues, the manner and the placement of those stories was a point of contention that we intend to further investigate. A surprising theme that emerged from these interviews was the fact that many practitioners were choosing to forgo documentation in favor of spending the time in creating test cases (12.9%). This was both added to the overall security assurance and was used as part of the acceptance criteria for the software under development which was both creative and effective.

RQ8A is partly discussed in the literature although the methods and the details that the practitioners discussed here might not be widely known or used within all agile teams it does show that while Agile does indeed have reduced documentation, the overall practice has not impacted the security that much. This is because the way each part of software works has been changed to more effectively fit in with the tools and/or additional practices that promote the same effect but with less manual effort. For example CASE tools have been cited to help create automatic documentation to and from the code but also have been known to be as of yet not mature enough to render traditional manual documentation obsolete. There is also a trend towards only documenting what's necessary as opposed to every aspect of the software that in the eyes of some can go out of date very quickly and become ineffective and in some cases counterproductive.

RQ8B is another emergent theme in which Agile is considered to be just as secure if not more secure than traditional methodologies' documentation practices. As it relates to both, while heavyweight security mechanisms such as Common Criteria are cited to work better with traditional mechanisms, some practitioners claimed that they had successfully implemented CC into their Agile projects without any major issues. Furthermore, the relative success of Agile processes in replacing manual documentation with other more effective practices has resulted in incremental gains in reducing time overhead. This means that Agile could effectively produce comparable software to traditional methodologies without the drawbacks of heavyweight processes that require tremendous manual documentation and design upfront.

RQ8C was widely discussed throughout the literature and offered as a possible solution to the issue of adding security to Agile projects. It was also a divisive issue amongst practitioners although not all agreed that it was a good idea in its entirety.

Some claimed it was good practice because it could lead to better security test cases but others while acknowledging abuse cases were a good idea they did not want it as a standalone story and only as a part of a regular user story under a security related section so it does not add too much overhead to Agile and the normal day-to-day practices of Agile. This means that indeed some kind of security related aspect should become a part of normal user stories. However, it should not overshadow or hinder the actual process in any way because the pace of development is already accelerated enough that adding extra stories, which do not functionally add any extra value or assurance to the project, is not desired.

While RQ8D has been known and discussed throughout the literature, the actual empirical evidence on what the practitioners think and do has been relatively obscure. Indeed a few practitioners agreed that at least some critical amounts of documentation was necessary to the project but the actual degree and measure of how much documentation was enough is put on the shoulder of each team to decide given their circumstances and the type of software they are developing. While this risk is indeed present in practitioner's minds, it is in no way a point of problem and many have found the correct balance between development and documentation to be able to move forward and create effective software.

RQ8E was emergent and a pleasant surprise to know that practitioners had found a way to write many test cases instead of writing documentation that would later on be turned into test cases. This has had a wonderful effect of being able to replace documentation altogether and be able to be used as a basis for faster upfront testing of software to make it more effective and discover issues earlier, reduce defects, and generally be used in place of design in a traditional setting. Because of this, developers now have to write test cases before they even start coding which forms the acceptance criteria for the project as a whole. What this means is that practitioners have found a way to completely replace certain types of documentation with more dynamic and effective practices that add more benefits to the project than manual documentation alone.

While RQ8F has been known in the literature some developers explicitly mentioned it and acknowledged that indeed simple code which is a part of Agile does not need to be documented which result in increased security of the software and reduced vulnerabilities and bugs.

4.5.9 Overall picture of the research questions

It should be evident by now that a reoccurring theme within the discussions of all the aspects that was discussed above hints at various aspects and points of view to the benefits of having a dedicated and/or periodic security expert. The details on where this expert comes from and at what level they could be more effective is the question of the setting and the overall aims and security requirements for the project specifically and the organization as a whole. To illustrate this point, Figure 4-27 depicts various places where security personnel could be added that could have various effects in terms of overall impact on security for the agile team and the resulting software more specifically.

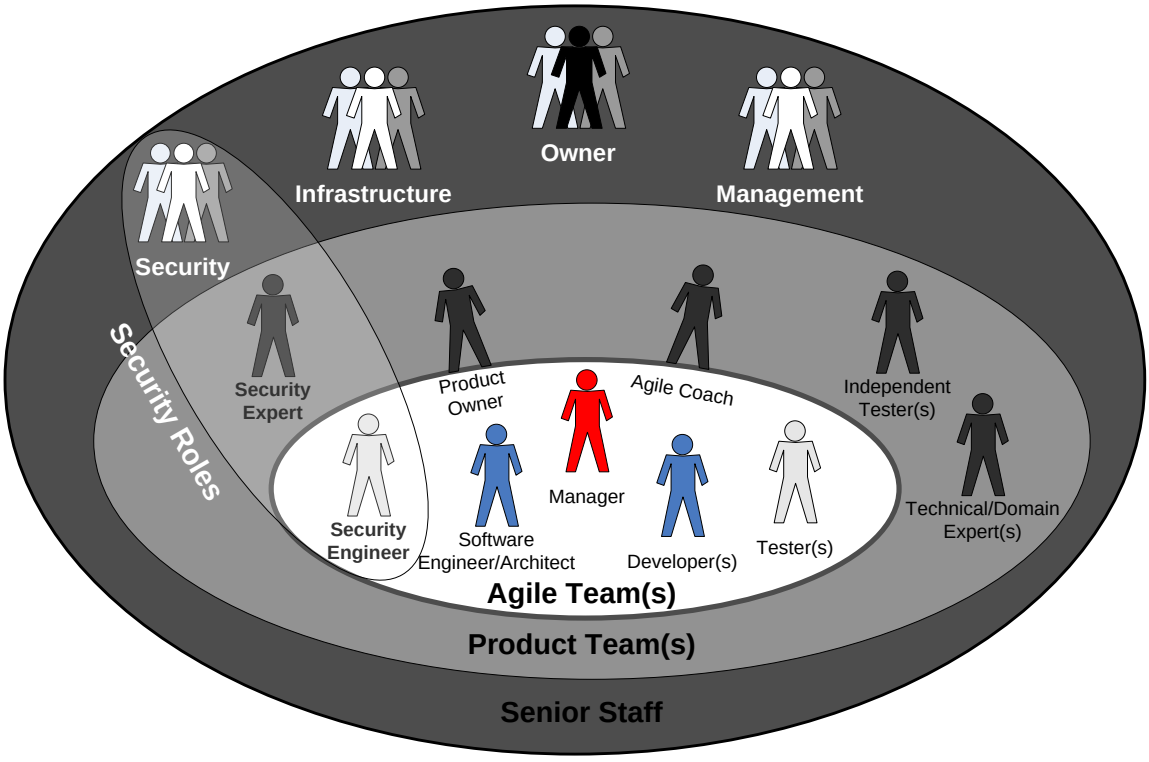


Figure 4-27 : Security Roles at various Levels within the Organization

Aside from the addition of a security engineer the relative impact of more experienced developers, their education, knowledge level, and sharing of information between developers was another common and emergent aspect that was discussed from many perspectives and points of view throughout the interviews. It is the

developers who need to be most aware of security issues when developing software and its their education and training that has the most lasting and effective impact on security overall. Even the proposed security expert is not going to be effective that much if they do not come from a background of having been developers. In other words, the best overall solution in our opinion, and from the results of all the interviews, analyses and interpretations is that having a relatively more experienced developer trained in various security aspects act as a security engineer to the team would provide the maximum effect to the resulting security of the software and the most lasting impact on the Agile team in terms of knowledge dissemination and diffusion, and other aspects. The following figure 4-28 shows a sample scenario where a number of security solutions including the security engineer or expert could be integrated into Scrum.

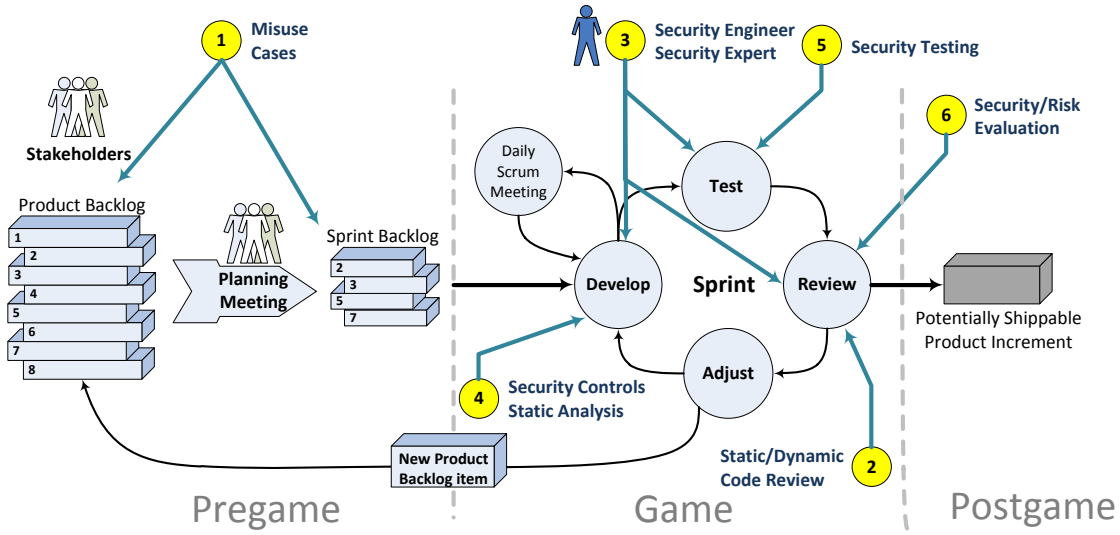


Figure 4-28 : Adding Security Practices to Scrum

4.5.10 Developing hypotheses

As we alluded to in earlier remarks in the interpretation of research questions, we chose to develop hypotheses for the research questions that at-least had more than 9% occurrence which means they were discussed and mentioned around 10% or more of the time in the interviews. This allowed us to focus the research on the most important empirical aspects to continue our investigation on. The following table shows the overall occurrence percentage for each research question.

Research Question	Research Question Description	# of Codes	Overall Total	Occurrence Percentage
RQ1	**Combining Security and Agility**	**138**	419	**32.9%**
RQ2	**Change Agile Practices for Security**	**92**	419	**21.9%**
RQ8	**Documentation and Security**	**54**	419	**12.8%**
RQ7	**Testing and Verification for Security**	**40**	419	**9.5%**
RQ5	Knowledge Dissemination and Diffusion	25	419	5.9%
RQ3	Security Software Vulnerabilities	25	419	5.9%
RQ4	Customer's Control of Security	24	419	5.7%
RQ6	Dedicated Security Iteration	21	419	5.0%

Table 4-20 : Overall Occurrence of the Research Question

For each of the 4 top research questions in terms of overall occurrence, the hypotheses for each theme assigned to the research question was generated if that theme had more than 9% occurrence as part of that research question. We would investigate further all of the themes and the research questions but due to the limitations of time and resources, and the fact that we are looking to investigate the most important security issues in agile empirically, the opinion of practitioners is important in this case in determining whether or not to continue investigating a theme or not. Therefore, we set the threshold at 9% or more because it indicates a rather considerable focus by the practitioners on a particular sub-topic or theme and allows us to focus more on these for the later stages of the research.

We ranked the research questions by occurrence because the investigation includes both positive as well as negative responses which more clearly show the relative interest of practitioners on the topic rather than only positive responses which does not show a full picture. If we only took the positive responses, a cross section of interested practitioners on the topic would have been unaccounted for in the next phase of the research which may negatively impact the quality and the direction of the investigation.

Research Question	Hyp.	Hypothesis
Combining Security & Agility	H1a	The addition of the dedicated security engineer is suitable for inclusion into Agile.
	H1b	Writing Software with Security in mind is suitable for inclusion into Agile.
	H1c	Use of security controls is suitable for inclusion into Agile.
	H1d	The use of more experienced developers is suitable for inclusion in Agile.
Change Agile Practices for Security	H2a	There is no significant difference between Agile Methodologies and Traditional Methodologies with respect to the resulting security of the software produced.
	H2b	Adding practices aimed at increasing awareness of security is necessary for integration with Agile.
	H2c	Accelerated timeframes and schedules of Agile contribute to less secure software.
	H2d	Internal Agile projects can ignore the security aspects of the software.
Testing & Verification for Security	H7a	Refactoring impacts the overall security assurance of the software.
	H7b	Security Focused Testing positively impacts the resulting security assurance of the software.
	H7c	Security Tools positively impact the resulting security assurance of the software.
Documentation and Security	H8a	Agile's alternatives to documentation impact the resulting security of the software.
	H8b	There is no significant difference between Agile's documentation methods and traditional documentation with respect to the security of the resulting software.
	H8c	Misuse or Abuser Stories impact the resulting security assurance of the software.
	H8d	Agile's testing is more effective than documentation on the resulting security of the software.

Table 4-21 : List of Generated Hypotheses

4.5.11 Outcome of the Qualitative study

The results of qualitative study and the emerging hypotheses we mentioned earlier will be tested through the quantitative part of the study and the insights gained through the identification and analysis of themes will form the basis of the questionnaire that we intend to develop as part of the more comprehensive empirical study of the consensus on major topics found.

4.6 Sampling and Design Issues

According to Boyatzis "Thematic analysis is sensitive to the quality of the raw data or information. Therefore, sampling decisions not only affect but to a large extent determine the degree of reliability and validity attainable" (Boyatzis 1998). The quality of the criterion selection and the sampling will determine the quality of the code and subsequently the quality of the findings. Our criterion for targeted sampling consists of multiple criteria which includes roles of various groups of stakeholders within a software development team such as Developers, QA, Managers, Security Experts, Consultants. The sampling frame consisted of predominantly developers including programmers, architects, and software engineers, some Managers, Security Experts, QA/Testers, and Consultants. Sub-samples include 10 Developers, 6 Managers, 3 Security Experts, 4 Consultants, and finally 3 QA/Testers. These individuals were at the time working in various organizations that develop software and since they could not be reached in their natural work environment, they were pursued in Conferences, periodically held professional meetings, at various lectures and events, and at other similar events. An important aspect to note is that there is some overlap between the roles of subjects for example some developers were also managers and some other developers were also consultants and we elected to count both roles for that individual.

In total, 15 in-depth interviews with 16 subjects were conducted and some comments from 4 additional subjects were gathered for the study. After conducting these interviews we became confident that we had interviewed enough individuals because the nature and types of answers that they were providing us was the same as other previously interviewed subjects. At the same time, we had achieved the goal of being comprehensive in sampling by obtaining information from various roles within

the agile team that we were targeting for the interviews which according to Boyatzis (1998) gives the most complete picture of the unit of analysis which in our case is the security issues within the Agile team. After interviewing the last 5 individuals we were assured that there was not much more new information we could gain from continuing the interviews with more people and therefore we stopped interviewing individuals after the 15 interviews were concluded.

4.7 Reliability

The aim of establishing reliability is to be able to show consistency of approach in how the process was established conducted (Gibbs 2007). According to Yin, the procedures of the study need to be as detailed as possible in order to increase the understanding of the other researchers and readers about how the qualitative study was conducted (Yin 2003). As such, many documents such as transcripts must be double checked, coding scheme is consistent, and definitions are clear and concise (Gibbs 2007) and in some cases analyzed (cross-checked) by more than one person and results compared (Boyatzis 1998; Creswell 2008).

4.7.1 Reliability Threats and Biases

4.7.1.1 Subject or participant Error

We took steps to make sure all participants are properly chosen based on their background and their relative work experience in the area of our research and a variety of roles and responsibilities were chosen for the semi-structured interview in order to minimize any bias by one particular group over an issue. We chose the format of Semi-structured interviews so that the questions could be posed from different perspectives and different people while the answers would contribute to the same basic research question.

4.7.1.2Subject or participant Bias

We took extra measures in keeping the anonymity of the respondents for their interviews. We went as far as anonymizing their organizations and erasing any personally identifiable information that might possibly be used to trace back to them in any way. The developers might be biased towards their role as instrumental in the process of adding security to the software therefore they might have an elevated

sense of ownership and might want to exaggerate their achievements in order to appeal to a higher sense of technological ability. Our analysis took this and other such biases into account and only considered the responses credible if they were corroborated by other participants with different roles from different organizations in various countries and cultures and genders.

4.7.1.3 Observer Error

We recorded interviews in order to minimize observer error (in transcription) and later went back to the recordings and summarized the findings as accurately as possible in order to minimize any possible observer errors. On the same note the structured format of the semi-structured interview provided us with an added assurance that the same exact wording for each question would be asked from all respondents therefore reducing the possibility of this threat to reliability.

4.7.1.4 Observer Bias

To ensure we have interpreted the replies in an unbiased way, we have used the analysis software package Nvivo in order to provide a more objective method of interpretation of our results in order to minimize observer bias in our results. We also send the results back to the participants if/when they asked us to do so in order to confirm their responses are still valid and accurate and they were given another opportunity to change/modify any previous answers.

4.7.2 Interrater Reliability

Interrater reliability is a form of consistency of judgment among various viewers (multiple observers) and is attained when the same themes are observed by different people who read the same information (Boyatzis 1998). This form of consistency has also been called Synchronic Reliability by Kirk and Miller (1986) (Kirk and Miller 1986) because it too outlines similarities between observers coding the same information at the same time. This form of reliability has also been endorsed by Holsti (1968) (Holsti 1968) and Smith (1992) (Smith 1992) which termed this category agreement with an expert which developed themes and codes within their research.

According to Boyatzis, having a standard protocol for conducting interviews can increase the consistency of judgment through providing a consistent setting for the capture and collection of qualitative data. Furthermore, the term "Confidence" is used to convey a sense of trust by the researcher(s) that in their research, they have captured the correct information and made sound judgments which could be considered a form of reliability (Boyatzis 1998). Similarly, other authors too have asserted that credibility is interchangeable with reliability in this case (Zyzanski, McWhinney et al. 1992).

Percentage agreement is an established procedure for measuring interrater reliability. There are variations in the way agreement scores are calculated, two of the most popular methods involve calculating the score of all judgments as a function of opportunity or calculating the percentage of agreement by the presence or absence of a coded theme. We have elected to follow the latter method for our study because they are known to be estimators of reliability and according to Boyatzis "Percentage agreement is the most typically cited measure of interrater or rater-expert reliability" (Boyatzis 1998). In our study coder A is the researcher and coder B is another researcher that is also conducting research on a different topic but using the same methods of analysis. Since the absence of a coded theme in our study does not mean the opposite of its presence, we have elected to use a popular variation on our calculation of percentage agreement which is to calculate the agreement scores based on presence only (Atkinson 1958; McClelland 1961; McClelland 1985; Smith 1992).

To calculate the agreement score, the "Percentage Agreement on Presence" equation by Boyatzis (1998), is used when there is not an equal likelihood of observing presence or absence:

$$\text{Percentage Agreement on Presence} = \frac{2\times(\#\ Both\ coders\ saw\ C\ preseent)}{\#\ First\ Coder\ saw\ C + \#\ Second\ Coder\ Saw\ C}$$

Note: C is the code being observed as present.

Table 4-21 shows the coder's various percentage agreement scores. Direct percentage agreement suggests that coders A and B saw a coded theme present.

Research Question	Theme	A	B	C	% agreement on presence
RQ1 – Combining Security and Agility	Dedicated Security Engineer	28	30	29	**94.9%**
	Experience of Developers	10	13	10	**86.9%**
	Informal Security Expert	14	14	15	**100%**
	Integration Risk	13	13	15	**100%**
	Security Control	18	18	18	**100%**
	Security in Planning	9	12	21	**54.5%**
	Software with Security in mind	16	23	19	**76.1%**
	Static Analysis Tools	7	10	8	**77.7%**
	Static Code Review	5	5	5	**100%**
RQ2- Change Agile Practices for Security	Agile Vs. Traditional Methods	23	27	31	**79.3%**
	Awareness of Security to Agile	16	26	24	**64.0%**
	Impact of Accelerated Schedules	16	21	16	**86.4%**
	No Change Necessary	7	7	7	**100%**
	Reduced Security for Internal Proj.	11	11	11	**100%**
RQ3 – Security Issues & Software Vulnerabilities	Impact of Experienced Developer	5	6	8	**71.4%**
	Relationship between Defects & Security	6	7	6	**92.3%**
	Security through Resolving Vulnerabilities	7	7	8	**93.3%**
	The Role of Security Training	5	5	5	**100%**
RQ4 – Customer's Control of Security	Customer must Dictate all Requirements	18	18	18	**100%**
	Customer needs help with security	6	6	6	**100%**
RQ6 – Knowledge Dissemination & Diffusion	Developers lack of Security Knowledge	12	13	13	**92.3%**
	Security through Training	5	6	5	**90.9%**
	Tacit Knowledge Dissemination & Sharing	6	6	6	**100%**

RQ7 – Dedicated Security iteration	More Public needs More Security	6	6	6	**100%**
	Periodic Security	14	15	14	**96.5%**
RQ7 – Testing & Verification for Security	Refactoring	21	21	21	**100%**
	Testing for Security	12	13	13	**92.3%**
	Tools Provide Added Security Assurance	5	6	8	**71.4%**
RQ8 – Documentation & Security	Abuse or Misuse Stories	12	12	12	**100%**
	Agile not Less Secure than Traditional Methods	11	12	11	**95.6%**
	Documenting the Agile Way	13	13	14	**96.2%**
	Reducing Documentation Risky in Agile	6	7	6	**92.3%**
	Simpler Code better than Documentation	3	3	3	**100%**
	Testing Over Documentation	7	7	7	**100%**
	Total:	373	419	419	**89.0%**

A: Number of Times Coder A and Coder B saw it present
B: Number of Times Coder A saw it present
C: Number of Times Coder B saw it present
% agreement on presence = 2*A/(B+C)

Table 4-22: Interrater Reliability Agreement Percentage table

According to Miles and Huberman, software can be used to estimate the consistency agreement level. They mentioned 80% or above agreement level is considered good qualitative reliability (Miles and Huberman 1994). Boyatzis (1998) also suggested that scores of 70% or better are necessary in this case which is in agreement with Miles and Huberman's estimate of good interrater reliability.

Variable	Coder A	Coder B
Mean	12.323	12.323
Standard Deviation	7.048	7.061
Kurtosis	0.130	0.516
Skew	0.945	0.998
Range	27.00	28.00

Table 4-23 : Descriptive Statistics about Each Coder

In order to determine the accuracy of interrater reliability estimates, three popular correlation coefficients are used to determine reliability (Li, Rosenthal et al. 1996). The Pearson product moment is the most used correlation coefficient which is used with a normally distributed interval data. Kendall's tau often referred to as a rank-order correlation "Based on counting the number of that pairs of things are in the same versus opposite order on both variables" (Cliff 1996). Spearman's rho computes a Pearson correlation between ranks established through converting scores into ranks.

Measure	**Coders A & B**
% Agreement Presence Only	89.0%
Pearson product-moment correlation	**.946****
Kendall's tau correlation	**.847****
Spearman Correlation	**.937****
NOTE: For correlations, N=34 ** Correlation is significant at the 0.01 level (2-tailed)	

Table 4-24: Correlation Calculation for Interrater Reliability

A look at table 4-22 reveals that our data is not normally distributed and therefore the Kendall's tau and Spearman correlation are more appropriate for interpretation in our case. Nevertheless regardless of whether or not the data is normally distributed or not, the correlation coefficients are all statistically significant which means there is a high degree of confidence in the reliability scores.

4.7.3 Qualitative Generalization

Qualitative studies are rarely if ever used to make generalizations, they are used to study a subject in depth and to develop themes and categorizations and therefore it is to a limited extent that the results of the qualitative work can be generalized. The aim of qualitative studies therefore is to achieve Particularity rather than Generalizability (Greene and Caracelli 1997).

4.8 Validating the findings

Validity means whether or not the findings really represent the results that we are present. According to Saunders, Lewis et al. (2007), "Concerned with whether the findings are really about what they appear to be about" (Saunders, Lewis et al. 2007).

4.8.1 History

For this threat to validity, the researchers must ensure that there have not been any recent events that might undermine people's opinions about the subject of your research. We have interviewed various stakeholders from different roles and background as well as different geographical locations in order to minimize any local, historical bias from entering into our results through artificial means. Additionally we are not aware of any incident where the responses of the participant(s) were not truthful or inaccurately represented their position within their respective organizations.

4.8.2 Testing

Testing says if the respondents feel their answers might in some way affect them after they have been interviewed this might affect the results. To guard against this threat, we chose to conduct our interviews in a completely anonymous fashion so the respondents may feel free from any pressure (job related or otherwise) to be able to answer our questions freely and honestly. Even with the assurance of anonymity we still encountered some reluctance by various people in sharing specific job related information. In some cases only after the recorder was turned off they started to open up about certain issues they were having that they were afraid to share formally.

4.8.3 Instrumentation

If the respondents' circumstances were such that it affected their results, we ensured that none of our questions were time and/or place dependant in order to minimize the chance of any unforeseen circumstances to affect the respondents' answers to our semi-structured interview questions. For example, people who opted for phone interviews tended to spend more time answering the questions perhaps because they were not under any time/scheduling pressure and were feeling more comfortable. On the other hand, meeting face-to-face allowed the interviewer to establish a greater

degree of trust and therefore the participants were more open to answer questions in detail.

4.8.4 Maturation

This threat is concerned with the question of: does any environmental and other events could possibly affect the participants' responses indirectly? In our research, unless something were to happen during the interview itself that might prompt a participant to change their answers, for example their boss walking into the interview meeting, there is no conceivable way that we could think of that could affect our results. On the case of the boss being around, we ensured this would not happen by scheduling the participant at a neutral date and time that is most suitable for them either on the phone or face-to-face in a public setting. We did encounter one case where the potential participant and their boss where in the same area and that made the potential participant hesitant and reluctant to answer questions freely.

4.8.5 External Validity

External validity is concerned with whether or not the results apply to settings other than those that was conducted by the researcher, such as other organizations, etc. Since we have chosen our participants from various roles and diverse geographical locations, we have tried to ensure that our results are applicable to as many software development firms as possible. These results would ideally be applicable to any team within any size organization that is of small to medium size which is following one or more Agile principles and/or methodologies. Furthermore the more experienced the team member the better insight they would have into the issues and discussions involved so the relative accuracy of each interview cannot be compared but the collective responses given by all participants should be consistent with a similarly chosen sample from the same diverse population of practitioners.

4.8.6 Logic leaps and false assumptions

These set of threats discuss Data Collection issues: "Is it logical to assume the way you are collecting your data is going to yield valid data?" (Saunders, Lewis et al. 2007). Since we are collecting our data in an entirely anonymous way, we have allowed for the highest degree of transparency from the respondents without the

possibility of any backlash or retribution therefore we feel our data collection is as valid as one could reasonably expect and the number of interviews and consensus that we have reached further suggests that we have indeed been getting consistent and valid results. To ensure that we progressed from a mountain of data to our conclusions in a coherent and structured (using a theoretical framework) fashion, we followed a hybrid approach of inductive and prior research driven framework. The first part of the approach (which is the qualitative aspect) gave rise to the hypothesis that will be tested throughout the later stages of the research (quantitative aspects).

4.8.7 Negative and Contrary Responses

Even though the process of semi-structured interviews was completed successfully and we gathered the required amount of information for each interview, there were instances where we were faced with inconsistencies and additional concerns despite our assurances of anonymity and confidentiality. For example, when we asked a participant, What are your project or code level security practices that are in use today in your organization? We got the answer of I have to tell you I cannot answer some specific questions for the company I work for. This is because that is about security. Only after we asked the same question from different perspectives but more subtly was when we got relative answers but in general some people exhibited an automatic negative response to sharing any kind of security related information anonymity or not which underscores the sensitive nature of the topic under investigation.

4.9 Conclusion and Further Research

This chapter concludes the qualitative aspect of the investigation into security issues in Agile which allowed us an in-depth look into the various details from multiple perspectives and points of view. From the analysis and interpretation of the results we formed a number of hypotheses that are related to the top 4 research questions in terms of the amount of occurrence and consensus that practitioners reached on the various issues related to these questions. From this point on we will shift focus to quantitative aspects where we will continue the investigation by testing the hypotheses with questionnaires on various issues and topics raised in the

qualitative study combined with the prior research issues that were similar to the one we discovered empirically.

Chapter 5 The Security Adoption Questionnaire

5.1 Introduction

The questions for the survey will be derived from the results of the interview process which should bring a more specific set of questions that are based on the responses given by practitioners and experts in the field. This questionnaire attempts to capture valuable information and metrics from the team members which provides concrete evidence for further analysis, review and decision making as part of assessing the suitability of inclusion of security specific practices into Agile. Furthermore, it will serve as a test for the hypotheses that were generated from the results of the qualitative phase of the research. Based on the overall occurrence of research questions table, it is clear that the top 2 research questions are the most practical in terms of practitioner focus in the interviews which combined cover almost 55% of all occurrences within the responses given to our interview questions. We chose the 8 hypotheses related to these two research questions in order to cover the majority of the responses for the second phase of the investigation. The aim is to further test the top two research questions as were found through our frequency analysis which underscored the most important and predominant security issues in Agile today.

5.1.1 Purpose of the survey

The purpose of this questionnaire is to build on the results of the qualitative method of in-depth semi-structured interviews in order to gain a broader consensus on the practitioners' attitudes towards major issues that resulted from the thematic analysis of the interview data conducted earlier. According to Cresswell, the general purpose of a questionnaire is to "generalize from a sample to a population so that inferences can be made about some characteristic, attitude, [or] behaviour of this population" (Babbie 1990; Creswell 2009). This survey will primarily test for the opinions of the Agile practitioners on the various issues found in the literature to be of significant importance as it relates to security which is the focus of this research to investigate further.

5.2 The Design

The target sample would be selected from various Agile practitioner groups through attending professional events and using mailing lists such as Agile developers group, the security analysis and review board members, the QA and testing group, the management and decision making group, and last but not least, the customers and major stakeholders in the project. Participants were contacted through emails, online postings, or in-person and asked to participate in our survey. Similar to the semi-structured interviews, the geographical distribution of the participants was taken into consideration in order to have a representative sample from around the world.

5.2.1 The choice of questionnaire

We chose the questionnaire method as a practical means to gage more widespread opinions than we found during the in-depth interviews using a larger sample. This allows us to make a better case for why these issues are important to larger audiences such as developers, managers, and other Agile stakeholders who wish to know and understand better how to implement security mechanisms into their software development lifecycles. This is beneficial to the overall research because it allows us to gain a wider perspective on these issues in order to be able to further substantiate the results and be able to corroborate the wider perspective with the more in-depth but smaller sample's results obtained from the in-depth semi-structured interviews. Also of benefit is the economy of the design as well as the rapid turn-around in data collection that a survey is known for.

5.2.2 The nature of the survey

The survey will be conducted as a cross-sectional questionnaire. This is because we are conducting the survey at a defined time based upon specific results found through a hybrid method of data driven plus prior-research driven approach.

5.2.3 Population and sample size

The population targeted for the survey is chosen based upon the level of experience and domain knowledge of the practitioners on the issues related to Agile software development and secure software development. The subjects are chosen from a pool of practitioners, managers, and stakeholders that have related their experience to

these topics and are interested to participate in the survey. The selection criteria would include being from a background of software engineering and development and related fields that impact the agility and security of the final product.

The Multi-Stage sampling design method would be ideal when it is impractical to compile a unified list of all the population elements (Babbie 2007). Since there is no reliable evidence for the total number of Agile practitioners, coupled with the fact that there is no way to be able to sample each sub-group independently before conducting the survey (as is the case with random, stratified sampling), we need to conduct a judgmental non-probabilistic sample through the use of online survey instruments. Although this convenience sample will not be as desirable as a random sample, it will give us enough information to be able to conduct this study keeping in mind the economies of scale and other benefits described earlier for online surveys. We gained access to our target population through joining specific events, mailing lists, discussion groups, specialized forums, and professional online portals which had members that were practitioners in Agile or a number of its related practices (such as XP and Scrum). From there we post invitations for all members to participate in our survey and add the number of members for each group to our running total in order to arrive at our target population number which turned out to be just over 100,000 practitioners. The number of completed surveys is expected to be 100 or more participants which vary depending on the analysis method we choose to employ on their response data.

5.2.4 The Survey Instrument

The SurveyMonkey tool (available from www.SurveyMoneky.com) was used to administer the survey and to collect information from practitioners anonymously. This instrument has been used to conduct multiple similar academic and empirical studies in the past. It is designed for any researcher who would need a questionnaire tool to be readily available and easy to administer. According to Creswell (2009), “instruments are being increasingly designed for online surveys (Sue and Ritter 2007)” (Creswell 2009). We left this tool intact since there were no additional modifications needed to be made for this study so it is used as-is.

To be able to carry out our questionnaire plan we requested approval from the Electronic and Computer Science School Ethics Committee at the University of Southampton (reference E/11/05/006). Appendix D contains the questionnaire given to the practitioners.

5.2.5 Data Collection

The data will be collected in the form of a self administered questionnaire. Self administered questionnaires are a popular form of data collection for many types of research (Nesbary 2000; Sue and Ritter 2007).

5.2.5.1 Scales and content Areas

A number of specific questions are asked and the opportunity for specific agreement response is given using a 7 point Likert scale (Likert 1932). The data from the survey will be collected and will be analyzed quantitatively using appropriate statistical methods and techniques to be discussed later.

5.2.5.2 Variables

Type	Name	Questionnaire Item	Hypothesis	Research Question
IV	Addition of Security Engineer	1-16	H1a	RQ1
DV	Suitability for inclusion in Agile	1-16	H1a	RQ1
IV	Security in mind	17-24	H1b	RQ1
DV	Suitability for inclusion in Agile	17-24	H1b	RQ1
IV	Security Controls	25-32	H1c	RQ1
DV	Suitability for inclusion in Agile	25-32	H1c	RQ1
IV	Experience of Developers	33-42	H1d	RQ1
DV	Suitability for inclusion in Agile	33-42	H1d	RQ1
IV	Agile Methods	43-51	H2a	RQ2
DV	Necessity of security integration w/Agile	43-51	H2a	RQ2
IV	Security Awareness	52-65	H2b	RQ2
DV	Necessity of security integration w/Agile	52-65	H2b	RQ2

IV	Accelerated Schedules of Agile	66-75	H2c	RQ2
DV	Necessity of security integration w/Agile	66-75	H2c	RQ2
IV	Internal Agile Projects	76-80	H2d	RQ2
DV	Necessity of security integration w/Agile	76-80	H2d	RQ2

Table 5-1 : Variables and their Related Research Questions and Hypotheses
(IV: Independent Variable, DV: Dependent Variable)

5.2.6 Field testing procedure (by a panel of experts)

Security related issues are extremely sensitive topics to discuss and research about (Straub and Welke 1998). Because of the perceived sensitivity towards security-related questions, which is partly due to the intrusive nature of the subject, many researchers have experienced inadequate response from practitioners which is why they urge caution when conducting surveys because a general fear by practitioners exists to any attempt of data gathering about the practices and behaviors of security professionals (Kotulic and Clark 2004).

These types of questions are a potential source of undesirable method bias. Therefore, a non-intrusive instrument reduces the method variance associated with the survey by urging participants to answer questions honestly and makes them feel more at ease by telling them that their identities will be kept completely anonymous. As part of the reliability measures taken for this survey, and since the nature of this research is more sensitive than others (security issues), we elected to first field test the survey and administer it to a panel of experts in order to assess whether or not the survey questions would be considered intrusive.

The researcher chose five experts from the pool of people that were previously considered as experts on software process and security which were to be interviewed but somehow could not find the time to do so.

The panel of experts evaluated each candidate item for intrusiveness. They were given the questionnaire items for the evaluation and were expected to rate them according to a five point Likert scale measuring the degree of comfortability in

answering the question. They were also encouraged to comment and make suggestions for improvement.

Weight	Scale
1	Uncomfortable
2	Somewhat Uncomfortable
3	Neutral
4	Somewhat Comfortable
5	Comfortable

Table 5-2: Comfortability Scale used in Field Testing

In total, panelists provided about 30 comments on the items in addition to the ratings for each question. The following criteria were used in evaluating each item. An acceptable item must:

1) be rated as either Neutral (3) or Somewhat Comfortable (4) or Comfortable (5) by at least 60% of the experts
2) have a mean score from all the panelists of at least a 3.0 on a 5.0 scale.

If one of the experts read an item that they thought especially problematic, they provided feedback on why the item was not acceptable. For example, one expert claimed that asking "Having a security engineer within the Agile team can Be better for less mature organizations' security needs" would make the respondent think their organization is not that mature if they feel the need for a security engineer and they might not want to acknowledge that, therefore he rated the item as Somewhat Uncomfortable (2) and the question was ultimately dropped because the mean of the rating fell below the required 3.0 threshold.

Combining the comfortability scores with the feedback from the experts resulted in the removal of 3 items (out of 83) and refined the instrument overall. As is evident from the questions themselves, the panel of experts felt these questions were not significant enough in terms of their technical substance and were more meant to elicit people's general opinions about the issues which was more appropriately addressed by other more specific questions.

The idea is to try and minimize the risk of willingness to answer threat that many surveys suffer from if they don't adjust their questions for this specific issue. This will help us to improve questions, format, and scales and also helps to minimize threats to validity of the instrument (refer to appendix G for details).

5.3 Data Preparation

One of the initial steps in quantitative analysis is data preparation. The aim is to prepare the survey data before conducting any statistical analysis technique(s). This is done by determining which questionnaire items are qualified by field testing them and by looking at the response to each questionnaire section to determine the respondent's eligibility to be included in the final results. For example, they may have only answered background questions and subsequently quit the survey which means that the participant is disqualified from the analysis since they obviously did not answer any specific question related to the survey.

We have codified the symbolic responses such as "Strongly Agree" into numbered codes such as 7 to make questionnaire item results suitable for analysis in SPSS. Each survey contains multiple sections that will be analyzed independently or in groups. We will exclude any respondent that left some answers in a given section of the survey unanswered.

Finally, we will end the data preparation through identification and possible exclusion of any response that is very different from the remaining scores (outliers) of a given questionnaire item in the survey by checking to see if the existing outliers are impacting the data substantially or not. This will help in reducing bias which could affect the result and subsequently weaken the conclusion. We will later assess if our targeted statistical techniques are suitable for our gathered data and present the evidence in support of that.

5.3.1 Data Screening

The first step in data preparation is data screening where we go through the survey data looking for issues and/or errors that need to be corrected. This screening usually

occurs either through manual inspection of the survey data or the use of specific techniques to identify issues such as the use of z-scores to help in identifying outliers in given questionnaire item. We now present two activities which are part of our data preparation: missing data and influential outliers.

5.3.1.1 Missing Data

Finding suitable practitioners for our highly sensitive and detailed survey was challenging. However, we were able to find enough participants to give us a meaningful results.

It is possible for a survey participant to quit before answering all the required questions in the survey. The following graph depicts what happened to the number of respondents as the number of sections that each participant was presented with increased. Since the practitioners did not know exactly how long each section was going to take, they chose not to continue the survey and prematurely quit participating in the survey. In total, out of 184 people who chose to take part in the survey, only 120 ended up finishing the survey completely which suggests an estimated 35% of the participants did not complete the survey.

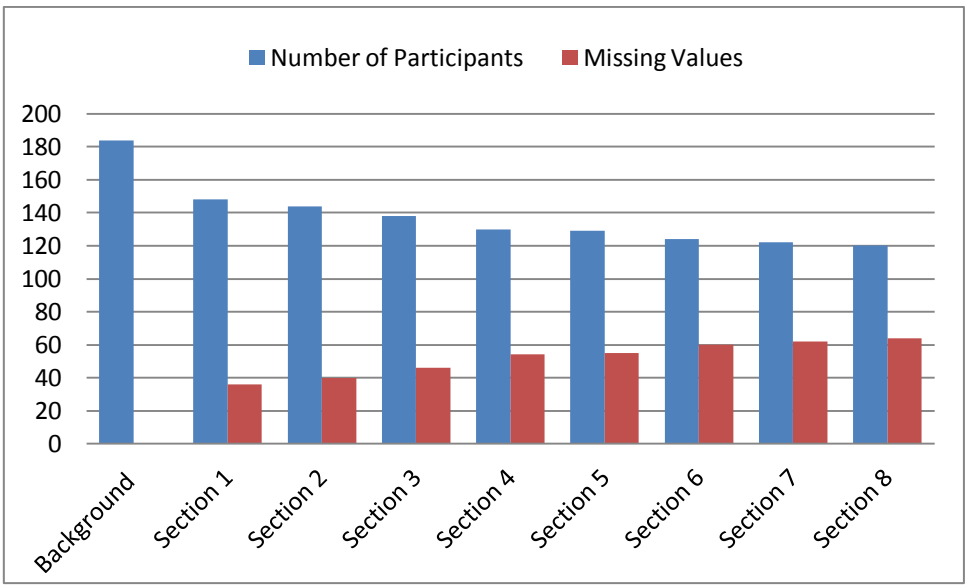


Figure 5-1: Number of Participants for each Section of the Survey

Since the survey was divided into discrete sections, each section could be analyzed independently. Therefore, the survey would be considered complete if one or more sections have been answered completely even though not all sections have been completed. To handle missing data, one method is to exclude any respondent that did not complete that targeted section for the analysis. This will help us to make conclusions based on their responses up to and including the last section that was completed. Although this resulted in reduced sample size for our analysis, it is considered the safest way to conduct the analysis (Tabachnick and Fidell 2007; Field 2009).

We could have alternatively handled the missing data through replacing the additional unanswered questions with the mean of other respondent's responses to the same question. However, this would have inflated the mean causing possibly significant changes to result that is not as accurate as possible (Tabachnick and Fidell 2007; Field 2009). Based on the merit of the former method, we chose to exclude any respondent that did not finish that particular section of the questionnaire for the analysis.

5.3.1.2 Outliers Screening

Another step in preparation of data for analysis consists of screening the collected data for outliers. An outlier is a score (a response from the respondent to the questionnaire item that we have codified) which means the chosen answer was very different than how the majority of the respondents answered the same questionnaire item (Field, 2009).

The existence of outliers in the collected data may lead to increase the standard deviation of data. At the same time, it will influence the mean of the scores causing a bias to be introduced into the result (Tabachnick and Fidell 2007). Outliers can be found through using z-scores (standardized scores). Tables 5-2 and 5-3 contain calculated z-scores for each questionnaire item.

Any questionnaire item that has an absolute value of z-scores (|z|) bigger than 3.29 at $p < 0.01$ is considered a questionnaire item with potential outlier(s) (Tabachnick and Fidell 2007). Tabachnick & Fidell (2007) also suggested that, the percentage of such outliers should not exceed one percent of the total score for any questionnaire item. Looking at the z-score in tables E-1 and E-2 in Appendix E shows that a number of questionnaire items with potential outliers exist and they are within the acceptable level of one percent.

To make sure that the existing outliers did not impact the data substantially, we decided to calculate the difference (ΔMean) between the mean and the 5% trimmed mean for questionnaire items with outliers. 5% trimmed mean is mean calculated of scores excluding the top and bottom 5% of the results (Pallant 2007). Pallant (2007) suggested that the existence of a large ΔMean in any given questionnaire item with potential outlier(s) indicate that the specific outlier(s) may negatively influence the result (Pallant 2007). The ΔMean of each item shows that all questionnaire items with potential outliers have a ΔMean that is relatively small ranging from 0.01 to 0.14. This means that the detected outliers are not going to negatively influence the analysis. Therefore, we decided to include all the cases for further analysis.

5.3.2 Assessing Suitability of Data for Analysis

5.3.2.1 Scale Types

Data is usually presented using various scale types because the captured data may not be in numeric form. In our case, the captured data is in symbolic form (ex: “strongly agree”). The Likert scale defines a mapping between our symbolic representations and a numeric rating which implies the strength of the responses in terms of the level of agreement of each response to the posed question (questionnaire item in the survey). The more appropriate the choice of scale type is, our subsequent analysis and results will be much stronger as a result. These types differ based on the nature of data that is to be measured. Some of the widely used scale types are: nominal, ordinal, interval, and ratio.

Nominal scale type is for data that can be classified into categories or classes (Fenton and Pfleeger 1997). In such classification there is no ordering between categories or classes. An example of such a scale type is the classification of participants into male or female. For our data, since we have specific relationship between categories (levels of agreement), this scale type is not appropriate.

Similar to nominal, ordinal scale type classifies the data into different categories. However, this scale type preserves the ordering of these categories which is more appropriate for our data. Each of these categories can be represented by a number but it does not make sense to perform mathematical operations on these numbers. We can use this scale type for our analysis, but there is a limitation with the manner in which the addition or subtraction of the categories can be achieved.

Similar to ordinal, interval scale type classifies the data into different categories. It also preserves the ordering and differences between these categories. Through the use of this scale type, we can perform mathematical operations such as addition and subtraction on our numeric categories but not multiplication or division. Since we have chosen the 7 point Likert scale for our questionnaire, it allows us to treat our data using an interval scale type (Oppenheim 2001). Another useful side-effect of choosing a 7-point Likert scale over the more widely used 5-point Likert scale is that we can treat the responses as more continuous than ordinal since we are now making relative differences between responses more clear to a point where it feels much more plausible that the distance between the scales is equal which is a requirement of continuous (interval) scales. According to Byrne, "the literature to date would appear to support the notion that when the number of categories is large [5 or more] and the data approximate a normal distribution, failure to address the ordinality of the data is likely negligible" (Muthén and Kaplan 1985; Babakus, Ferguson et al. 1987; Atkinson 1988; Byrne 2010). It is also notable to mention that a Likert scale result is similar to a normal distribution (Likert 1932). One limitation of this scale type remains which is that we cannot take a ratio of two classes. For our purposes however, this is not required by our targeted analyses.

Similar to interval, ratio scale type classifies the data into different categories. It also preserves the ordering and differences between these categories and ratio make sense between categories. For these reasons, most of mathematical operations such as addition, subtraction, multiplication, and/or division are applicable but in our case it does not make sense to use multiplication or division on our mapped categories (to numbers).

For our survey data, we can safely assume that the ordinal and interval scale types are both applicable as scale types for analysis but the interval scale type is the most appropriate since we have used a large number of categories for responses to our questionnaire items as evidenced by the choice of the 7-point Likert scale.

5.3.2.2 Assumption of Ordinality

As a condition prior to use parametric test such as ANOVA on any data, the data needs to meet a number of assumptions and/or concerns. With 7-piont Likert Scale as the classification of our data, a number of concerns need to be addressed before the use of the parametric test. The first concern that we have already addressed is the issue of Ordinality. The other is the assumption of normality which needs to be satisfied before applying the parametric test.

5.3.2.3 Assumption of Normality

The assumption of normality is the most essential assumption for many statistical techniques such as parametric tests and factor analysis (Field 2009). Normality of a variable (a questionnaire item in our case) is an estimation of how similar the shape of its data distribution is to a normal distribution (which resembles a bell-shaped curve) which is typically derived from certain other characteristics of the distribution such as skewness and kurtosis.

The normality of any variable can be evaluated using statistical or visual methods. For statistical methods, researchers generally use kurtosis and skewness (Tabachnick and

Fidell 2007). Kurtosis measures the flatness of the distribution compared to a normal distribution whereas skewness measures the distribution's symmetry (Hair, Black et al. 2006). A positive kurtosis indicates a peaked distribution, while negative kurtosis shows the distribution is relatively flat. On the same note, positive skewness means the right end of the distribution is longer, while negative values indicate the left end is longer. In theory, a perfectly normal distribution would have a zero value for both kurtosis and skewness which is unusual in the social sciences research. Garson (2012) argued that for a distribution to be assumed normal, kurtosis and skewness values have to be in the range of ± 2.0 (Garson 2012).

In tables 5-2 and 5-3, all values of skewness are between -1.21 and 0.78 and kurtosis are between -1.18 and 2.0 which are within the recommended ranges. In addition to assessing normality using kurtosis and skewness, Field (2009) recommended the visual inspection of the histogram of the data distribution for all variables, which showed that our data distribution was normal. Therefore, we can conclude the normality of our data set when it comes to both statistical and visual methods.

According to Norman (2010), "Parametric statistics can be used with Likert data, with small sample sizes, with unequal variances, and with non-normal distributions, with no fear of "coming to the wrong conclusion"". For parametric tests such as ANOVA, he argued that researchers often misunderstand the assumption of normality by assuming that the data must be normally distributed where it is important that the means are normally distributed. Based on the Central Limit Theorem, any data with sample sizes bigger than 5 per group, the means are approximately normally distributed no matter what the original distribution of the data was (Norman 2010).

When it comes to the use of other methods such as factor analysis, Tabachnick & Fidell (2007) stated that "as long as PCA [Principal Component Analysis] and FA [Factor Analysis] are used descriptively as convenient ways to summarize the relationships in large set of observed variables, assumptions regarding the distribution of variables are not in force. If variables are normally distributed, the solution is enhanced. To the extent that normality fails, the solution is degraded but may still be worthwhile" (Tabachnick and Fidell 2007).

The previous quotes from Norman (2010) and Tabachnick & Fidell (2007), in addition to the support provided by statistical and visual methods clears any issues or concerns regarding the use of parametric tests or factor analysis on our 7-point Likert scale data.

5.4 Analysis & Interpretation

The results of the questionnaire will be analyzed descriptively through a number of statistical procedures including means, standard deviations, and range of scores for the responses to questionnaire items. We will follow this with a codification scheme that allows us to consider all positive and negative responses as one relatively distinguished scale for analysis. SPSS will be used to conduct an Exploratory Factor Analysis (EFA) on the data. We will follow through with ANOVA and other more appropriate methods. For reliability, we plan to use Cronbach's Alpha statistic to establish internal consistency of scales for our data and determine whether there was consistency in our testing procedure and scoring (Borg, Gall et al. 1993).

5.4.1 Background Information of Respondents

The sampling for the questionnaire targeted a broad geographical spectrum from all of the continents and of course since the study was being done from Europe, there are more responses from this region because the researcher could more readily reach professionals from the UK and the surrounding countries. This shows a diverse sampling which was the aim of the questionnaire.

Where are you located?		
Answer Options	**Percent**	**Count**
Europe	**71%**	**105**
North America	18 %	27
Asia	8%	12
South America	1%	2
Africa	1%	1
Australia	1%	1
answered question		**148**

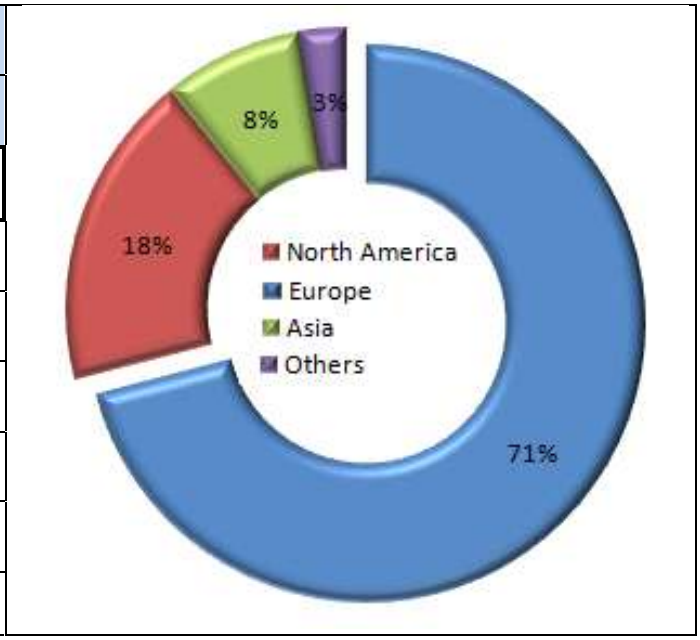


Figure 5-2 : Respondents' Location

From amongst the organizations that were surveyed, 47% of the respondents were professionals from IT, Security and Telecom industries that took the time to respond to our questionnaire. The next major group belonged to Consulting and Professional Services firms that captured 21% of the results which underscores the importance of security especially for these two major areas. The third most influential industry for the study was from the Finance, Banking, and Insurance industries which provided 11% of the responses. Surprisingly, the banking and finance industries came behind Telecom, IT, and consulting firms in terms of the interest its professionals showed in responding to our questionnaire. The remaining 21% of respondents came from a wide variety of industries which may not necessarily place an important emphasis on security.

Type of Organization		
Answer Options	**Percent**	**Count**
Information Technology, Security, Telecom	**47%**	**69**
Consulting/ Services	21%	31
Finance, Banking, Insurance	11%	17
Manufacturing/ Retail	5%	8
Education, Training	4%	6
Government	3%	4
Other	9%	13
answered question		**148**

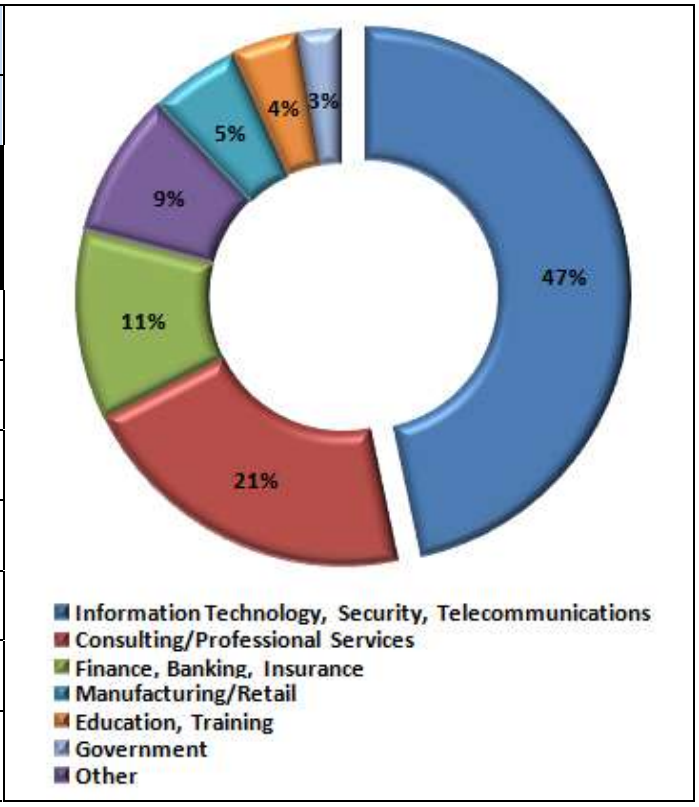


Figure 5-3 : Respondents' Organization Type

The following shows the distribution of the estimated number of people in the organization in which each respondent came from. The distribution is a good mix between smaller organizations with less than 300 employees which had 49% along with medium sized organizations which had between 300-2000 employees (24%) and larger enterprise class of organization with more than 2000 employees (27%). This distribution underscores that fact that if we get a good consensus amongst these professionals which are representative of all major organizational sizes then we will have a representative result.

What is the total number of employees in your organization?		
Answer Options	**Percent**	**Count**
Less than 300	**49%**	**73**
301 -2,000	24%	35
2,001 – 5,000	4%	6
5,001 – 20,000	6%	9
More than 20,001	17%	25
answered question		**148**

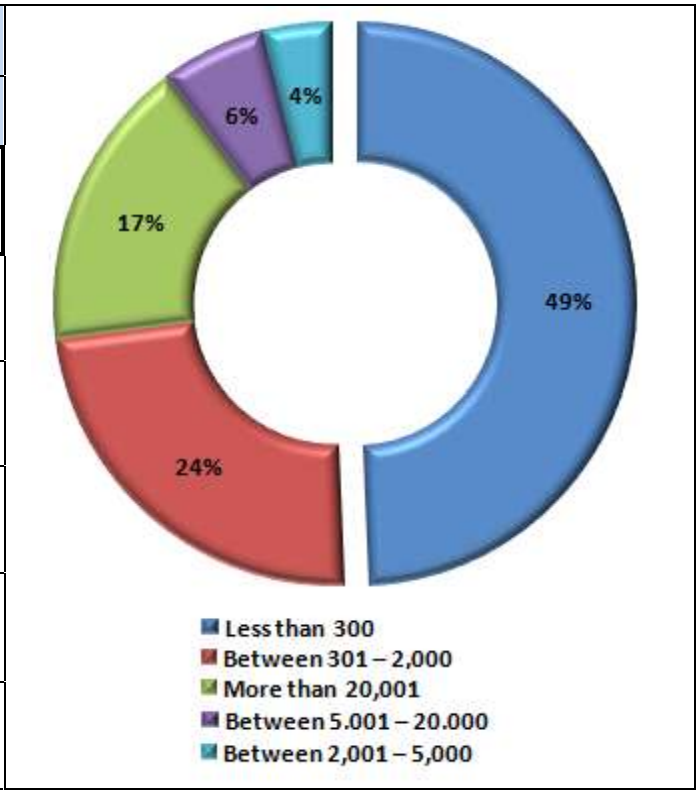


Figure 5-4 : Respondents' Organizational Size

The level of education of respondents shows that the majority 94% of the respondents are highly educated professionals which shows the responses are of high quality and based on experience.

What is your level of education?		
Answer Options	**Percent**	**Count**
Less than Bachelor's Degree	6%	9
Bachelor's Degree	**48%**	**71**
Master's Degree	43%	63
Ph.D	3 %	5
answered question		**148**

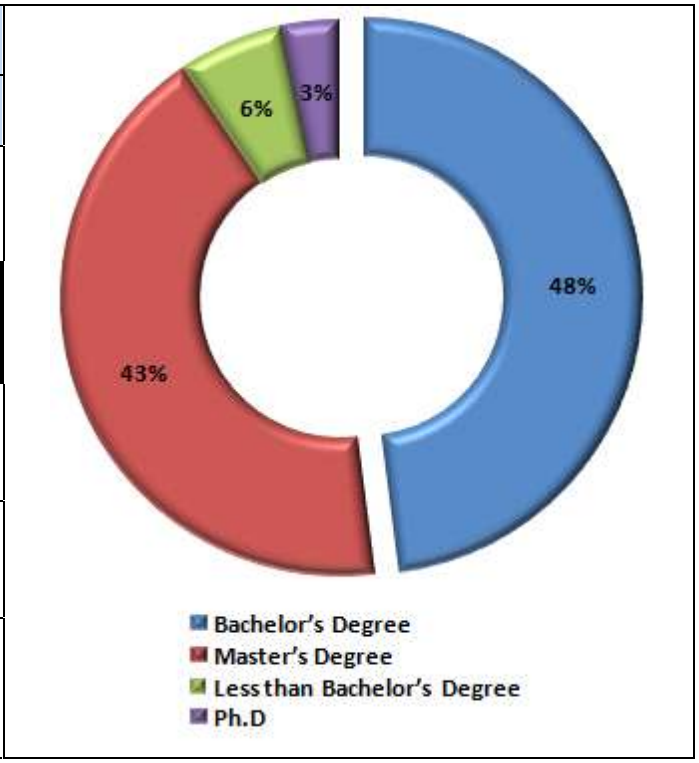


Figure 5-5 : Respondents' Education Level

Since the focus of the research was on security issues in Agile projects and project management in general, the role(s) that were most important to our research were that of Software Engineer and Architects and Project Managers which make up 58% of the responses. Of course, the Programmers, Consultants, QA and Systems engineers' opinions are very relevant and valuable as well which account for 30% of the respondents. From the rest of the 12% we see a mix of various professionals whose interest in the subject makes their opinions valuable for the research.

Where is your role within the organization?		
Answer Options	**Percent**	**Count**
Software Engineer /Architect	**38%**	**56**
Senior Manager /Project Manager	20%	29
Programmer	12%	17
Consultant	8%	12
Tester/ Quality Assurance	7%	11
Systems Engineer /Analyst	3%	5
Security Engineer/ Security Auditor	3%	5
Customer/ Stakeholder/ Owner/Partner	3%	4
Other	6%	9
answered question		**148**

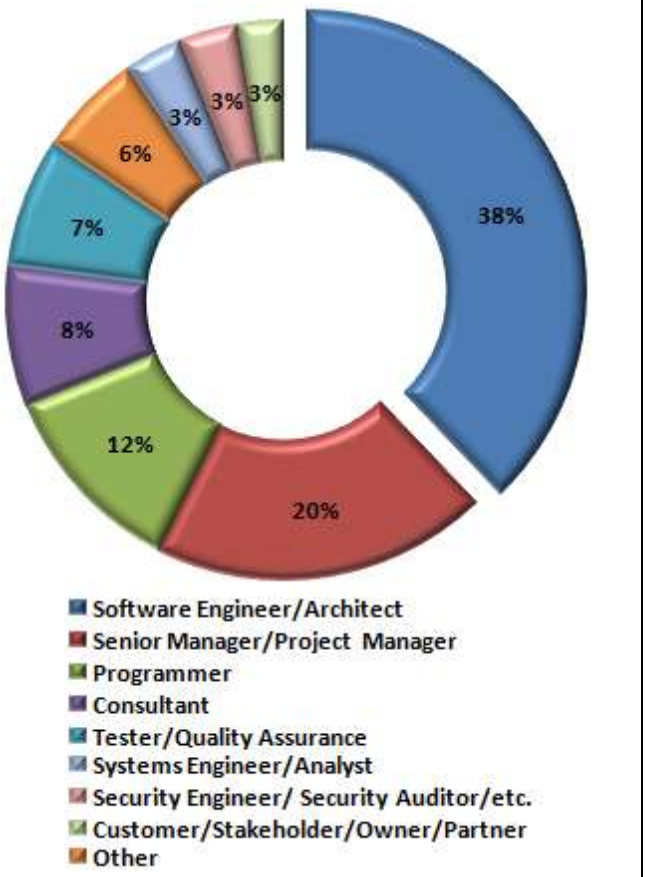


Figure 5-6 : Respondents' Role within Organization

The experience level of 95% of the respondents is above 4 years of professional work experience in various sized teams and organizations which shows the majority of the respondents have the adequate experience and knowledge to draw from in answering our questions.

How many years of overall experience do you have?		
Answer Options	**Percent**	**Count**
Less than 1 Year	2%	3
2-3 Years	4%	6
4-6 Years	15%	22
6-10 Years	20%	30
10+ Years	**59%**	**87**
answered question		**148**

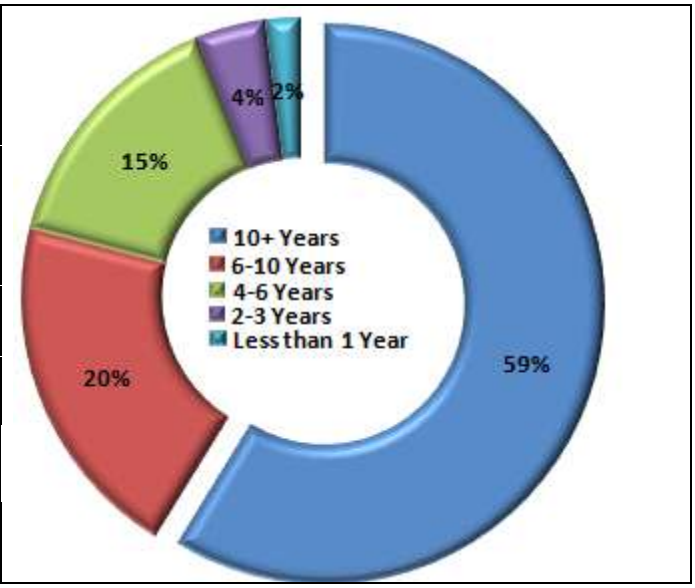


Figure 5-7 : Respondents' Overall Experience

The following question specifies above and beyond the overall experience level of the respondents how many years they have been working in their current position within their respective organization which clearly shows a very good overall distribution of experiences within each person's respective area of expertise.

How many years of overall experience do you have in your current role?		
Answer Options	**Percent**	**Count**
Less than 1 Year	18%	26
2-3 Years	**24%**	**35**
4-6 Years	22%	33
6-10 Years	16%	24
10+ Years	**20%**	**30**
answered question		**148**

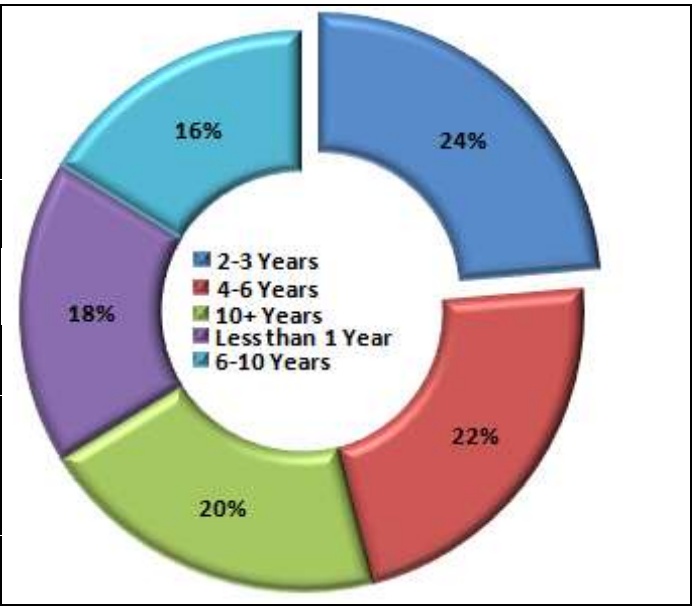


Figure 5-8 : Respondents' Experience in their Current Role

The most prevalent methodology being used amongst professionals was Scrum followed by XP and TDD which collectively represents 78% of the software development methodologies that the respondents had adopted in their teams. Since the "Other" category received many inputs, we looked into the self entered methodologies and they range from ad-hoc to a mix of multiple Agile flavors used at once.

Which Agile methodology was most recently used in your organization?		
Answer Options	**Percent**	**Count**
Scrum	**51%**	**75**
eXtreme Programming (XP)	16%	24
Test Driven Development (TDD)	11%	16
Feature Driven Development (FDD)	4%	6
Other	18%	27
answered question		**148**

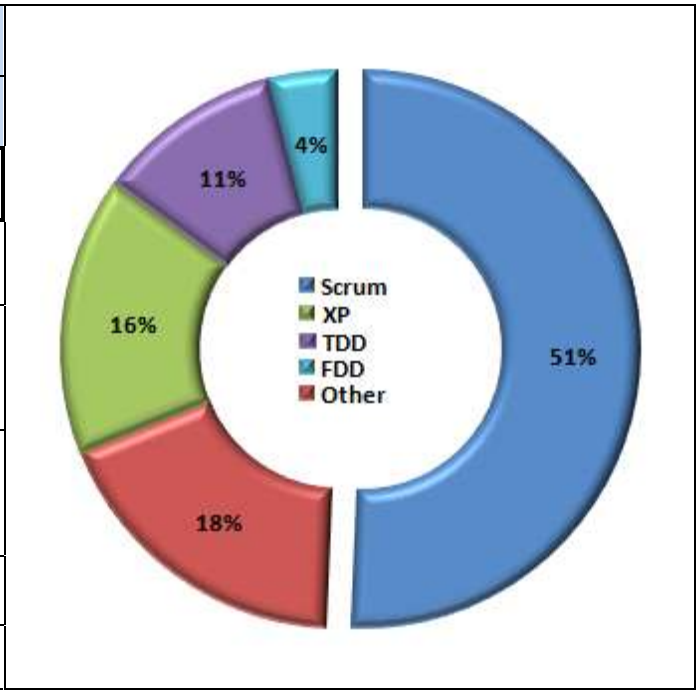


Figure 5-9 : Respondents' Recently used Methodology

The respondents in answering the question on the team size provided the expected average number which is in the majority of the cases less than 10 team members per team (74%) and the rest that are beyond 10 may have sub-groups within the team or project that they are a part of.

Where was the team size?		
Answer Options	**Percent**	**Count**
1-5	**33%**	**49**
6-10	**41%**	**60**
11-20	11%	17
21-50	9%	13
51-100	3%	5
101+ people	3%	4
answered question		**148**

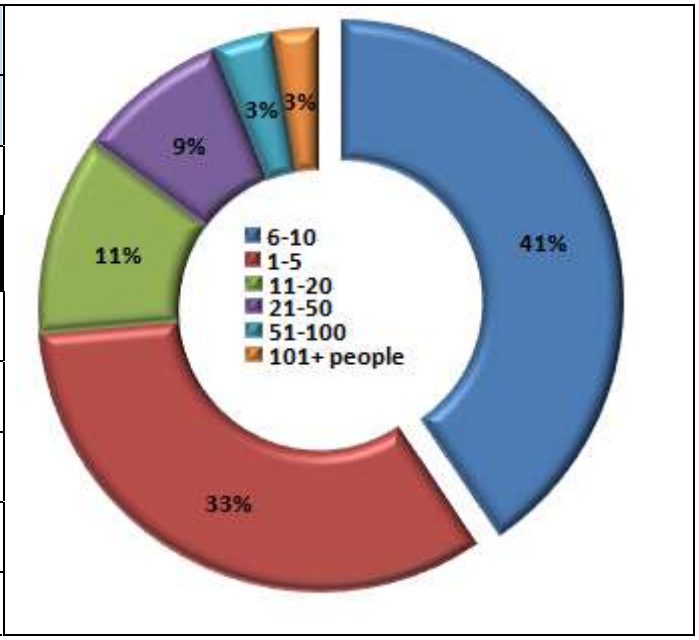


Figure 5-10 : Respondents' Team Size

Since Agile methods are relatively new to the industry then it makes sense for the majority of the respondents to have less than 6 years of experience with the new Agile methods (79%) and the rest either follow no specific methods or have a traditional waterfall methodology that they follow (21%).

How many years of experience do you have working with the current methodology?		
Answer Options	**Percent**	**Count**
Less than 1 Year	21%	31
2-3 Years	**34%**	**50**
4-6 Years	24%	36
6-10 Years	17%	25
10+ Years	4%	6
answered question		**148**

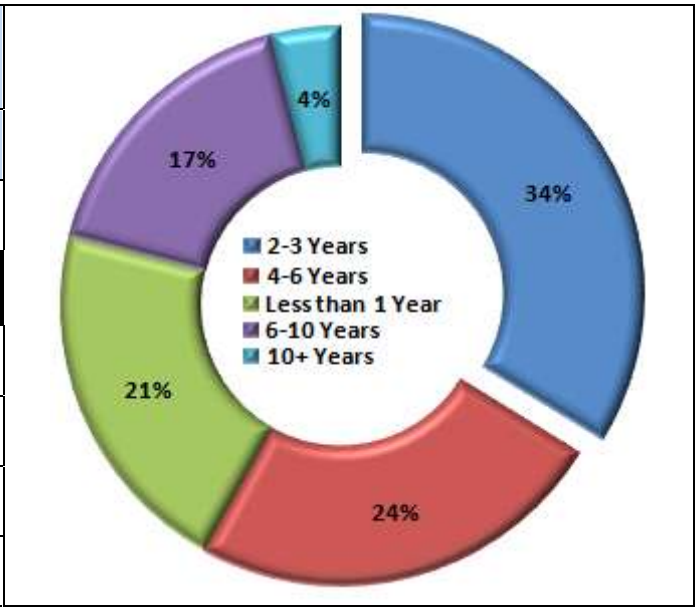


Figure 5-11 : Respondents' Experience in Current Methodology

The following question interestingly shows that even though the practitioners might be following an Agile method, they are still using the iterations in a traditional sense of the word which means a number of iterations per release (88%) and only 11% are following a strict iterative process which means 1 release per iteration.

If you used an Iterative methodology, how many iterations did it take to produce a major release?		
Answer Options	**Percent**	**Count**
1 Iteration	11%	16
2-3 Iterations	19%	28
4-5 Iterations	20%	30
6-10 Iterations	**29%**	**42**
11+ Iterations	20%	30
answered question		**146**

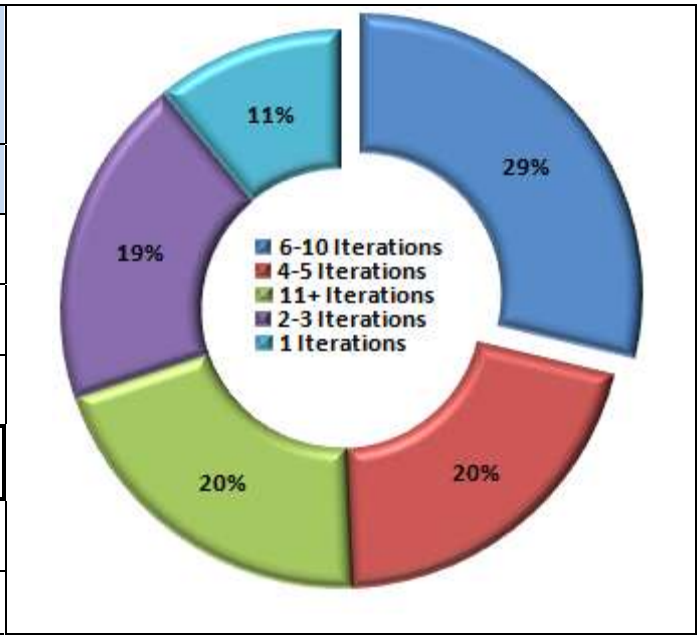


Figure 5-12 : Number of Iteration(s) to Produce a Major Release

The following question shows that the majority of professionals are following an Agile time-frame for the length of an iteration (82%) and the remaining respondents that chose 4+ weeks for an iteration may be using a combination of Agile and Traditional process for their software development needs.

What is the length of one iteration?		
Answer Options	**Percent**	**Count**
1 Week or less	20%	29
2 Weeks	**53%**	**78**
3 Weeks	9%	13
4-6 Weeks	13%	20
7+ Weeks	5%	8
answered question		**148**

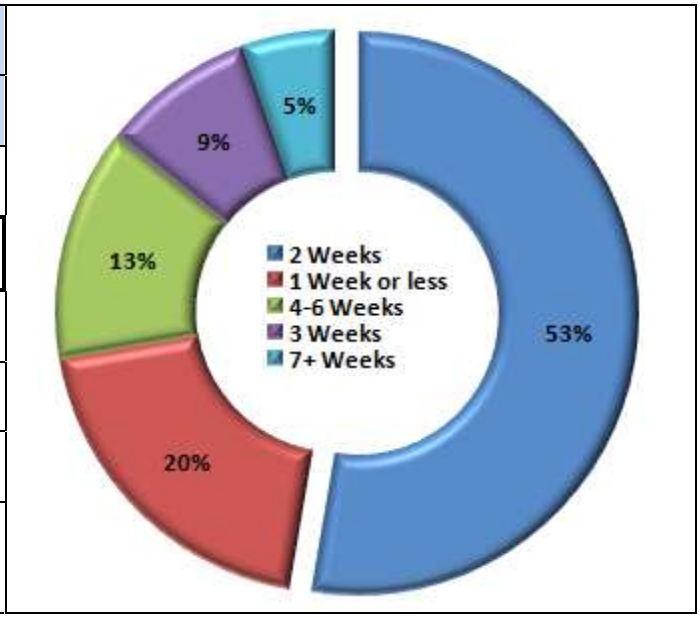


Figure 5-13 : Iteration Length

5.4.2 Descriptive Statistics

This section focuses on using descriptive statistics on our data gathered from responses to each questionnaire item in order to assess and interpret the results for our survey questions. Descriptive statistics attempts to numerically describe the characteristics of any collected data (survey results in our case). The assessment of the mean, as a representative score for each item, will be based on the value of the standard deviation, while the interpretation of the mean values will be made on the basis of the 7-point Likert scale which represents the relative agreement/disagreement of the participants for each questionnaire item. The goal is to give an indication of the range of responses given the number of response samples that were collected for each question and to summarize the collected data in a meaningful way. This is done by first computing frequencies, mean, and standard deviation of each questionnaire item, and then examining them to extract useful information for the purposes of the research.

In order to ascertain whether or not a given mean is representative of the majority of the responses for a given questionnaire item, we first need to look at the standard deviation for that item. If the standard deviation is small, it indicates that the individual responses are clustering closer to the mean which shows that the mean is an accurate representation of the majority of the responses (Field 2009). However, if the standard deviation is large, it indicates that individual responses are more scattered and therefore we need to look into additional information such as the frequency of scores to be able to better interpret the nature of the responses given for that questionnaire item.

The used Likert scale rating is: 7 for Strongly Agree, 6 for Agree, 5 for Somewhat Agree, 4 for Neutral, 3 for Somewhat Disagree, 2 for Disagree and 1 for Strongly Disagree. We have also ranked items within each table based on the value of their mean which means the higher the mean is, the higher the rank will be for that particular item.

5.4.2.1 RQ1-Combining Security & Agility

For the first research question, we have provided participants' agreement/disagreement ratings toward items related to the following proposed solutions to seamlessly combine security and Agility. These solutions are: 1. Having a security engineer within the Agile team, 2. Building software with security in mind, 3. Using software security controls in Agile projects, and 4. The inclusion of experienced developers within the Agile team.

5.4.2.1.1 RQ1A Dedicated Security Engineer

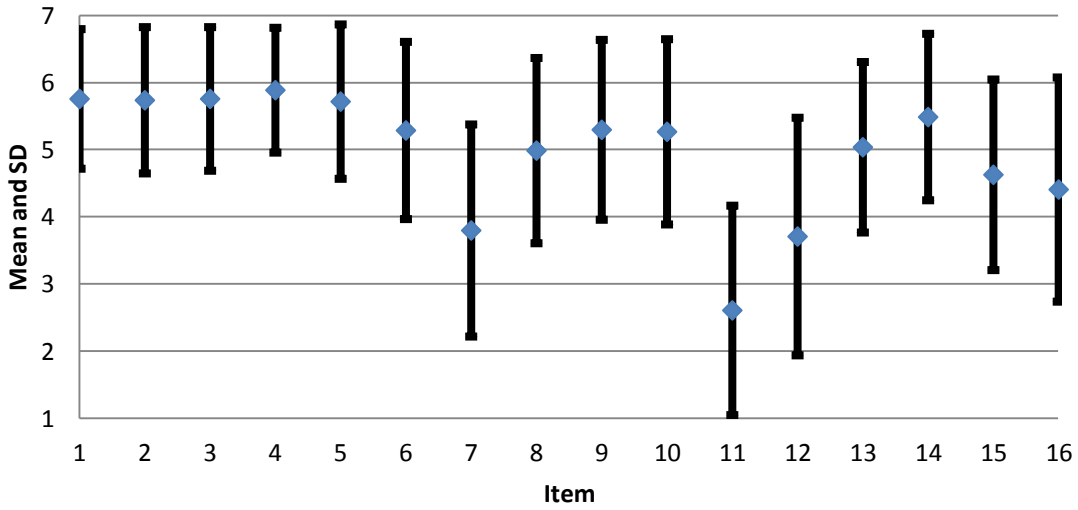


Figure 5-14: 'dedicated security engineer' Items Mean and their Standard Deviation

5.4.2.1.2 RQ1B Software with Security in Mind

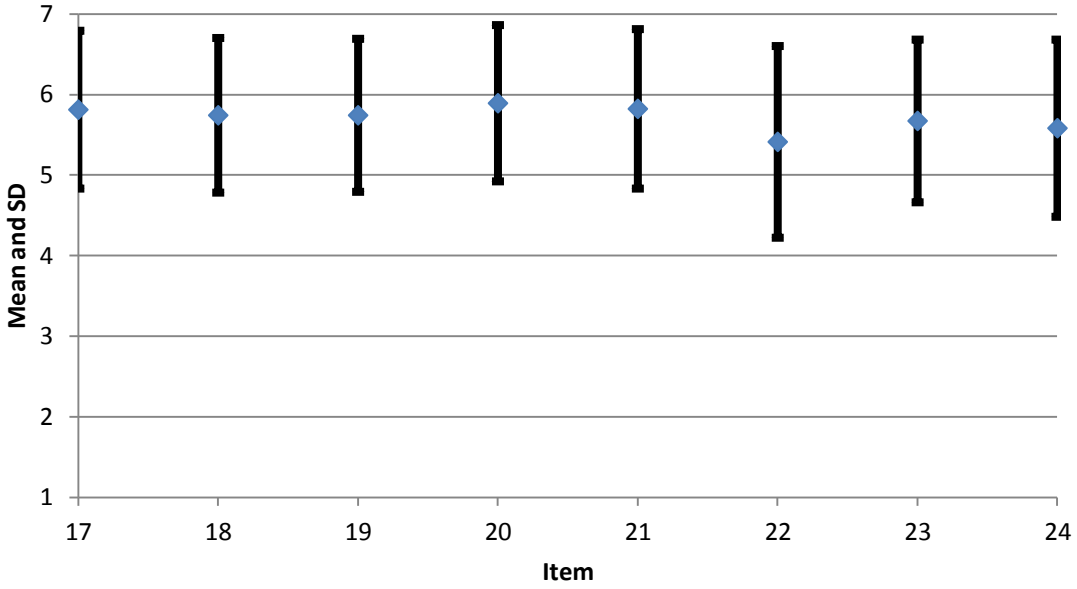


Figure5-15: 'software with security in mind' Items Mean and their Standard Deviation

5.4.2.1.3 RQ1C Security Controls

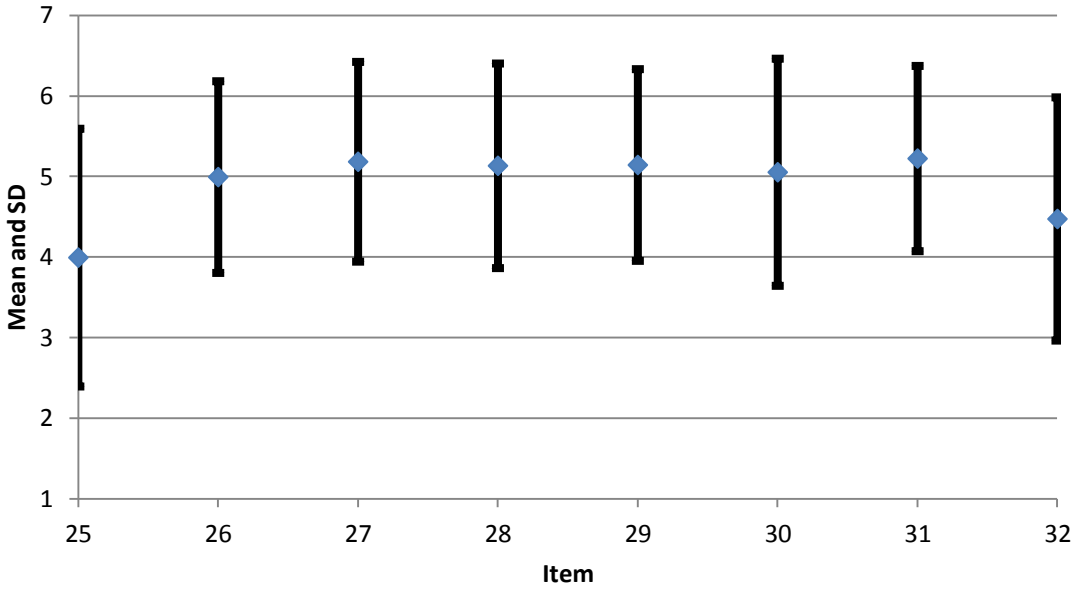


Figure 5-16: 'security controls' Items Mean and their Standard Deviation

5.4.2.1.4 RQ1F Experience of Developers

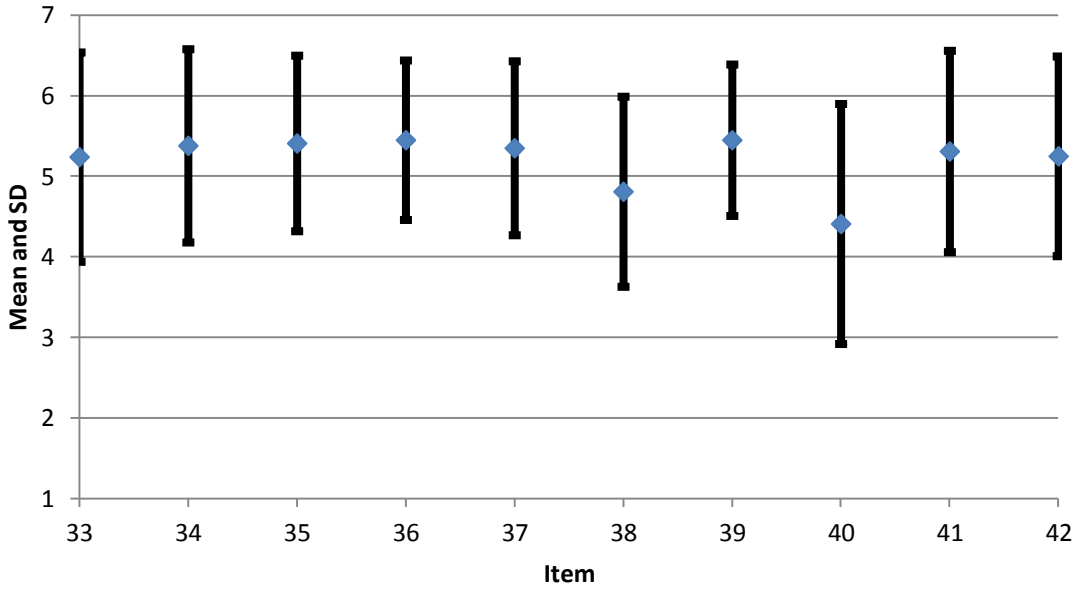


Figure 5-17: 'experience of developers' Items Mean and their Standard Deviation

5.4.2.2 Findings of Descriptive Analyses of RQ1

The mean for Figure 5-14 is representative of the items because of small standard deviations as discussed earlier. Many participants agreed that the existence of the security engineer within the Agile team will help with many aspects of the development process and also with the final product (Items 1, 2, 3, 4, 5, 6, 8, 9, 10, 13, 14, and 15). They disagreed however, with the idea that having a security engineer within the Agile team decreases the need for highly experienced developers/architects (item 11) indicated by the mean 2.61 out of 7. Out of 16 questionnaire items, 3 items (items 7, 12, and 16) have a mean close to 4.0 (Neutral) and standard deviation that is relatively large (> 1.5) which shows the range of opinions that exists between the participants and the indifference of the majority regarding these particular statements.

Overall most practitioners agreed that the existence of the security engineer will lead to increased awareness of security within the team as evidenced by its mean of 5.89. On the same note, the next two agreed upon statements which coincidentally had the same

level of agreement (5.76) were: the effect of the security engineer on increasing each team member's security knowledge and his help in solving security issues faster which is the security engineer's primary responsibility. This result goes on to further legitimize and substantiate our earlier findings through our targeted and in-depth semi-structured interviews and establish consistency between our earlier findings with a small sample size and the broader opinions of the larger population of practitioners.

All questionnaire items in Figure 5-15 have small standard deviation and mean of 5.4 or more which shows a good strong level of agreement among the majority of participants by most practitioners with high level of consensus regarding the benefits of building software with security in mind in Agile projects (RQ1B).

For RQ1B everyone pretty much agreed with all of our statements as we said earlier which tells us that having security in mind when building software with Agile is beneficial from many different aspects and points of view. Most of the items in Figure 5-16 have a mean of 5 and above which illustrates agreement among participants towards the benefits of using software security controls in certain aspects of the project (and the resulting software) that were surveyed (items 26, 27, 28, 29, 30, and 31). However, two items (items 25 and 32) have a mean close to 4 (Neutral) and standard deviation that is relatively large (>1.5) which indicates less consensus among practitioners when it comes to the developers' focus on security when security controls are used.

Two major themes are evident from RQ1C which tells us that practitioners like the idea of security controls because they agree (mean bigger than 5) it can help them increase their software's reusability (items 26, 27, and 29) and at the same time reduce costs and weaknesses and vulnerabilities (items 30, 31).

Most of the items in figure 5-17 have small standard deviation and mean of 4.81 and above which illustrates the practitioners fairly agree with good consensus on what experienced developers can achieve for the sake of security (items 33, 34, 35, 36, 37, 38, 39, 41, and 42). However, one item (item 40) has relatively large standard deviation with mean of 4.41 which indicates less consensus among participants regarding the

benefits of having experienced developers with the Agile team compared to QA when it comes to security focused testing.

Among the various security activities experienced developers can be involved in is their contribution in adding overall security awareness to the team as well as in increasing the overall software security as supported by their means of 5.45 which indicates a strong level of agreement by most practitioners with high level of consensus. These are followed by the experienced developer's help in implementing security requirements faster which had a mean of 5.41 and also a small standard deviation which indicates strong consensus.

5.4.2.3 RQ2-Change Agile Practices for Security

The second research question relates to issues about whether changing Agile practices for sake of security is really needed or not. These issues are: 1. A comparison between Traditional Waterfall and Agile methods in terms of security, 2. Awareness of security in Agile, 3. Impact of accelerated schedules in Agile projects, and 4. Reduced security for internal Agile projects.

5.4.2.3.1 RQ2A Agile vs. Traditional Waterfall

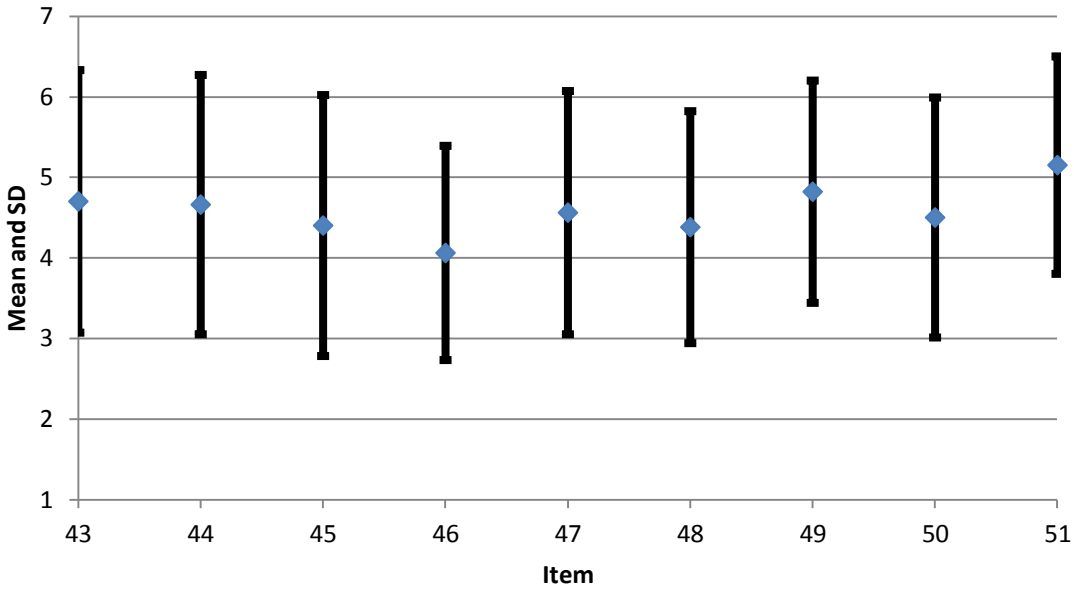


Figure 5-18: 'Agile vs. Waterfall' Items Mean and their Standard Deviation

5.4.2.3.2 RQ2B Awareness of Security to Agile

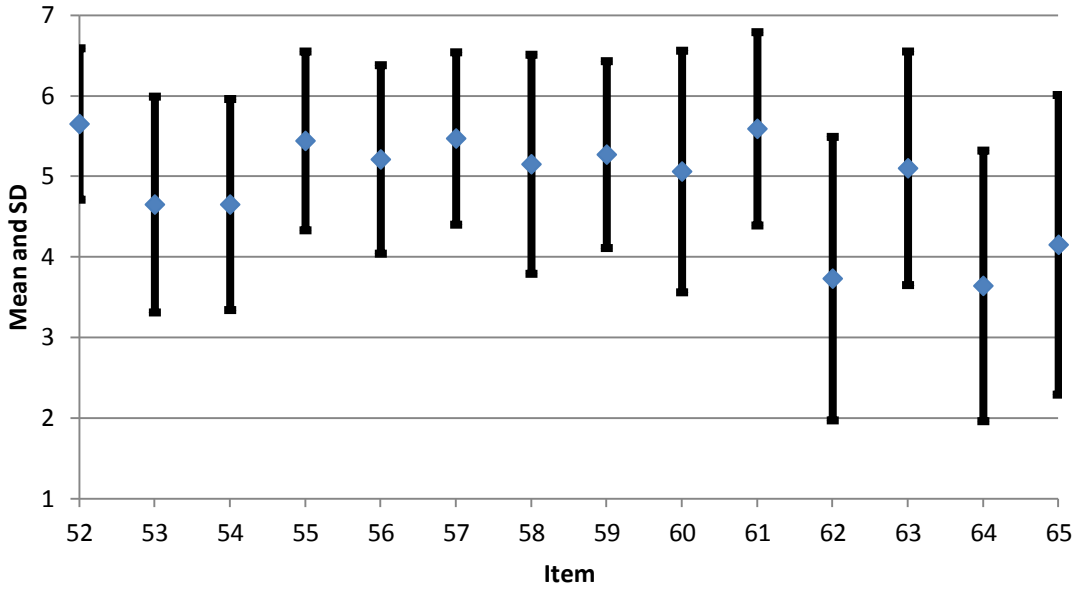


Figure 5-19: 'awareness of security to Agile' Items Mean and their Standard Deviation

5.4.2.3.3 RQ2C Impact of Accelerated Schedule

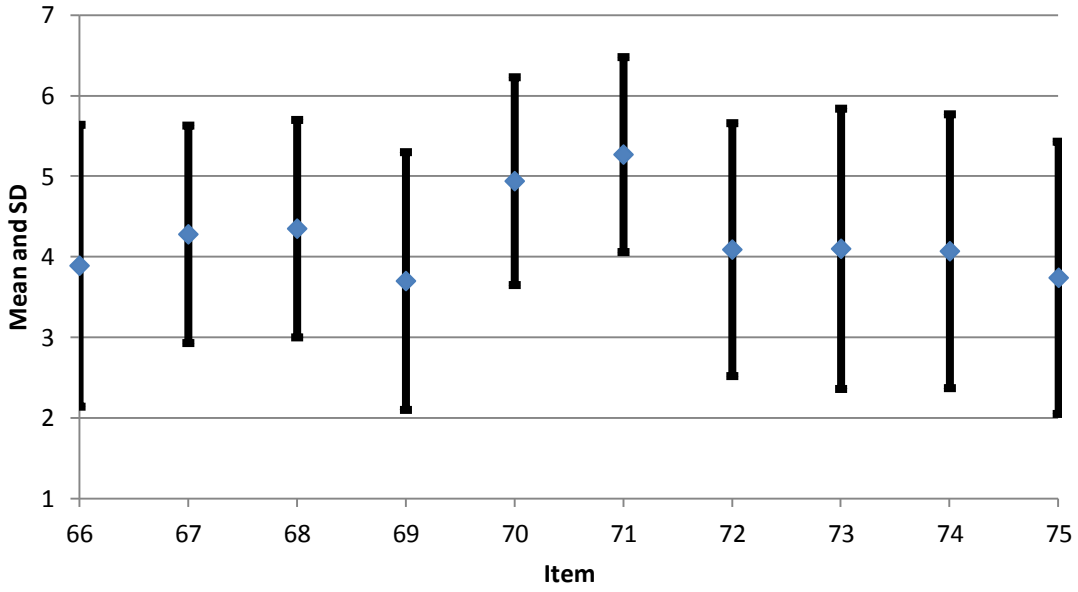


Figure 5-20: 'impact of accelerated schedule' Items Mean and their Standard Deviation

5.4.2.3.4 RQ2D Reduced Security for Internal Projects

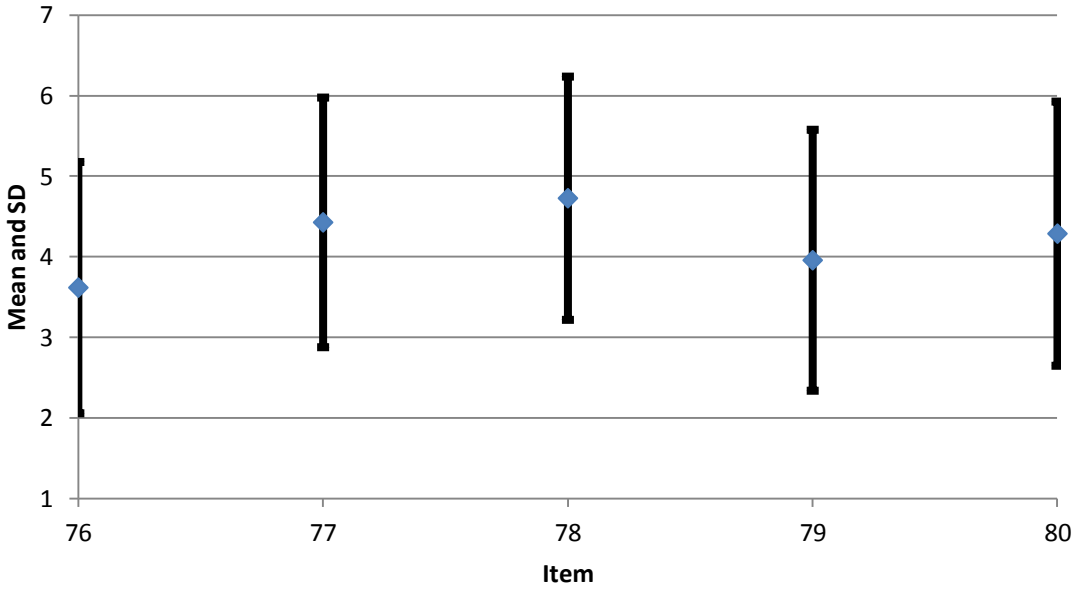


Figure 5-21: 'reduced security for internal projects' Items Mean and their Standard Deviation

5.4.2.4 Findings of Descriptive Analyses of RQ2

In Figure 5-18, we asked 9 questions from practitioners aimed at comparing Agile with traditional Waterfall in terms of security, 3 items (items 43, 44, and 45) were reverse-phrased items and therefore we decided to subtract their mean from 8 in order to turn them into positive questions for the purposes of analysis and interpretation.

Since the standard deviation for items 43, 44, 45, 47, and 50 are relatively high (>1.5), we can say that the mean is less accurate for these specific items and therefore we would rely on percentage of respondents' scores for the interpretation. The way we achieve this is by combining the 3 levels of agreement (and disagreement) into a larger category of "Agree" or "Disagree" for the purposes of comparison and contrast.

While many practitioners prefer to choose neutral position in answering most of the items regarding this issue, the majority of the respondents agreed that Agile was not less secure than traditional Waterfall in terms of security (items 43, 44, 45, 46, and 47). This level of agreement increases in the item that stated Agile was better in terms of providing more opportunities to mitigate security issues (item 49) and finding security related issues faster (item 51) than traditional Waterfall.

The respondents agree (Mean of 5.21 or more) that there are benefits in awareness of security in Agile projects (item 52, 57, 55, 59, and 56). They also agreed that organizational support (item 61) and to a lesser degree organizational maturity (item 60) are required for this to be accomplished. Surprisingly, many respondents of this questionnaire agreed on the importance of the awareness of security in all Agile projects (item 63 with mean of 5.10) and not only in some projects (item 64 with mean of 3.64) or in projects that require security (item 65 with mean of 4.15)

As explained earlier, since item 66 and 69 are reverse-phrased items, we decided to subtract their mean from 8 in order to turn them into positive questions for the purposes of analysis and interpretation.

The Figure 5-20 shows reduced consensus in responses to many items that attempt to address the impact of accelerated schedules of Agile projects (items 66, 69, 72, 73, 74, 75) on the project and the resulting software. However, there's agreement from the respondents in responses to two items (items 70, 71). The highest among the two is the acknowledgement that accelerated schedules of agile projects affect every aspect of the project including security (item 71) as supported by its mean of 5.27 followed by whether accelerated schedules of agile projects are affected by security requirements (item 70) with a mean of 4.94.

In Figure 5-21, for the issue of reduced security for internal Agile projects, practitioners agreed that internal Agile projects are less concerned with security (item 77), can have less security requirements (item 78), and can have less security if they're not exposed publically (item 80). They rejected however, the idea of relying on the infrastructure for security in internal Agile projects (item 76). For comprehensive and detailed descriptive statistics tables of these questionnaire items refer to appendix E.

5.5 Reliability of Scale

Reliability of scale is a way to measure how consistent the scale is in representing a construct in general and a proposed solution/issue more specifically (Field 2009). Another way to check reliability is to see if our questionnaire is producing similar results when used by the same participants and under a similar condition. According to Peterson (1994), any scale has to be reliable in order for the scale to be valid and has a practical benefit (Peterson 1994).

In this survey, the questionnaire is mainly divided into two sections. Each one of them has four scales to measure an issue or a proposed solution presented in our empirical investigation. First section has 4 scales that attempt to measure the suitability of employing one or more of the solutions to seamlessly combine security and agility. For the second section, there are 4 scales that attempt to measure issues that help in understanding whether changing agile for the sake of security is needed or not.

To make sure that these scales are consistently and accurately reflecting the constructs (whether they are proposed solutions or issues) that they are measuring, an evaluation of internal consistency and item-total correlation were conducted in order to assess the reliability for each scale and for each item within that specific scale.

5.5.1 Internal Consistency

In order to measure the reliability of a scale, we need to look at how consistent responses or rating scores are across questionnaire items within a specific scale. This is usually referred to as "internal consistency". The most widely used technique to

measure scale reliability is Cronbach's Alpha (Cortina 1993). A low value of Cronbach's alpha indicates that the scale is unreliable which means that it performs badly in measuring the construct. However, Cronbach's Alpha coefficient of 0.70 and above is considered an acceptable threshold of reliability (Field 2009). Some researchers went as far as saying that a coefficient between 0.60 and 0.70 is the minimum limit of acceptance (Hair, Black et al. 2006; Pallant 2007).

Research Question	Issue	No. of Items	Alpha before reversing	Alpha after reversing
Combining Security & Agility	Having a Security Engineer within the agile team can	16	0.89	0.89
	Building Software with Security in mind in agile projects can	8	0.92	0.92
	Using Software Security Controls (pre-made security modules) in the agile project can	8	0.91	0.91
	Experienced Developers within the agile team can	10	0.91	0.91
Change Agile Practices for Security	Compared to Traditional Waterfall, Agile methodologies	9	0.64	0.81
	Awareness of security in Agile Projects	14	0.77	0.77
	Accelerated Schedules of Agile Projects	10	0.65	0.77
	Internal Agile Projects	5	0.73	0.73

Table 5-3: Reliability before and after reversing item 43,44,45,66, and 69

As we have identified earlier in the previous section, there are five items with reverse-phrased; 3 items (items 43, 44, and 45) for the issue of Agile vs. traditional Waterfall (RQ2A) and 2 items (items 66 and 69) for impact of accelerated schedules in Agile projects (RQ2C). The goal of including reverse-phrased items is to minimize the response bias from participants (Field 2009) which helps in collecting more accurate and thoughtful responses. In the scale reliability analysis, reverse-phrased items usually impact the Cronbach's Alpha coefficient negatively. Therefore we conducted the reliability analysis before and after reversing these items. Table 5-12 shows the Cronbach's Alpha coefficients for the 8 scales. Reversing these items would result in significant increase in alpha coefficient from 0.64 to 0.81 for Agile vs. Waterfall and 0.65 to 0.77 for accelerated schedules of Agile projects. Overall, the Alpha coefficient

values are between 0.73 and 0.92 which suggest very good consistency for the scale. Then, we can conclude that each scale is consistently reflecting the solution/issue that it is measuring.

5.5.2 Item-total Correlations

The item-total correlation refers to the correlation between an item (a statement in our case) with the total score of all items in that scale. The goal is to see if the item or statement is positively contributing to the scale that it is measuring. If all items are measuring the same construct/issue, the correlation between each item and that construct/issue should be high (Churchill 1979; Briggs and Cheek 1986). According to Churchill (1979), item-total correlation analysis would help in filtering the measurement of the construct/issue by taking out the "garbage items" before conducting any further analysis on the items.

In our survey, the questionnaire is mainly divided into two sections, to each of which we would apply item-total correlation independently. The reason for this is because each section tries to answer a different research question. For the first section, we used the traditional cut-off value for assessing item-total correlation which is 0.50 (Green, Tull et al. 1988; Lu, Lai et al. 2007), therefore any item/statement that is less than the said cut-off value will be eliminated from further analyses.

Issue	No	correlation coefficient	No	correlation coefficient	No	correlation coefficient
Having a Security Engineer within the agile team can…	1	0.777**	7	~~0.237~~**	13	0.728**
	2	0.772**	8	0.694**	14	0.724**
	3	0.667**	9	0.733**	15	0.630**
	4	0.739**	10	0.739**	16	~~0.474~~**
	5	0.777**	11	~~0.415~~**		
	6	0.655**	12	0.507**		
Building Software with Security in mind in agile projects can…	17	0.820**	20	0.814**	23	0.784**
	18	0.849**	21	0.841**	24	0.816**
	19	0.839**	22	0.722**		
Using Software Security Controls (pre-made security modules) in the agile project can…	25	0.627**	28	0.832**	31	0.793**
	26	0.881**	29	0.857**	32	0.696**
	27	0.863**	30	0.832**		
Experienced Developers within the agile team can…	33	0.636**	37	0.854**	41	0.804**
	34	0.792**	38	0.644**	42	0.806**
	35	0.785**	39	0.871**		
	36	0.786**	40	0.653**		

* Correlation is significant at the 0.05 level (2-tailed).
** Correlation is significant at the 0.01 level (2-tailed).
Note: stroked values are the items that have been dropped

Table 5-4 : Correlation Analysis for RQ1

For the first research question, Table 5-13 shows 3 items (items 7, 11, and 16) that have item-total correlations less than 0.50. Therefore, we decided to drop these items from the subsequent analyses on this section.

Issue	No	correlation coefficient	No	correlation coefficient	No	correlation coefficient
Compared to Traditional Waterfall, Agile methodologies…	43	0.729**	46	0.576**	49	0.747**
	44	0.702**	47	0.357**	50	0.755**
	45	0.406**	48	0.681**	51	0.747**
Awareness of security in Agile Projects…	52	0.493**	57	0.576**	62	0.508**
	53	0.629**	58	0.508**	63	~~0.115~~
	54	0.564**	59	0.646**	64	0.478**
	55	0.590**	60	0.540**	65	0.503**
	56	0.533**	61	0.588**		
Accelerated Schedules of Agile Projects…	66	0.532**	70	0.437**	74	0.832**
	67	~~0.025~~	71	0.621**	75	0.854**
	68	~~0.045~~	72	0.867**		
	69	0.462**	73	0.868**		
Internal Agile Projects…	76	0.594**	78	0.763**	80	0.791**
	77	0.728**	79	0.599**		

* Correlation is significant at the 0.05 level (2-tailed).
** Correlation is significant at the 0.01 level (2-tailed).
Note: stroked values are the items that have been dropped

Table 5-5: Correlation Analysis after reversing item 43,44,45,66, and 69 for RQ2

The minimum recommended cut-off item-total correlation value is 0.3 (Nunnally and Bernstein 1994; Field 2009). Since this questionnaire is mainly divided into two sections which answer two different research questions and since the overall strength of item-total correlation values for first section (combining security & Agility) is bigger than second section (change Agile practices for security), we opted to choose two different cut-off values for each section.

For the second section, keeping to the same traditional cut-off value would not produce satisfactory analysis results, therefore we decided to use the minimum recommended cut-off item-total correlation value which allowed us to take the same number of items (3 items) from this section as we did in the first section. Table 5-14 shows 3 items (items 63, 67, and 68) that have the lowest item-total correlation. As a result, we decided to drop them from further analyses on this section.

5.6 Factor Analysis

After the assessment of scale reliability through internal consistency and item-total correlation, we employed factor analysis in order to extract factors (latent variables or dimensions) that exist within our questionnaire data. The main purpose of using factor analysis is to allow us to assess the extent to which the constructed questionnaire items (statements in our case) are reflecting an underlying trait that they are supposed to measure. This is sometimes referred to as "factorial validity" (Bryman 2001; Field 2005). It also helps in reducing a large number of questions to a manageable size of variables. This is usually done for exploratory purposes where the researcher cannot clearly anticipate what factors could be extracted from the data.

In our survey, the questionnaire is mainly divided into two sections, to each of which we would apply factor analysis independently. The reason of this is because each section tries to answer a different research question. For the first section, factor analysis allows us to understand what it means for a solution to be suitable for inclusion into Agile. In order to accomplish this, this analysis gives us a method to be able to understand the overall structure of the underlying latent variables and be able to measure the effects of such variables on the process and finally to be able to gain valuable information from the analysis to help reach a conclusion about the question of suitability. To conduct factor analysis on the first section, 42 questions were asked from practitioners minus 3 items filtered through item-total correlation. The remaining 39 questions were analyzed in order to extract factors that measure the suitability of employing 4 different solutions to combine security and Agility. Similarly for the second section, 38 questions were asked from practitioners to measure issues that help in understanding whether changing agile for the sake of security is needed or not. We also removed 3 items for the second section using the same method and the remaining 35 questions were used for factor analysis.

5.6.1 Assessment of Data Factorability

Factorability of data means that the data is appropriate and suitable for factor analysis in terms of the correlation between variables (Pallant 2007; Tabachnick and Fidell 2007). By measuring the correlation between variables, factor analysis aggregates

variables that are considered to measure aspects of underlying factors. For this to happen, the correlation matrix between variables is needed to have sizeable correlation values (Tabachnick and Fidell 2007; Field 2009). To decide whether the correlation matrix is factorable or not, Kaiser-Meyer-Olkin's (KMO) sampling adequacy test, and Bartlett's sphericity test are usually employed (Pallant 2007; Field 2009).

KMO and Bartlett's Test		
Kaiser-Meyer-Olkin Measure of Sampling Adequacy.		.875
Bartlett's Test of Sphericity	Approx. Chi-Square	4242.086
	df	741
	Sig.	.000

Table 5-6: KMO and Bartlett's Test for RQ1

KMO and Bartlett's Test		
Kaiser-Meyer-Olkin Measure of Sampling Adequacy.		.780
Bartlett's Test of Sphericity	Approx. Chi-Square	3086.919
	df	703
	Sig.	.000

Table 5-7: KMO and Bartlett's Test for RQ2

Kaiser-Meyer-Olkin (KMO) values usually fall between 0 and 1 (Kaiser 1970). He recommended a KMO value to be at a bare minimum of 0.5. He also classified the range of KMO values into 4 categories. Values between 0.5 and 0.7 are mediocre, values between 0.7 and 0.8 are good, values between 0.8 and 0.9 are great, and values above 0.9 are superb (Kaiser 1974). Tables 5-15 and 5-16 present a KMO value of 0.875 for Combining Security and Agility and 0.780 for Change Agile projects for Security respectively. Both KMO values are well above the minimum limit of acceptance of 0.60 (Tabachnick and Fidell 2007) and therefore show sampling adequacy for both sections.

For Bartlett's test of sphericity, both sections were highly significant at $p < 0.001$. This indicates that the relationships between variables were adequate to be included in the analysis for both sections. Therefore, we can conclude that KMO value and Bartlett's test of sphericity confirm the suitability of the data for factor analysis.

5.6.1.1.1 Factor Extraction Methods

Factor analysis commonly consists of 2 steps: factor extraction and factor rotation. In order to validate our prior expectation of the underlying factors (whether they are proposed solutions to combine security and agility or issues related to changing agile practices for security), factor extraction aims to uncover factors that exist within the collected questionnaire data. In order to extract the right number of factors, two criteria are used in our study to get the expected number of factors from a data set which will be discussed shortly.

5.6.1.2 Extraction Techniques

The choice of extraction technique will be based on the objectives of the analysis and the study. Principal Component Analysis (PCA) and Principal Factor Analysis (PFA) (also called Principal Axis Factoring) are two of the most commonly used factor extraction techniques. PCA is usually employed in exploratory studies that attempt to uncover the underlying latent variables (or factors) of the data set. On the other hand, PFA (or FA for short) is used when the study is designed based on earlier expected factors (Tabachnick and Fidell 2007). For this study we have chosen PCA as our extraction method that will be discussed in detail shortly.

The mathematical difference between PCA and PFA is shown in the values of diagonal elements of the correlation matrix where these values represent the correlation between a certain variable and itself. PCA uses the original correlation matrix in which each value in the diagonal has a value of one and then analyzes a variance resulted from the sum of these elements. Such a technique will allow all the original data (including the unique variance and error of each variable) to be distributed among extracted factors. As a result, PCA will be able to reproduce the exact observed correlation matrix in case no discarded factor exists. On the same note, PFA uses communalities (the percentage of shared variance that exists in a variable, excluding unique variance and error of each variable) as the diagonal element(s) of correlation matrix and then analyzes the variance that resulted from the sum of communalities. This technique will allow the communalities to be distributed among extracted factors. Because of the exclusion of unique variances and errors in the distributed communalities, PFA won't

be able to reproduce the exact observed correlation matrix as in PCA (Tabachnick and Fidell 2007).

According to Guadagnoli and Velicer (1988), the obtained results from PCA are slightly different from the results obtained from PFA (Guadagnoli and Velicer 1988). Stevens (2002) argued that this is not the case in all situations. He concluded it's unlikely to obtain different results when there is 30 or more observed variables and communalities bigger than 0.7 for all variables. However, different results can be found when there is less than 20 observed variables and any communalities lower than 0.4(Stevens 2002).

For our questionnaire, both sections have more than 30 observed variables (39 for first section and 35 for second section) and most of the communalities are greater than 0.4, which allows us to conclude that it's less likely for differences to occur and if any differences occurred they won't be significant which means we can use PCA as our extraction method with no issues.

5.6.1.3 Determining Number of Factors

To determine the number of factors that needs to be extracted, there are several criteria proposed in the literature. These criteria are: Kaiser's criterion, Jolliffe's criterion, Cattell's scree test, a priori criterion, and proportion of variance criterion.

Kaiser's criterion suggested retaining factors with eigen-value bigger than 1 for further analysis (Kaiser 1960) but Jolliffe (1972, 1986) argued that Kaiser's criterion is too rigid and proposed another option of retaining all factors with eigen-value bigger than 0.7 (Jolliffe 1972; Jolliffe 1986). Cattell's scree test recommended retaining factors at the point of inflexion. In scree plot graph, the point of inflexion is positioned where the sharp fall of the curve becomes flat (Field 2009). A priori criterion involves specifying the number of factors based on prior anticipation of the underlying factors (Hair, Black et al. 2006). Lastly, proportion of variance criterion suggested retaining factors that explain a predetermined amount of variance or more (i.e. 5% or 10%) in order to ensure the importance of the retained factors for analysis and study (Tabachnick and Fidell

2007). Measuring information that involves humans often lacks the precision and, therefore, researchers usually accept the solution that explains lower than 60% of the total variance (Hair, Black et al. 2006).

In the case of our questionnaire where we are measuring the suitability, a variable that cannot be measured directly, of including different proposed solutions into Agile projects, the priori criterion along with the proportion of variance criterion are used to determine the number of factors.

5.6.2 Factor Rotation

In most cases, interpreting factors right after their extraction is not an easy task. This is because every variable load highly on one factor and not on the other factors. To increase the loading of each variable on its important factor and decrease it on the others, factor rotation is employed. There are two rotation techniques that can be used: Orthogonal rotation (varimax) and Oblique rotation (oblimin). Orthogonal rotation assumes that the extracted factors are independent and don't correlate to each other, while oblique rotation assumes that the extracted factors are related (Tabachnick and Fidell 2007; Field 2009).

According to Field (2009), "In practice, there are strong grounds to believe that orthogonal rotations are a complete nonsense for naturalistic data, and certainly for any data involving humans (can you think of any psychological construct that is not in any way correlated with some other psychological construct?). As such, some argue that orthogonal rotations should never be used." (Field 2009).

Since the data gathered involving humans for both sections, we can conclude that oblique rotation (oblimin) is the most appropriate factor rotation technique to use for both sections of the questionnaire. This choice can be checked by examining the component correlation matrix produced after conducting oblique rotation. It's important to note that oblique rotation produces two matrices: Pattern and Structure matrix. Pattern matrix provides regression coefficient value of every variable on every factor and most researchers use it for interpretation (Field 2009). Structure matrix provides correlation coefficient value of every variable on every factor. If the produced

correlation matrix of factors is not an identity matrix (all correlation coefficients between factors are zero), then we can confidently say that the extracted factors are not independent and therefore it's appropriate to use oblique rotation.

5.6.3 Factor Scores

Factor scores are an estimation of a participant's score on each extracted factor. Calculating factor scores allows us to conduct further analysis such as t-test or ANOVA on the produced factor scores rather than going back to the original data (Field 2009).
Through the use of factor scores we can compare mean scores of suitability of all proposed solutions to combine security and agility. This will help us find which solution is significant in terms of suitability of inclusion by conducting repeated-measures ANOVA on the calculated factor scores. In addition, it will help us to test 4 hypotheses related to the first section which is Combining Security & Agility.

There are two techniques for calculating factor scores: weighted average and regression method. Weighted average (or weighted sum score) is a technique that uses factor loadings (obtained from pattern matrix in case of oblique rotation) to calculate scores of a certain factor. The weighted average WA_{ai} is computed by adding up the results of multiplying factor loadings A_{iv} (or weight) *v* on this factor *i* with the obtained participant's rating score S_{av} on same variable *v* (7-point Likert Scale in our case) for each variable (DiStefano, Zhu et al. 2009; Field 2009). The equation that is used to calculate weighted average factor scores is:

$$WA_{ai} = \sum_{v=1}^{n} S_{av} A_{iv}$$

WA_{ai}: Weighted average score of a participant *a* toward factor *i*
S_{av}: Participant *a*'s score on variable *v*
A_{iv}: Factor loading of variable *v* on factor *i*
n : Number of variables included in factor analysis

Field (2009) argued that in order to compare weighted average factor scores of different factors, the measurement scales used to measure variables (or statements in our case)

have to be the same (Field 2009). Since all our statements are using 7-point Likert scale, we can confidently assume that we are complying with this requirement.

Another technique used to calculate factor scores is the regression method. Regression method is a technique that takes into account the correlations between variables when calculating factor scores. This can be done by calculating a new matrix called factor score coefficients that act as factor loadings in the weighted average. The equation to calculate factor score coefficient matrix is:

$$B = R^{-1}A$$

B: factor score coefficients matrix
R^{-1}: inverse correlation matrix
A: factor loading matrix

Since the correlations between variables are considered in the calculation, the obtained factor scores from the regression method are much lower than the factor scores from the weighted average method. A characteristic of regression factor scores is that the mean of all scores is zero.

Since we are planning to compare mean of factor scores, using either the regression method or its extensions (Bartlett method and Anderson-Rubin) are not applicable. This is because they produce factor scores with mean of zero and therefore they are not useful for our analysis. As a result, we decided to use the weighted average as our method to produce factor scores.

5.6.4 Result and Interpretation

5.6.4.1 Combining Security & Agility

For the first research question, we will provide the results obtained from factor analysis on variables (or questionnaire items) related to the proposed solutions to seamlessly combine security and Agility. Before interpreting the obtained results, the procedures we followed for conducting factor analysis are presented here. It's important to note that we have codified variables/items to represent the scale that we are expecting to measure. Table5-17 shows how the codification relates the items. This codification will help in grouping related items that are used to interpret the extracted factors.

Scale	Code	Item number
Security Engineer	SecEng1-16	Item1-16
Security in Mind	SecInMind1-8	Item17-24
Software Security Controls	SecCon1-8	Item25-32
Experienced Developers	ExpDev1-10	Item33-42

Table 5-8: Codification for Scale items (Combining Security & Agility)

Out of the 42 initial questionnaire items for section 1, 39 items (or statements) were included in the factor analysis for this section. A PCA extraction technique was applied to extract the underlying factors. To determine the number of factors to be extracted, we first conducted factor analysis using the Kaiser's criterion. Appendix F contains the results of factor analysis when using Kaiser's criterion. Additionally, we decided to suppress any factor loading with absolute value less than 0.30 from being displayed in the final results. This is because researchers, in general, consider any factor loadings with absolute value bigger than 0.30 to be significant (Field 2009).

As explained earlier, we are following an a priori and proportion of variance criteria to determine the number of factors to be extracted. While Kaiser's criterion supports extracting 6 factors, our prior expectation when we designed this questionnaire suggested extracting 4 factors. This corroborated with the predetermined amount of variance which is 5% in our case or more to ensure the importance of the retained

factors for the analysis. Table 5-18 shows that the 5th and 6th factor explain less than 5% of the total variance and therefore, we decided to specify the number of factors to be extracted from conducting factor analysis to be 4 factors. Appendix F contains the results of factor analysis when extracted number of factors is four and suppressing factor loadings less than 0.30. It's important to note that there is no dropped item after forcing the analysis to extract four factors and the total variance explained is 62.26%.

Total Variance Explained							
Component	Initial Eigenvalues			Extraction Sums of Squared Loadings			Rotation Sums of Squared Loadings
	Total	% of Variance	Cumulative %	Total	% of Variance	Cumulative %	Total*
1	14.571	37.362	37.362	14.571	37.362	37.362	10.336
2	4.549	11.663	49.025	4.549	11.663	49.025	7.769
3	2.839	7.278	56.304	2.839	7.278	56.304	7.491
4	2.324	5.958	62.262	2.324	5.958	62.262	8.933
5	1.684	4.319	66.581	1.684	4.319	66.581	4.108
6	1.099	2.819	69.399	1.099	2.819	69.399	1.333
Extraction Method: Principal Component Analysis.							
* When components are correlated, sums of squared loadings cannot be added to obtain a total variance. Rotation Method: Oblimin with Kaiser Normalization.							

Table 5-9: Total Variance Explained without specifying number of factors (RQ1)

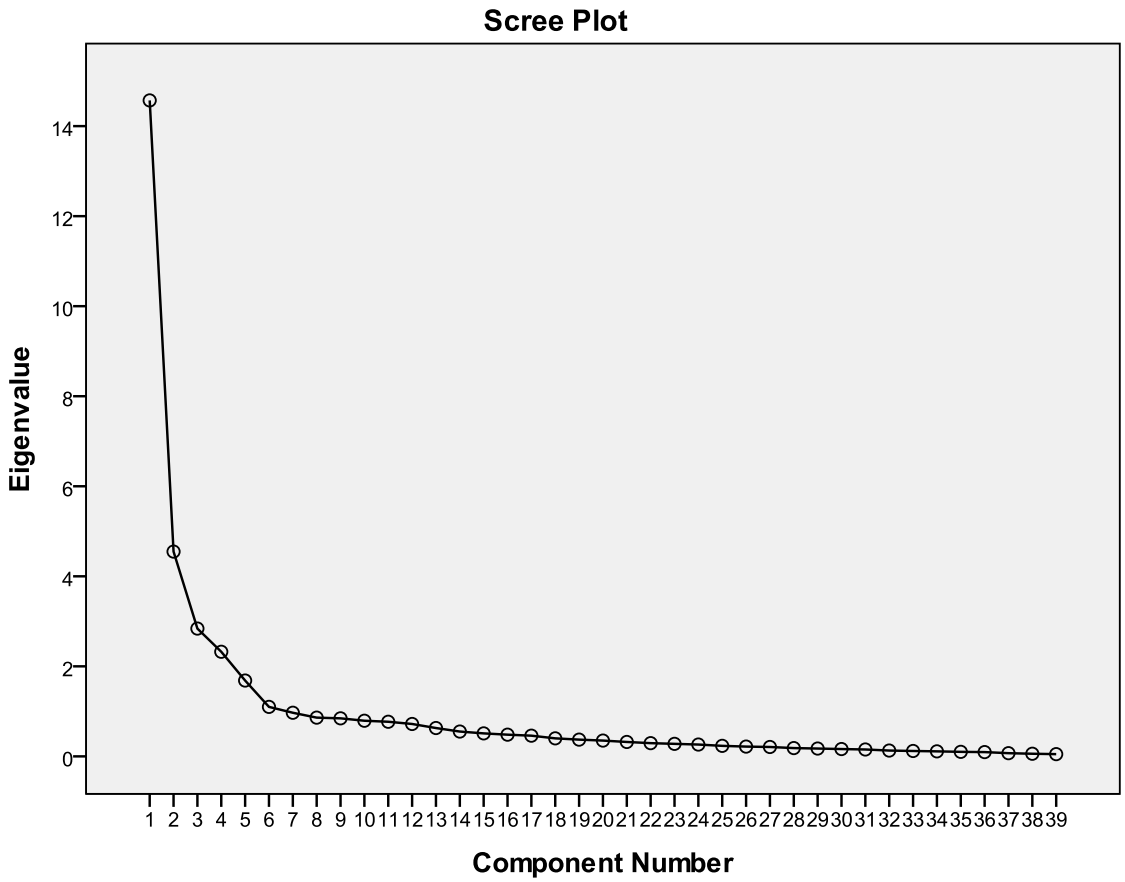


Figure 5-22: Scree Plot (Combining Security & Agility)

Total Variance Explained							
Component	Initial Eigenvalues			Extraction Sums of Squared Loadings			Rotation Sums of Squared Loadings
	Total	% of Variance	Cumulative %	Total	% of Variance	Cumulative %	Total*
1	14.571	37.362	37.362	14.571	37.362	37.362	11.556
2	4.549	11.663	49.025	4.549	11.663	49.025	8.567
3	2.839	7.278	56.304	2.839	7.278	56.304	9.258
4	2.324	5.958	62.262	2.324	5.958	62.262	4.640
Extraction Method: Principal Component Analysis.							
a. When components are correlated, sums of squared loadings cannot be added to obtain a total variance. Rotation Method: Oblimin with Kaiser Normalization.							

Table 5-10 : Total Variance Explained with 4 Factors (Combining Security & Agility)

Statement Code	Statement	Factor Loading*
Factor 1: Having a Security Engineer within the agile team can		
SecEng5	Uncover security issues earlier in the project	.890
SecEng4	Increase awareness of security issues	.862
SecEng1	Increase each team member's security knowledge	.859
SecEng14	Put more importance on security for the project and the team	.784
SecEng3	Help solve security issues faster	.754
SecEng9	Help the project achieve compliance with security standards faster	.752
SecEng2	Help in producing more secure software	.752
SecEng13	Maximize the precision of security requirement assessment	.738
SecEng10	Lower the security risk of the produced software further than developers alone can	.717
SecEng8	Help Establish/Increase the overall security assurance of Agile	.706
SecEng6	Help the customer make better security decisions	.646
SecEng15	Mitigate the lack of mature security vulnerability assessment tools	.471

*Extraction Method: Principal Component Analysis.
Rotation Method: Oblimin with Kaiser Normalization.

Table 5-11: Statements and their Loadings of Extracted Factor 1 (RQ1)

Statement Code	Statement	Factor Loading*
Factor 2: Experienced Developers within the agile team can		
ExpDev7	Increase the overall security of the software	.879
ExpDev5	Mitigate more security issues than regular developers	.861
ExpDev9	Play a key role in avoiding to introduce security vulnerabilities into software	.827
ExpDev10	Predict security vulnerabilities at earlier stages of the project	.810
ExpDev4	Add to the team's overall security awareness	.779
ExpDev3	Help implement security requirements faster	.745
ExpDev2	Discover more security issues/vulnerabilities/bugs than regular developers	.740
ExpDev6	Be more effective than using tools on security	.703
ExpDev8	Be more beneficial than QA when it comes to security focused testing	.662
ExpDev1	Create/write more secure user stories	.590

*Extraction Method: Principal Component Analysis.
Rotation Method: Oblimin with Kaiser Normalization.

Table 5-12: Statements and their Loadings of Extracted Factor 2 (RQ1)

Statement Code	Statement	Factor Loading*
Factor 3: Using Software Security Controls (pre-made security modules) in the agile project can		
SecCon4	Become a part of the framework/infrastructure	.877
SecCon3	Be reused on multiple projects	.860
SecCon5	Be available as part of the infrastructure before the next project begins	.854
SecCon6	Reduce costs if used on multiple projects	.851
SecCon2	Be beneficial throughout the project	.788
SecCon8	Reduce reliance on individuals within the team for security expertise	.672
SecCon7	Help the resulting software to have reduced weaknesses/vulnerabilities	.656
SecCon1	Alleviate developers from having to focus on security	.654

* Extraction Method: Principal Component Analysis.
Rotation Method: Oblimin with Kaiser Normalization.

Table 5-13: Statements and their Loadings of Extracted Factor 3 (RQ1)

Statement Code	Statement	Factor Loading*
Factor 4: Building Software with Security in mind in agile projects can		
SecInMind6	Promote the collective ownership of security by all stakeholders	-.601
SecInMind4	Expand the awareness of security throughout the project	-.594
SecInMind3	Help in achieving security focused testing	-.562
SecEng12	Alleviate other team members from having to focus on specific security issues	.558
SecInMind2	Help in writing more secure code	-.556
SecInMind5	Put more overall focus on security	-.501
SecInMind8	Reduce security risk to the project and the resulting software	-.473
SecInMind1	Establish/Increase the Security Assurance for the project	-.463
SecInMind7	Help in decreasing vulnerabilities in the software	-.370

* Extraction Method: Principal Component Analysis.
Rotation Method: Oblimin with Kaiser Normalization.

Table 5-14: Statements and their Loadings of Extracted Factor 4 (RQ1)

Factor 1: The Suitability of Including Security Engineer into the Agile Team

Among the various security activities a Security Engineer can be involved in within the Agile team, his contribution in discovering security issues earlier in the project turned out to be the most influential indicator. This is followed by the Security Engineer's role in increased security awareness and knowledge within the team and their effect in increasing the security knowledge of other team members as other important contributions that the addition of a Security Engineer can bring to an Agile team. This allows us to conclude given what we know of the Security Engineer that they can indeed have a positive effect on the process as well as the final produced software. This factor corresponds directly to hypothesis H1a (see Table 4-21).

Factor 2: The Suitability of Including Experienced Developers into the Agile Team

The results of adding experienced developers to the Agile team suggests they could increase the overall security of the software. By putting their experiences and knowledge into action, the experienced developers would help the development process in many ways. They could bring influence in mitigating more security issues, avoiding the introduction of security vulnerabilities into software, and prediction of security vulnerabilities at earlier stages of the project. In addition, the interactions between experienced developers with other team members would help in spreading their experiences and knowledge, including the ones related to security, among all members making them aware of many possible security issues that may arise during development and testing. What this suggests is, the addition of the experienced developer into the Agile team brings the above important benefits in terms of security into the Agile process. This factor corresponds directly to hypothesis H1d.

Factor 3: The Suitability of Including Software Security Controls into the Agile Team

From amongst the possible benefits and uses of software security controls within the Agile project, the most important benefit turned out to be for them to become part of the existing framework and/or infrastructure already in place within the organization. This also allows these modules (or components) to be reused on other similar projects with little to no modification which is the second most influencing indicator for its suitability of inclusion into Agile projects. The above combination of benefits naturally reduces

the overall cost of integrating security into software a more cost effective solution than building security into each software module separately. Surprisingly, their help in reducing software weakness/vulnerabilities and alleviating team members from having to focus on security turned out to be the least influential indicator for its suitability of inclusion into Agile. Since many software projects are facing similar security issues, the possibility of reusing software security controls on multiple projects would lead to reduced time and cost needed to handle these issues properly which has a positive effect on the Agile process and the produced software overall. This factor corresponds directly to hypothesis H1c.

Factor 4: The Suitability of Building Software with Security in Mind in Agile Projects

Encouraging the collective ownership of security by all stakeholders along with expanding security awareness throughout the project turned out to be the most important indicators from amongst the benefits that can be gained by building software with security in mind. The least influential indictor was, unexpectedly however, the role of having security in mind in helping to decrease vulnerabilities in the software. While all statements regarding the benefits of considering security at each step of the Agile process are clustered in this factor, one statement related to Security Engineer was grouped into this factor by SPSS which is not related to this factor (it goes against the other statements in the same factor) which is supported by negative factor loading this statement has to the overall structure of this factor (De Vaus 2002). This factor corresponds directly to hypothesis H1b.

Overall, the obtained factors for this section were in agreement with what we expected when we measured the suitability of including different proposed solutions to combine security and agility. As a result, we can confidently state that we achieved factorial validity since all items are measuring what they are supposed to measure.

Moving away from the factor loadings and their interpretation we will now focus on correlations between factors. From the result of the oblique rotation, factor correlation

matrix was obtained which shows that Security in Mind (Component 4) is more independent of any other factors as its correlation coefficients with them are low. However, all other factors correlate positively (between themselves) to a bigger degree. Most notably, Security Engineer (Component 1) with Security Controls (Component 3), and Security Engineer (Component 1) with Experienced Developers (Component 2). This indicates that these factors cannot be assumed unrelated and therefore, the oblique rotation is the most suitable rotation technique for our data.

Component Correlation Matrix				
Component	1	2	3	4
1	1.000	.343	.503	-.213
2	.343	1.000	.270	-.209
3	.503	.270	1.000	-.117
4	-.213	-.209	-.117	1.000
Extraction Method: Principal Component Analysis. Rotation Method: Oblimin with Kaiser Normalization.				

Table 5-15 : Factors/Component Correlation Matrix (Combining Security & Agility)

5.6.4.1.1 Factor Scores

Since the obtained factors for first section were found to measure the suitability of including different proposed solutions to combine security and agility, weighted average factor scores were calculated for each participant to measure the suitability of a particular security solution's inclusion into Agile.

The following table provides the factor loading of each variable on each extracted factor which we used to calculate weighted average factor scores.

Pattern Matrix[a]					
Item No.	Indicators	Component			
		Factor 1: Security Engineer	Factor 2: Experienced Developers	Factor 3: Security Controls	Factor 4: Security in Mind
1	SecEng5	.890	-.046	-.072	.013
2	SecEng4	.862	-.122	-.042	-.153
3	SecEng1	.859	-.048	-.033	-.059
4	SecEng14	.784	.008	-.073	-.104
5	SecEng3	.754	-.031	.010	-.032

6	SecEng9	.752	.088	-.037	.109
7	SecEng2	.752	.028	.071	-.131
8	SecEng13	.738	.102	.053	.180
9	SecEng10	.717	.017	.049	.043
10	SecEng8	.706	-.012	.029	-.148
11	SecEng6	.646	.049	.034	-.039
12	SecEng15	.471	.067	.196	.293
13	ExpDev7	-.025	.879	-.046	-.125
14	ExpDev5	.087	.861	-.036	.076
15	ExpDev9	-.068	.827	-.042	-.041
16	ExpDev10	-.058	.810	.046	-.013
17	ExpDev4	-.052	.779	-.023	-.166
18	ExpDev3	-.008	.745	.077	-.090
19	ExpDev2	.158	.740	.029	.053
20	ExpDev6	.033	.703	-.242	.031
21	ExpDev8	-.029	.662	.043	.113
22	ExpDev1	-.060	.590	.162	-.002
23	SecCon4	-.088	-.071	.877	-.215
24	SecCon3	-.027	.000	.860	-.125
25	SecCon5	-.031	.030	.854	-.130
26	SecCon6	-.035	-.035	.851	-.079
27	SecCon2	.135	.026	.788	-.051
28	SecCon8	.051	.083	.672	.317
29	SecCon7	.217	.021	.656	-.076
30	SecCon1	.035	.043	.654	.313
31	SecInMind6	.085	.151	.104	-.601
32	SecInMind4	.286	.019	.251	-.594
33	SecInMind3	.301	.231	.070	-.562
34	SecEng12	.344	.035	.355	.558
35	SecInMind2	.312	.120	.221	-.556
36	SecInMind5	.224	.213	.243	-.501
37	SecInMind8	.204	.185	.276	-.473
38	SecInMind1	.381	.139	.139	-.463
39	SecInMind7	.231	.188	.311	-.370
$\sum_{1}^{39} \lvert F \rvert$		12.496	9.804	9.627	7.998
	Extraction Method: Principal Component Analysis. Rotation Method: Oblimin with Kaiser Normalization.				
	a. Rotation converged in 9 iterations.				

Table 5-16: Factor Loading of each Variable on each Extracted Factor

It's important to note, the factor's direction (whether positive or negative) is arbitrary (De Vaus 2002; Gorsuch 2013). Gorsuch (2013) suggested that if the majority of loadings of a specific factor are negative, then they can be reversed by multiplying them with -1 (Gorsuch 2013). Since most of factor loadings for the 4th factor (Security in Mind) were negative as it appears in table 5-25, we decided to reverse each obtained factor score for the 4th factor. Appendix F contains the calculated weighted average factor scores for all participants in the first section of the questionnaire.

There are noticeable differences in factor scores for a given participant. This is due to the differences in factor loadings (or weights) each variable has on a factor. To adjust the obtained factor scores on different factors to a common scale, we normalize each factor score of a particular factor by dividing it by the total absolute weights of all variables on this factor. The descriptive analysis of factor scores along with their histograms can be seen in the following tables and figures. For more details including the normalized factor scores refer to Appendix F.

Factor	Number of Participants	Mean	Standard Deviation
Security Engineer	130	4.97	0.79
Experienced Developers	130	4.85	0.76
Security Controls	130	4.33	0.83
Security in Mind	130	2.96	0.51

Table 5-17: Descriptive Statistics of Normalized Weighted Average Factor Scores

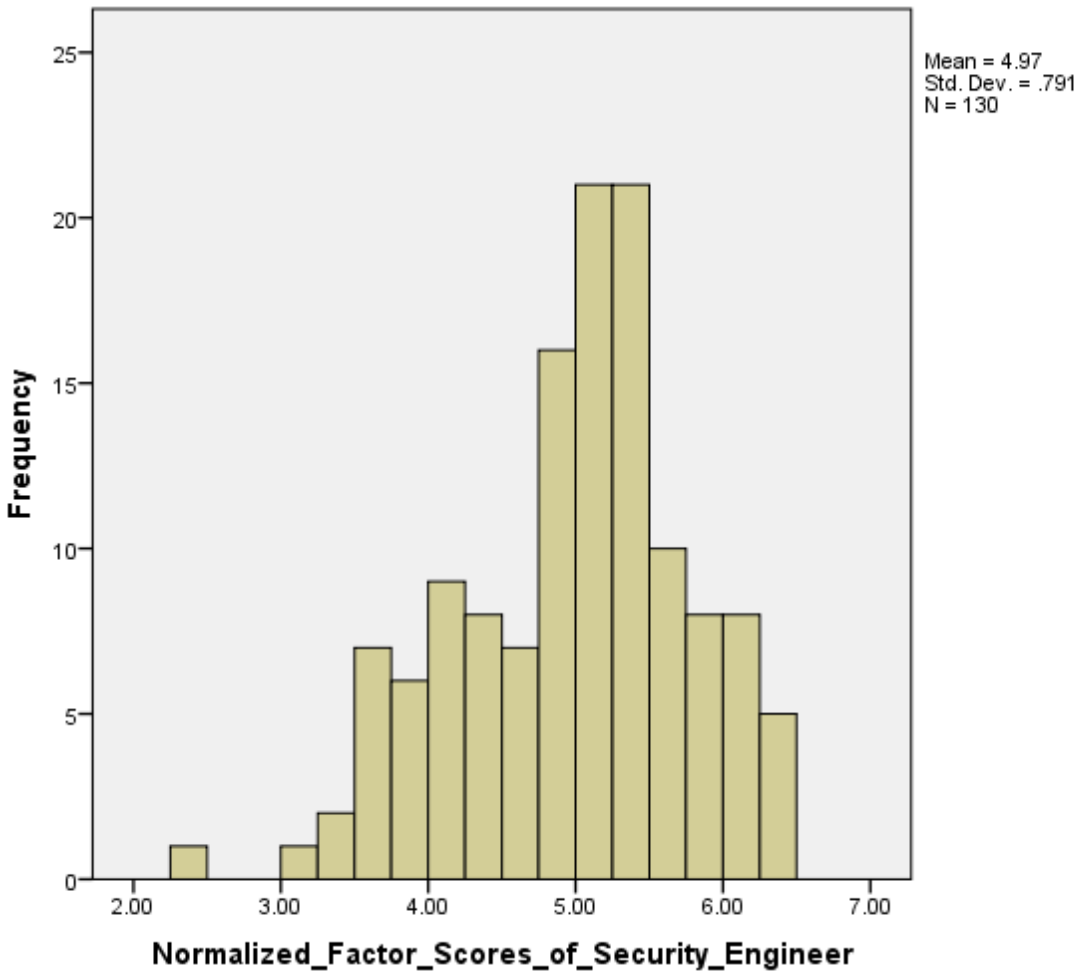


Figure 5-23 : Normalized Weighted Average Factor Scores of Security Engineer

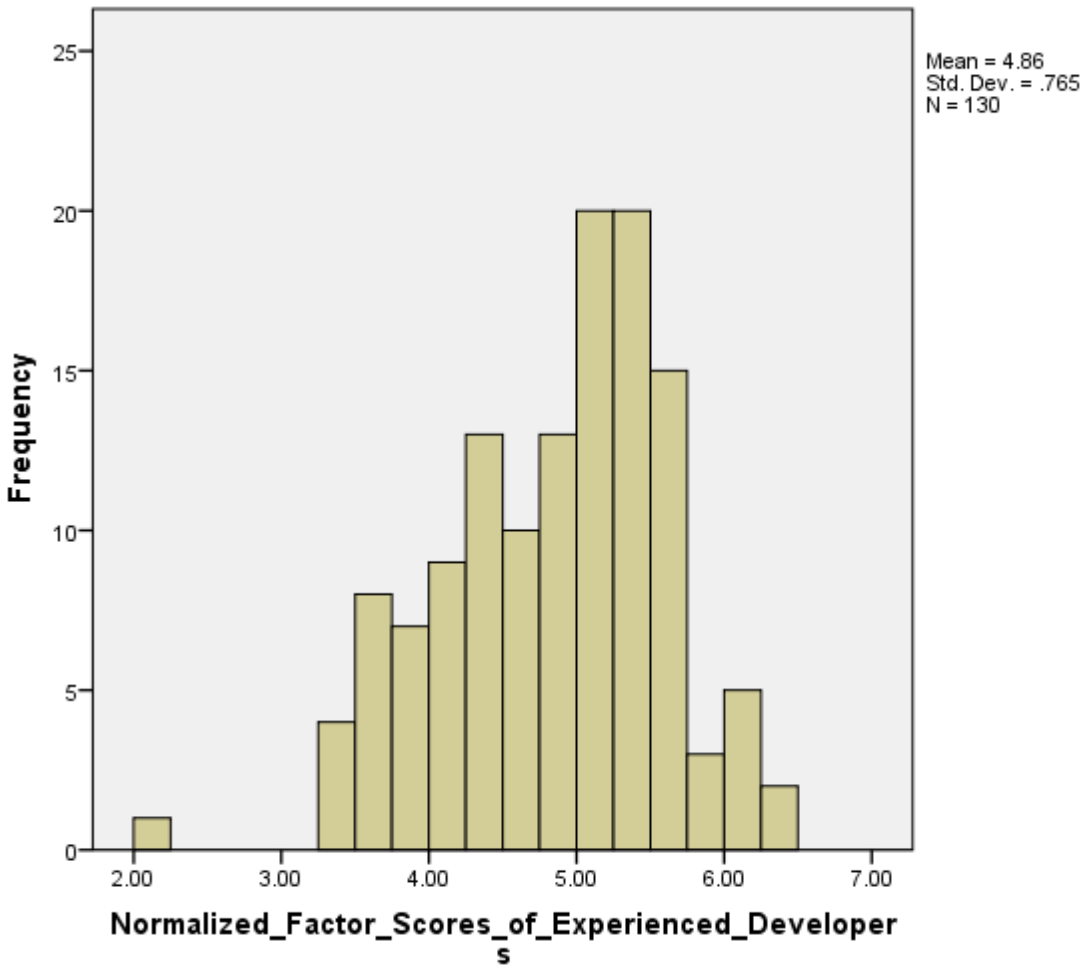


Figure 5-24: Normalized Weighted Average Factor Scores of Experienced Developers

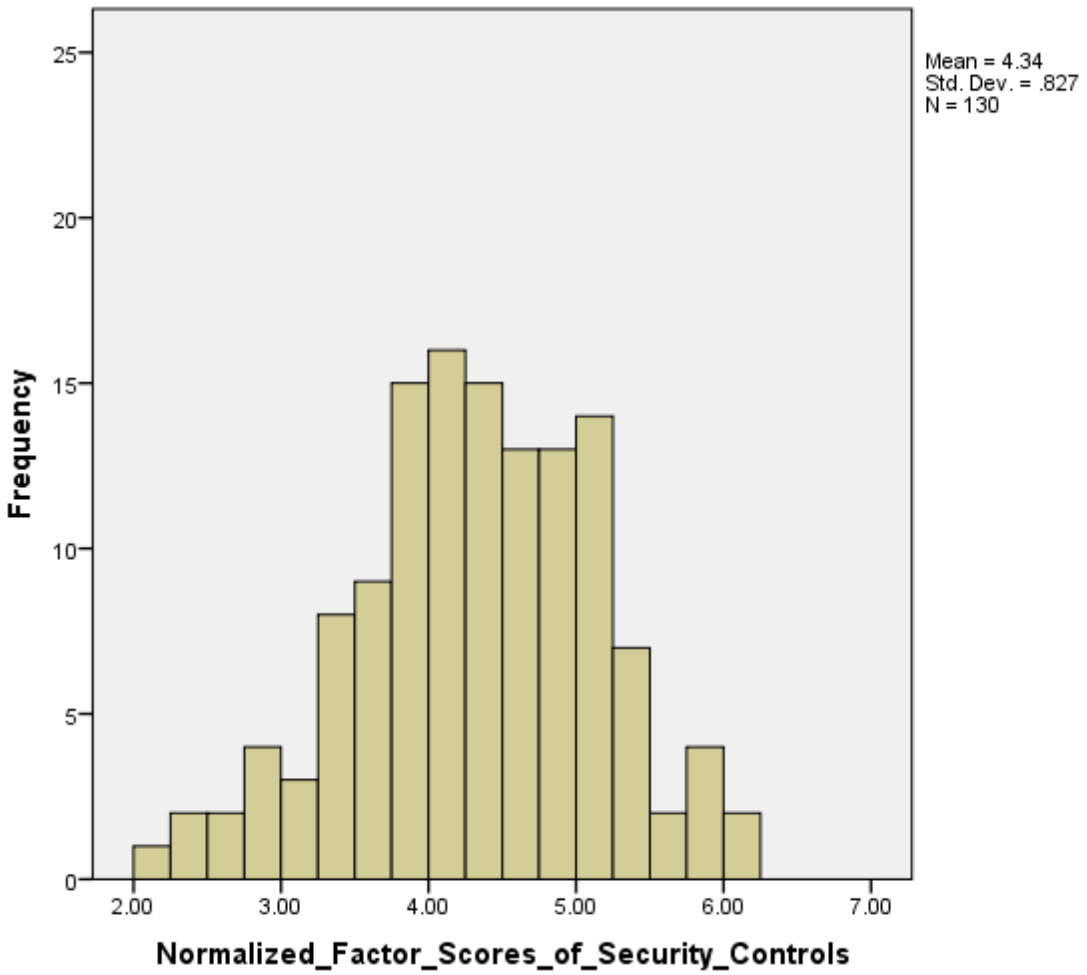


Figure 5-25: Normalized Weighted Average Factor Scores of Security Controls

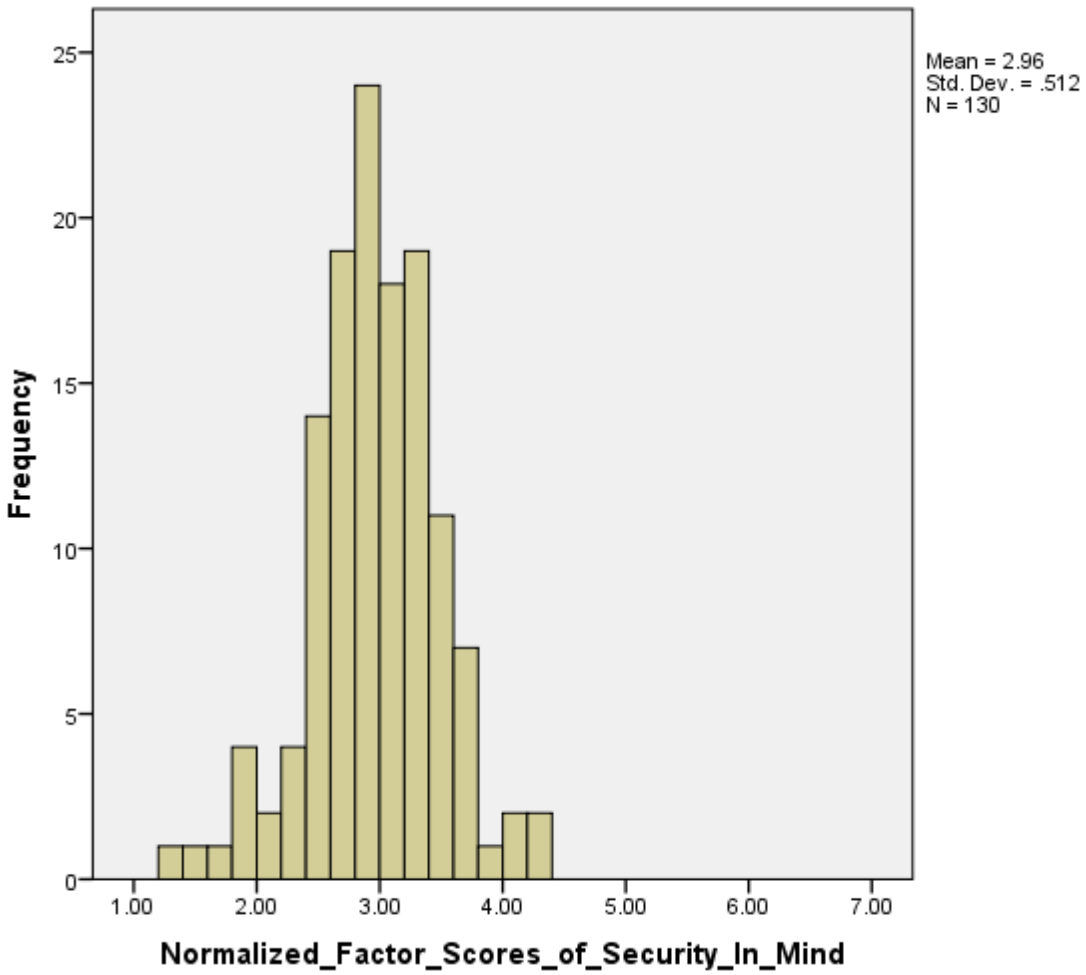


Figure 5-26 : Normalized Weighted Average Factor Scores of Security in Mind

In order to be able to seamlessly combine security and agility and evaluate the suitability of security mechanisms for use in Agile, the addition of the Security Engineer was the best and most suitable solution (from amongst the proposed solutions) as 78 participants (60% of all participants) had their highest factor scores in support of this solution. The second most suitable solution turned out to be the inclusion of experienced developers within the Agile team as evidenced by the highest factor scores of 48 participants (37% of all participants). Finally, the use of software security controls into Agile was supported by 3 participants to be the most suitable solution for them.

5.6.4.2 Change Agile Practices for Security

For the second research question, we will provide the results obtained from factor analysis on variables (or questionnaire items) related to whether changing Agile practices for sake of security is really needed or not. Table 5-27 shows how the codification is related to the items.

Scale	Code	Item number
Waterfall vs. Agile	WaterVs.Agile1-9	Item43-51
Awareness of Security	AwareOfSec1-14	Item52-65
Accelerated Schedule	AccSch1-10	Item66-75
Internal Agile Projects	InternalAgile1-4	Item76-80

Table 5-18 : Codification for Scale items (Change Agile Practices for Security)

Out of the 38 initial questionnaire items for section 2, 35 items (or statements) were included in the factor analysis for this section. A PCA extraction technique was applied to extract the underlying factors. To determine the number of factors to be extracted, we first conducted factor analysis using the Kaiser's criterion the results of which are presented in Appendix F. Additionally, we decided to suppress any factor loading with absolute value less than 0.30 from being displayed in the final results.

As explained earlier, we are following an a priori and proportion of variance criteria to determine the number of factors to be extracted. While Kaiser's criterion supports extracting 9 factors, our prior expectation when we designed this questionnaire was 4 factors. This agreed with the predetermined amount of variance which is 5% or more to ensure the importance of the retained factors for the analysis. Table 5-19 shows that 5^{th}, 6^{th}, 7^{th}, 8^{th}, and 9^{th} factor explain less than 5% of the total variance and therefore, we decided to specify the number of factors to be extracted from conducting factor analysis to be 4 factors. Appendix F contains the results of factor analysis when extracted number of factors is four and suppressing factor loadings less than 0.30. It's important to note that there is one dropped item after forcing the analysis to extract four factors and the total variance explained is 53.26%.

Total Variance Explained							
Component	Initial Eigenvalues			Extraction Sums of Squared Loadings			Rotation Sums of Squared Loadings
	Total	% of Variance	Cumulative %	Total	% of Variance	Cumulative %	Total*
1	7.322	20.921	20.921	7.322	20.921	20.921	5.727
2	6.179	17.655	38.576	6.179	17.655	38.576	3.902
3	3.174	9.068	47.644	3.174	9.068	47.644	3.740
4	1.966	5.618	53.262	1.966	5.618	53.262	3.839
5	1.721	4.917	58.179	1.721	4.917	58.179	3.697
6	1.443	4.124	62.303	1.443	4.124	62.303	4.557
7	1.313	3.751	66.054	1.313	3.751	66.054	3.135
8	1.156	3.303	69.357	1.156	3.303	69.357	2.546
9	1.067	3.049	72.406	1.067	3.049	72.406	1.632
Extraction Method: Principal Component Analysis.							
a. When components are correlated, sums of squared loadings cannot be added to obtain a total variance. Rotation Method: Oblimin with Kaiser Normalization.							

Table 5-19 : Total Variance Explained without specifying number of factors (RQ2)

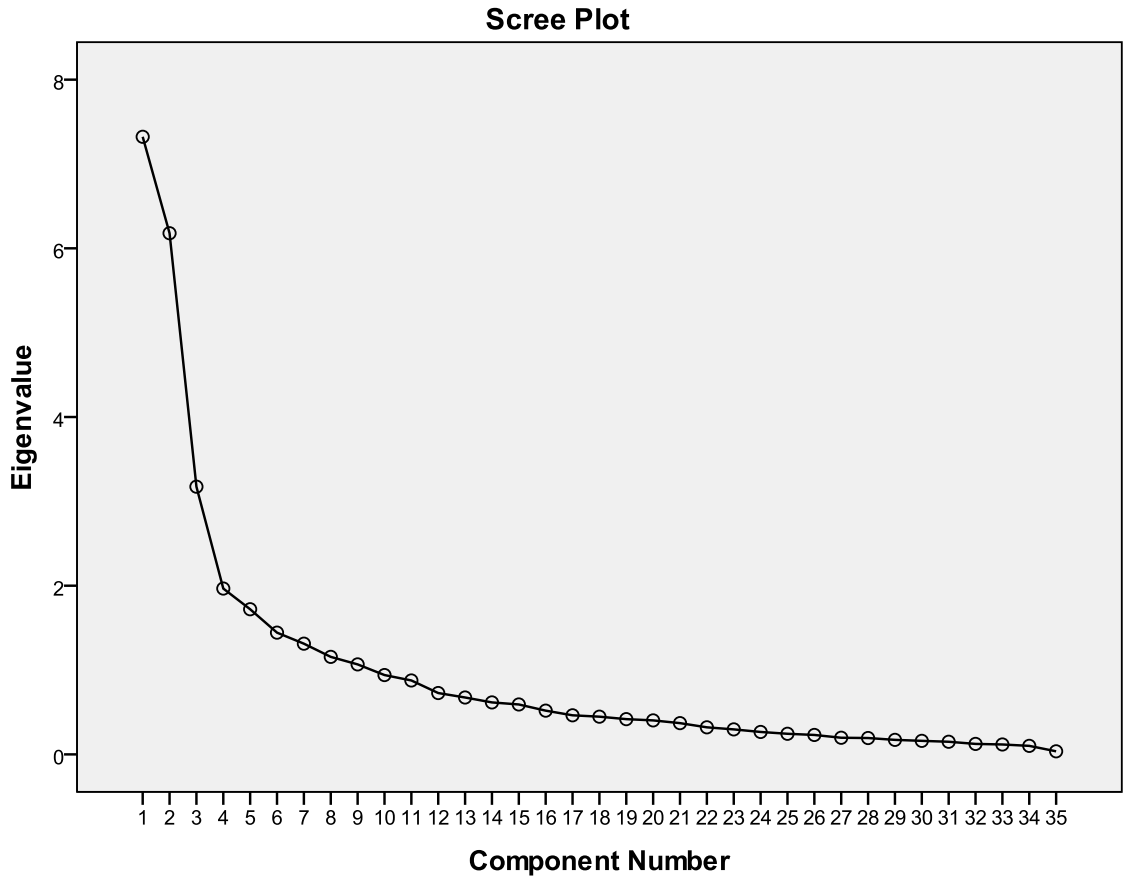


Figure 5-27: Scree Plot (Change Agile Practices for Security)

Total Variance Explained							
Component	Initial Eigenvalues			Extraction Sums of Squared Loadings			Rotation Sums of Squared Loadings
	Total	% of Variance	Cumulative %	Total	% of Variance	Cumulative %	Total*
1	7.322	20.921	20.921	7.322	20.921	20.921	6.170
2	6.179	17.655	38.576	6.179	17.655	38.576	4.726
3	3.174	9.068	47.644	3.174	9.068	47.644	4.127
4	1.966	5.618	53.262	1.966	5.618	53.262	5.643
Extraction Method: Principal Component Analysis.							
a. When components are correlated, sums of squared loadings cannot be added to obtain a total variance. Rotation Method: Oblimin with Kaiser Normalization.							

Table 5-20: Total Variance Explained with 4 Factors (RQ2)

Statement Code	Statement	Factor Loading*
Factor 1		
WaterVs.Agile2	Compared to Traditional Waterfall, Agile methodologies do less on security	-.775
WaterVs.Agile3	Compared to Traditional Waterfall, Agile methodologies do not deal with security directly	-.757
AccSch10	Accelerated Schedules of Agile Projects contributes to less secure software	.739
WaterVs.Agile1	Compared to Traditional Waterfall, Agile methodologies impact security more negatively overall	-.702
AccSch9	Accelerated Schedules of Agile Projects put major focus on achieving functionality and not much else	.698
AccSch7	Accelerated Schedules of Agile Projects affect security practices negatively	.646
AccSch8	Accelerated Schedules of Agile Projects increases the overall risk including security risk to the project	.633
InternalAgile2	Internal Agile Projects are less concerned with security	.606
InternalAgile3	Internal Agile Projects typically have less security requirements	.538
InternalAgile1	Internal Agile Projects can rely on the infrastructure for security	.440

*Extraction Method: Principal Component Analysis.
Rotation Method: Oblimin with Kaiser Normalization.

Table 5-21: Statements and their Loadings of Extracted Factor 1 (RQ2)

Statement Code	Statement	Factor Loading*
Factor 2		
AwareOfSec10	Awareness of security in Agile Projects requires organizational support	.673
AwareOfSec4	Awareness of security in Agile Projects allows team members to voice their security concerns openly	.623
AwareOfSec1	Awareness of security in Agile Projects allows for increased consideration of security throughout the project	.568
AwareOfSec8	Awareness of security in Agile Projects makes stakeholders more considerate of potential security issues	.567
AwareOfSec6	Awareness of security in Agile Projects adds more focus/emphasis on security within the project	.540
AwareOfSec9	Awareness of security in Agile Projects requires organizational maturity	.536
AwareOfSec7	Awareness of security in Agile Projects should include upfront planning for security specific issues	.526
AwareOfSec2	Awareness of security in Agile Projects requires elaboration steps to be included in the iteration	.520
AccSch5	Accelerated Schedules of Agile Projects are affected by security requirements	.419
AccSch6	Accelerated Schedules of Agile Projects affect every aspect of the project including security	.407

*Extraction Method: Principal Component Analysis.
Rotation Method: Oblimin with Kaiser Normalization.

Table 5-22: Statements and their Loadings of Extracted Factor 2 (RQ2)

Statement Code	Statement	Factor Loading*
Factor 3		
AwareOfSec13	Awareness of security in Agile Projects is important to some projects only	-.862
AwareOfSec11	Awareness of security in Agile Projects should be required in certain projects only	-.823
AwareOfSec14	Awareness of security in Agile Projects is only important if security is required for the project	-.792
InternalAgile4	Internal Agile Projects do not need security in some cases	-.701
InternalAgile5	Internal Agile Projects can have less security if they're not exposed publically	-.510
AwareOfSec3	Awareness of security in Agile Projects contributes more to security for the project than training can	-.419

* Extraction Method: Principal Component Analysis.
Rotation Method: Oblimin with Kaiser Normalization.

Table 5-23: Statements and their Loadings of Extracted Factor 3 (RQ2)

Statement Code	Statement	Factor Loading*
Factor 4		
WaterVs.Agile7	Compared to Traditional Waterfall provide team members more opportunities to mitigate security issues	-.803
WaterVs.Agile6	Compared to Traditional Waterfall are more secure because of testing	-.758
WaterVs.Agile4	Compared to Traditional Waterfall produce more secure software overall	-.729
WaterVs.Agile8	Compared to Traditional Waterfall are better on security because of their iterative nature	-.681
WaterVs.Agile9	Compared to Traditional Waterfall can discover security related issues faster	-.587
AccSch4	Accelerated Schedules of Agile Projects do not affect security substantially	.576
AwareOfSec5	Awareness of security in Agile Projects reduces security risks overall	-.505
AccSch1	Accelerated Schedules of Agile Projects does not hinder security	.446

*Extraction Method: Principal Component Analysis.
Rotation Method: Oblimin with Kaiser Normalization.

Table 5-24: Statements and their Loadings of Extracted Factor 4 (RQ2)

The following paragraphs attempts to interpret each factor and provide common theme that the factor is measuring:

Factor 1

By looking at the content of statements within this factor, a common theme is not clear and therefore, we cannot identify what this factor is measuring. This is because the responses given to the statements were from a mixture of various issues that were not directly on the same subject even though they were generally related to the Agile methods overall. Therefore we cannot clearly establish a link between this factor and the hypotheses related to the second research question (see Table 4-21).

Factor 2

From amongst the possible benefits and requirements of adding practices aimed at increasing awareness of security in Agile projects, the need for organizational support turned out to be most important indicator. Through the addition of the above mentioned practices into Agile projects, team members would have more opportunities to voice their security concerns openly and increasing their consideration over security issues throughout the project. Going over statements that load highly on this factor, it shows that they measure various aspects of security awareness in Agile and therefore, we will label it as 'awareness of security to Agile'.

Factor 3

The need and importance of adding practices to increase security awareness for certain Agile projects are the most influential indicators for this factor, followed by statements that address the reduced security for internal Agile projects. Not all projects require security and for some projects the security aspect is not a concern because either it is being handled at a different layer or the trusted and private nature of the deployment environment minimizes any security risk at the software level. Looking at the statements that load highly on this factor, it seems that they measure various security aspects on certain projects and therefore, it is interpreted as 'Security for Some Projects'.

Factor 4

Many statements related to the positive effect of using Agile methods on the security aspect of the produced software as opposed to Waterfall turned out to be the most influential indicators for this factor. Some of these positive effects of Agile methods are: the opportunities to mitigate security issues during the development process and the production of more secure software overall. As a result, this factor seems to be about measuring 'the positive effect of Agile on Security as opposed to Waterfall'.

Overall, the obtained factors for this section were not in agreement with what we expected when we designed this section of the questionnaire to measure certain issues related to changing Agile for sake of security and therefore, we decided to rely on the original raw scores of the questionnaire data for further analysis.

Aside from interpreting the obtained factors, now we will focus on correlation between factors. From the result of the oblique rotation, factor correlation matrix was obtained which shows that even though the correlation between factors are not large, it indicates that these factors cannot be assumed unrelated and therefore, the oblique rotation is the most suitable rotation technique for this section.

Component Correlation Matrix				
Component	1	2	3	4
1	1.000	.193	-.043	.150
2	.193	1.000	-.034	-.134
3	-.043	-.034	1.000	.218
4	.150	-.134	.218	1.000
Extraction Method: Principal Component Analysis. Rotation Method: Oblimin with Kaiser Normalization.				

Table 5-25: Factors/Component Correlation Matrix (RQ2)

5.7 Repeated-Measures ANOVA of Factor Scores

Previously, we calculated factor scores that represent the suitability of including four different proposed security solutions into Agile methods. These solutions (ordered through the factor analysis results) are:

Factor 1. Having a Security Engineer within the Agile team
Factor 2. The inclusion of experienced developers within the Agile team
Factor 3. Using software security controls in Agile projects
Factor 4. Building software with security in mind

The results of factor scores we mentioned earlier will be tested through Analysis Of Variance (ANOVA) and the insights gained through the identification of the most suitable solution(s) - that are found to be feasible and useful - will form the basis of the experimental trial that we intend to develop and conduct as part of our research on the topic of investigation of security issues in Agile methods.

Comparing the factor scores' mean of each solution allows us to be able to identify which solution(s) is most suitable to be included into Agile. Since the same participants have a score on each of the above factors (which represent our solutions), Repeated-Measures ANOVA was employed to test the differences between the mean of factor scores.

5.7.1 Assessing suitability of factor scores for Analysis

As an initial step for this analysis, the calculated factor scores need to be examined in terms of the existence of influential outliers (data screening) and the assumption of normality.

5.7.1.1 Data Screening of Factor Scores

As we have done on the original data of this questionnaire which was to check whether or not the existing outliers impacted the data, we calculated the difference (ΔMean) between the mean and the 5% trimmed mean for factor scores of each factor. All factor scores have a ΔMean that is relatively small ranging from 0 to 0.02. This suggests that outliers are not likely to influence the analysis (Pallant 2007). Therefore, we decided to include all the cases for Repeated-Measures ANOVA.

5.7.1.2 Assumption of Normality

To evaluate the normality of the data, skewness and kurtosis of each proposed solution was calculated. Table 5-35 shows that all values of skewness are between -0.51 and -0.18 and kurtosis are between -0.12 and 0.90 which are within the recommended ranges of ±2.0 (Garson 2012). In addition to assessing normality using kurtosis and skewness, we inspected the histogram of data distribution for the four solutions which showed that the data distribution was normal. As a result, we can conclude the normality of the new data set when it comes to both statistical and visual methods.

Factor	Number of Participants	Mean	5% Trimmed Mean	Δ Mean	Standard Deviation	Skewness	Kurtosis
Security Engineer	130	4.97	4.99	-0.02	0.79	-0.41	0
Experienced Developers	130	4.85	4.87	-0.02	0.76	-0.51	0.19
Security Controls	130	4.33	4.34	-0.01	0.82	-0.24	-0.12
Security in Mind	130	2.96	2.96	0	0.51	-0.18	0.90

Table 5-26: Analysis of Factor Scores

5.7.2 Analysis

5.7.2.1 Assumption of Sphericity

For the analysis, the factor scores cannot be assumed independent/unrelated and therefore conventional ANOVA will lack accuracy (since we used the same participants). This necessitated making a new assumption where we assume the relationship between pairs of proposed solutions is roughly similar and it's called the assumption of Sphericity (denoted by ε or Epsilon). According to Field (2009), "sphericity refers to the equality of variances of the differences between treatment levels [proposed solutions in our case]" (Field 2009). This means that the differences between pairs of factor scores in all combinations of proposed solutions need to have roughly equal variances.

To assess if the condition of sphericity has been met or not, Mauchly's test was employed. Mauchly's test examines the hypothesis whether the variances of differences between all possible treatments are equal. If Mauchly's test is significant ($p < 0.05$), then we can conclude that the differences between the variances of differences are significant and therefore the condition of sphericity doesn't hold. Otherwise, if Mauchly's test is not significant ($p > 0.05$), then it's sensible to say that there are no significant differences between variances of differences. Table 5-36 shows that the assumption of sphericity has been violated $\chi 2(5) = 20.46$, $p < .05$.

Mauchly's Test of Sphericity[b]

Within Subjects Effect	Mauchly's W	Approx. Chi-Square	df	Sig.	Epsilon[a]		
					Greenhouse-Geisser	Huynh-Feldt	Lower-bound
Issues	.852	20.460	5	.001	.897	.917	.333

Tests the null hypothesis that the error covariance matrix of the orthonormalized transformed dependent variables is proportional to an identity matrix.

a. May be used to adjust the degrees of freedom for the averaged tests of significance. Corrected tests are displayed in the Tests of Within-Subjects Effects table.
b. Design: Intercept Within Subjects Design: Issues

Table 5-27: Mauchly's Test of Sphericity

SPSS generates the corrections that are required to deal with violation of the assumption of sphericity. To be able to produce a valid F-ratio, we modify the degrees of freedom that used to assess the observed F-ratio. The most used corrections are: Greenhouse-Geisser and Huynh-Feldt (Field 2009).

Greenhouse-Geisser correction ranges between lower-bound estimate and 1 where lower-bound can be calculated by the result of 1 divided by the number of treatments (proposed solutions in our case) minus one. The lower-bound estimate for our case is 0.33 which should fall between 0.33 and 1. The closer the value is to 1, the more similar the variances of differences are which subsequently means the closer the data is to being spherical.

Table 5-36 shows that the Greenhouse-Geisser correction is 0.89. A value of 1 indicates sphericity and the observed value suggests the data is close to sphericity.

Tests of Within-Subjects Effects						
Source		Type III Sum of Squares	df	Mean Square	F	Sig.
Issues	Sphericity Assumed	331.647	3	110.549	398.787	.000
	Greenhouse-Geisser	331.647	2.690	123.310	398.787	.000
	Huynh-Feldt	331.647	2.752	120.500	398.787	.000
	Lower-bound	331.647	1.000	331.647	398.787	.000
Error (Issues)	Sphericity Assumed	107.282	387	.277		
	Greenhouse-Geisser	107.282	346.950	.309		
	Huynh-Feldt	107.282	355.042	.302		
	Lower-bound	107.282	129.000	.832		

Table 5-28: F-ratio and Associated Degrees of Freedom

Huynh and Feldt (1976) argued that Greenhouse-Geisser correction is too conservative and it wouldn't be able to reject too many false null hypotheses when its estimate is bigger than 0.75 (Huynh and Feldt 1976). This is also supported by Collier, baker et al. (1967) when the sphericity estimate was as big as 0.90 (Collier Jr, Baker et al. 1967).

Table 5-37 shows that in all cases whether we used too conservative correction such as Greenhouse-Geisser ($F(2.69, 346.95) = 398.78$, $p < .05$) or less conservative one such as Huynh-Feldt ($F(2.75, 355.04) = 398.78$, $p < .05$), F-value is statistically significant which indicates that the proposed solutions differed in terms of their suitability of inclusion into Agile methods. However, this doesn't show which solution(s) differed from the others.

5.7.3 Result and Interpretation

Estimates				
Issues	Mean	Std. Error	95% Confidence Interval	
			Lower Bound	Upper Bound
1) Security Engineer	4.973	.069	4.836	5.110
2) Experienced Developers	4.856	.067	4.723	4.989
3) Software Security Controls	4.337	.073	4.193	4.480
4) Security in Mind	2.962	.045	2.874	3.051

Table 5-29: Statistics of Factor Scores

Pairwise Comparisons						
(I) Issues	(J) Issues	Mean Difference (I-J)	Std. Error	Sig.[a]	95% Confidence Interval for Difference[a]	
					Lower Bound	Upper Bound
1	2	.117	.070	.565	-.069	.304
	3	.637*	.058	.000	.482	.792
	4	2.011*	.060	.000	1.850	2.171
2	1	-.117	.070	.565	-.304	.069
	3	.519*	.075	.000	.317	.721
	4	1.893*	.060	.000	1.734	2.053
3	1	-.637*	.058	.000	-.792	-.482
	2	-.519*	.075	.000	-.721	-.317
	4	1.374*	.068	.000	1.193	1.555
4	1	-2.011*	.060	.000	-2.171	-1.850
	2	-1.893*	.060	.000	-2.053	-1.734
	3	-1.374*	.068	.000	-1.555	-1.193
Based on estimated marginal means						
a. Adjustment for multiple comparisons: Bonferroni. *. The mean difference is significant at the .05 level.						

Table 5-30: Post Hoc Test of Repeated-Measures ANOVA

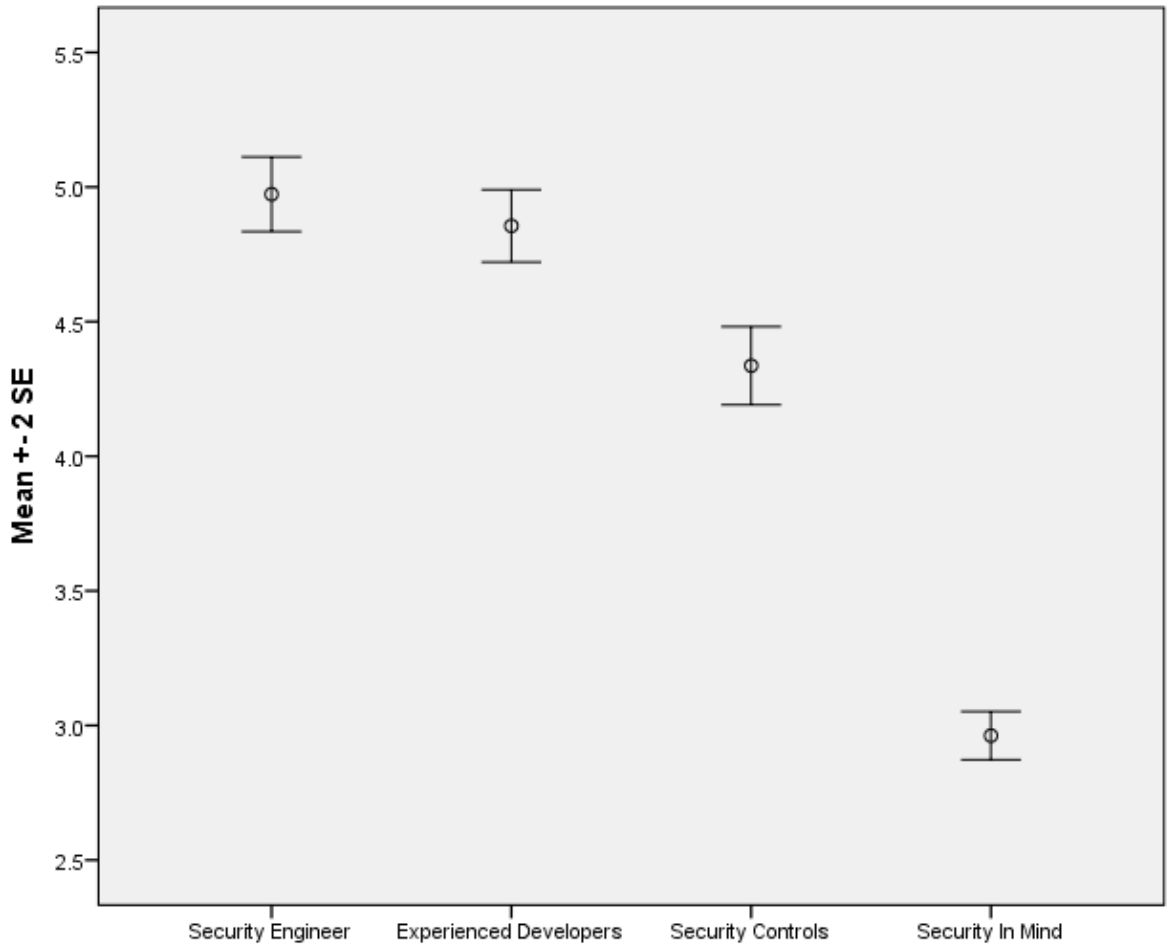


Figure 5-28: Factor Scores Mean and their Corresponding Standard Errors

Since Sphericity was violated, the Bonferroni option was selected for multiple comparisons as it is generally the most robust when it comes to univariate techniques (Field 2009). Table 5-39 represents the pairwaise comparisons of suitability of inclusion of different security solutions into Agile methods. Through the examination of the significance values and the differences of means of each pairwise comparison, we found that although suitability of including (which is represented by the factor scores) Security Engineer and Experienced Developers into Agile team are not significantly higher from each other (p = 0.565), they are significantly higher than other proposed solutions such as the use of Software Security Controls (p = 0.000 for both) and building software with security in mind (p = 0.000 for both). We also found that the suitability of using Software Security Controls in Agile project is significantly higher than building software with security in mind (p = 0.000).

In summary, we can conclude that the most suitable security mechanisms into Agile methods are: the inclusion of a dedicated Security Engineer or the use of more experienced developers.

5.8 Hypotheses Testing using One-Sample t-Test

Hypothesis testing enables the researcher to be able to draw inferences from the study sample regarding the population (Creswell 2009). In order to accomplish this, the one-sample t-test is performed in a situation in which the mean score of a sample is compared to a constant. Since we used a 7-point Likert Scale for our questionnaire, we will statistically test our hypotheses that were generated from the results of the qualitative phase (semi-structure interviews) of the research by comparing the mean score of each solution/issue with the fourth option which represents Neutral in our Likert Scale.

5.8.1 Combining Security & Agility

In this section, we will use the calculated factor scores to test the suitability of including four proposed solutions into Agile methods.

One-Sample Statistics				
	N	Mean	Std. Deviation	Std. Error Mean
Security Engineer	130	4.97	.79	.069
Experienced Developers	130	4.85	.76	.067
Software Security Controls	130	4.33	.82	.072
Security in Mind	130	2.96	.51	.044

Table 5-31: Statistics of Factor Scores for RQ1

One-Sample Test						
	Test Value = 4					
	t	df	Sig. (2-tailed)	Mean Difference	95% Confidence Interval of the Difference	
					Lower	Upper
Security Engineer	14.03	129	.000	.97	.83	1.11
Experienced Developers	12.75	129	.000	.85	.72	.98
Software Security Controls	4.63	129	.000	.33	.19	.48
Security in Mind	-23.12	129	.000	-1.03	-1.12	-.94

Table 5-32: One-Sample t-test Results for RQ1

The above tables indicate that there are significant differences between the mean of factor scores of each proposed solution from four (Neutral option).

Since our hypotheses in this section involved determining whether the mean score is greater than 4.0, one-tailed test is required rather than a normal two-tailed test. SPSS has no option to specify one-tailed for one-sample t-test so we had to check this manually by looking up the value in the critical t-distribution table. For critical t-value with 129 degrees of freedom and α of 0.05 (one-tailed) the result is 1.6568. In a one-tailed t-test if the t-value of a solution is higher than 1.6568 we can conclude that our mean score is significantly higher than 4.0 and subsequently our hypothesis is supported. Otherwise, if the t-value is lower than 1.6568, then we need to reject the hypothesis (Field 2009).

RQ	Hyp.	Hypothesis	One-Sample t-test	Hyp. Results
Combining Security & Agility	H1a	The addition of the dedicated security engineer is suitable for inclusion into Agile.	$t(129) = 14.03 > t_{critical}(129) = 1.65$	Supported
	H1d	The use of more experienced developers is suitable for inclusion in Agile.	$t(129) = 12.75 > t_{critical}(129) = 1.65$	Supported
	H1c	Use of security controls is suitable for inclusion into Agile.	$t(129) = 4.63 > t_{critical}(129) = 1.65$	Supported
	H1b	Writing Software with Security in mind is suitable for inclusion into Agile.	$t(129) = -23.12 < t_{critical}(129) = 1.65$	Not Supported

Table 5-33: RQ1 Hypotheses Testing Results

Suitability scores (which are represented by the factor scores) of including Security Engineer, Experienced Developers, and Software Security Controls into Agile are significantly higher than 4.0. This however is not the case for building software with security in mind where the t-value is less than 1.65 which means that suitability scores of this solution is significantly lower than 4.0.

5.8.2 Change Agile Practices for Security

As explained earlier, since the obtained factors for this section were not in agreement with what we expected when we designed this section of the questionnaire to measure certain issues related to changing Agile for sake of security, we decided to rely on the original raw scores of the questionnaire data for further analysis.

We've calculated the mean rating score of each participant for all questionnaire items related to a specific issue and tested it with the value of 4 by performing one-sample t-test. It's important to note that 5 items (items 43, 44, 45, 66, and 69) were reverse-phrased items and therefore we decided to subtract their mean from 8 in order to turn them into positive questions for the purposes of this analysis. In addition, item 47 was excluded from this analysis since the content of the questions is different for this item since it implied an answer of Strongly Agree was in fact indicating a neutral position by the participant (See Table 5-8).

One-Sample Statistics				
	N	Mean	Std. Deviation	Std. Error Mean
Waterfall vs. Agile	129	4.58	1.00	.08
Awareness of Security	124	4.91	.69	.06
Accelerated Schedule	122	4.24	.88	.07
Internal Agile Projects	120	4.20	1.09	.10

Table 5-34: Statistics of Rating Scores for RQ2

One-Sample Test						
	Test Value = 4					
	t	df	Sig. (2-tailed)	Mean Difference	95% Confidence Interval of the Difference	
					Lower	Upper
Waterfall vs. Agile	6.60	128	.000	.58	.40	.75
Awareness of Security	14.57	123	.000	.91	.78	1.03
Accelerated Schedule	3.03	121	.003	.24	.08	.40
Internal Agile Projects	2.04	119	.043	.20	.00	.40

Table 5-35: One-Sample t-test Results for RQ2

The result of two-tailed one-sample t-test rejected the null hypothesis of H2a which indicates that there is a significant difference between Agile methods and Traditional methods in terms of the security of the produced software. This significant difference favors Agile where it shows that Agile is better in terms of security as evidenced by the positive mean difference of 0.58. Moreover, As we have done on the earlier hypotheses, the results of one-tailed one-sample t-test showed that rating scores' mean is significantly greater than 4.0 for Awareness of security in Agile (H2b), Impact of accelerated schedules in Agile projects (H2c), and Reduced security in internal Agile projects (H2c).

RQ	Hyp.	Hypothesis	One-Sample t-test	Hyp. Results
Change Agile Practices for Security	H2a	There is no significant difference between Agile Methodologies and Traditional Methodologies with respect to the resulting security of the software produced.	$t(128) = 6.601 > t_{critical}(128) = 1.97$	Not Supported
	H2b	Adding practices aimed at increasing awareness of security is necessary for integration with Agile.	$t(123) = 14.57 > t_{critical}(123) = 1.65$	Supported
	H2c	Accelerated timeframes and schedules of Agile contribute to less secure software.	$t(121) = 3.03 > t_{critical}(121) = 1.65$	Supported
	H2d	Internal Agile projects can ignore the security aspects of the software.	$t(119) = 2.04 > t_{critical}(119) = 1.65$	Supported

Table 5-36: RQ2 Hypotheses Testing Results

5.9 Reliability

5.9.1 The Central Tendency Bias

We have opted to use a 7-point Likert scale in part to avoid the people's tendency to not choose extreme values in a questionnaire that measures attitudes and opinions. The 7-point Likert scale adds 1 more item in between the Strongly agree and Somewhat agree and between Strongly disagree and Somewhat disagree in order to soften the language and allow respondents to feel more comfortable choosing the "Agree" or "Disagree" statement if they feel they strongly agree but do not want to choose the strong option as a response.

5.10 Threats to validity

5.10.1 Content Validity

According to Field (2005), "Items on a questionnaire must relate to the construct being measured" (Field 2005). We can state with certainty that the items relate to the constructs being measured because we have based the questions on the result of our detailed and in-depth semi-structured interviews which clearly were on the subject and came from practitioners working in the field. Therefore we can say that we have achieved content validity for the questionnaire.

We have further conducted field testing in order to reduce the chances of intrusiveness of the content of the questionnaire items which allowed us to be able to achieve our desired number of respondents for the questionnaire.

5.10.2 Construct Validity

The point of Construct Validity is to write whether the scores or the results obtained "serves a useful purpose and have positive consequences when they are used in practice" (Hubley and Zumbo 1996). We can state with confidence that through finding statistically significant results from practitioners working in the field that these proposed solutions are the most promising in terms of their prospects for producing desirable results. However, since the discussions are purely based on the opinions of the experts and practitioners and not have real projects based upon them, there is a possibility that one or more proposed solution would not produce the intended benefits.

That is why we are attempting to conduct an academic study to experiment one or more of these proposed solutions in the real world.

5.10.3 External validity

Since we have chosen our participants from various roles and diverse geographical locations, we have tried to ensure that our results are applicable to as many software development firms as possible. These results would ideally be applicable to any team within any organization that is of small to medium size which is following one or more Agile principles and/or methodologies. Larger sample allows us to generalize the results found through the qualitative method accomplished earlier (interviews) and strengthens the conclusions of those issues that were discovered.

5.10.4 Logic leaps and false assumptions

We chose people that worked within Agile teams or had experience in security that had worked closely with such teams and therefore our results should be repeatable for any other sample with the same population characteristics as ours and therefore other subjects from the same population should produce the same or similar result. Since we are collecting our data in an entirely anonymous way, we have allowed for the highest degree of transparency from the respondents without the possibility of any backlash or retribution therefore we feel our data collection is as valid as one could reasonably expect and the number of participants and consensus that we have been able to attract further suggests that we have indeed been getting consistent and valid results.

5.11 Summary

This chapter includes the quantitative aspect in the form of questionnaire of the investigation into security issues in Agile which allowed us to generalize and test the hypotheses regarding various issues and topics raised in the qualitative study combined with the prior research issues that were similar to the one we discovered empirically on much bigger sample than what we were able to achieve in the qualitative phase.

Achieving a high degree of reliability for both sections and correct factor structure for the first section indicates a high degree of replicability of the obtained

results. It's also clear that since we were able to reach a bigger sample size (120-148 respondents out of the population of over 100,000 practitioners) with a broad geographical spectrum as well as a broad number of industries involved (look over the background section for more information) we can confidently say that the result of this questionnaire is more generalize-able to the population of Agile practitioners than the interviews conducted earlier. In order to get a true generalize-able result the sampling method needs to be fully random which was beyond the scope of the current study in terms of time and resource limitations. Further discussion on the sampling method is provided in section 5.2.3.

Chapter 6 Agile Security and Experienced Developers: An Experimental Trial

6.1 Introduction

This experimental trial is the final part of our mixed method approach which we conducted in order to gain a real-world perspective of the theories and results we obtained earlier through our qualitative and quantitative techniques. With better practical knowledge and understanding of security issues and their interaction with Agile methodologies, the experiment serves as the last perspective in the multiple perspectives on the important issues that were empirically identified and classified throughout our research. This chapter will outline the steps we went through during which a subset of an Agile development process was studied in depth to compare and contrast the effect of having more experienced developers on the resulting software. The experiment attempts to get more targeted data based on the top issues concerning the suitability of including more experienced developer(s) to the Agile team (as were found through our quantitative survey analysis) which uncovered the most important and suitable security integration mechanisms in Agile today. As a result of these experimental trials, we assess in terms of the resulting software, how effective more experienced developers are in addressing the need for an added security assurance argument to Agile methods.

The aim of the experiment would be to find out why the experienced developer would be suitable to be included into the Agile team. The reason why the security expert hypothesis was not chosen for the experiment is because of a rather high requirement in the number of years of expertise in software development as well as formal security related knowledge, training, and/or experience that is rarely if ever found in a student population. Even with practitioners this would be a challenge because there is not as yet a formally accepted role or commonly recognized certification such as: “software security expert” or “software security engineer” defined within the software development industry and more specifically Agile software development. It was therefore much more feasible and practical to see how relatively more experienced

developers with more knowledge and experience affect the security of the software produced and how they could have contributed to the increased overall awareness of security within the Agile team. The choice of more experienced developers is also more relevant to the industry because of the abundance of more experienced developers that are readily available and recognized formally that could be used to increase the quality and therefore security of the software without a considerable effort.

As part of this effort both objective and subjective data were gathered in order to have a more appropriate set of measurements available for analysis and review. The result of the top issues that were found to be feasible and useful (based on the results of the analysis and experimental work) will form the basis of the culmination of our research on the topic of investigation of security issues in Agile.

6.2 An Experimental Method Plan

"The Formal Experiments, Case Studies, and Surveys are three key components of empirical investigation in software engineering" (Fenton and Pfleeger 1997). Formal experiments are controlled investigations where during the accomplishment of a task or an activity a number of variables are singled out, used, and controlled to observe their effects. In our case experienced developer(s) are used in the Agile team (manipulated) to document their effect on the process as well as the final product which is the software produced. Since these experiments need a great amount of control, they are usually small in scope and involve a few people or events. In other words, experiments are "research in the small" (Fenton and Pfleeger 1997). The central factor in an experiment is to identify the level of control needed. If the level of control possible is high, then it is said to be good to do this kind of experiment. Otherwise, performing a case study is preferred. The degree to replicate (duplication factor) is also important to experiments which consist of the number of times the basic situation could be repeated. If the degree of replication is high then the experiment is preferred. However, if the degree is low, then perhaps a case study would yield better results. Usually, comparative studies are good fits

for formal experiments since the researcher(s) can test to see if the data they collect confirms or refutes the hypothesis. It should be clear how each type of data either confirms or refutes the hypothesis and therefore the hypothesis could be restated using more specific terminology to help with analysis. With this kind of experimentation the results of the experiment are said to be more generalizable (Fenton and Pfleeger 1997). For the experiment, the researcher attempted to document the relation between measures taken and the factors they are intended to reflect. This can be done through multiple indicators (metrics or measurements) for a factor rather than just one and allows for clear quantitative measures and enables the researcher to see what factors can affect the suitability of including experienced developer(s) into Agile. There are many studies that experiment using students throughout the literature (Williams, Kessler et al. 2000; Chao and Atli 2006; Sfetsos, Stamelos et al. 2006). In the work of Laurie Williams for the journal of Software (published by IEEE), she attempted to validate quantitatively how using pair programming during software development will help in producing higher-quality software in less time. This was done using a formal experiment with students in the same academic level with the same mix level of GPA. The first group formed the control group where each student worked individually and the second group formed the active group where students had been grouped into pairs in order to collaboratively accomplish the same task (Williams, Kessler et al. 2000).

6.2.1 Description

The experiment proposed here will involve a small Agile team to test the effect(s) of having more experienced developer(s) within the Agile team. This is contrasted with just having two ordinary developers which forms our control group as part of a comparative study. This is related to our RQ1 on the issues that were found to be one of the most important and popular solutions recommended by practitioners which showed the most probable compatibility with Agile methods. For the experimental work, according to Dean and Voss, the ultimate goal of any experiment is to assess Cause and Effect relationships (Dean and Voss 1999). Two main cause and effect relationships for our experiment is:

Goal(s)	Question(s)	Metrics
Investigate the suitability of adding experienced developer(s) into the Agile team	Does the existence of the experienced developer result in increased security awareness of the team?	Process Metrics
Investigate the effect of adding experienced developer on the produced software in terms of security	Does the existence of the experienced developer increase the security of the software?	Product Metrics

Table 6-1: GQM paradigm for experimental trial

Through the experimental trials and through the study of the team writing software we collected objective data on how and why the experienced developer(s) were effective in establishing a security assurance argument for Agile. Furthermore, we also got and account of how the members of the team feel about the changes that are introduced as a result of including more experienced member(s) to the Agile team.

6.2.2 The setting

The experiment was conducted in an academic setting involving students of the University of Southampton. During the beginning 3-4 week period the prospective students were notified of the existence of the experiment and their participation was solicited through email and for every interested candidate that was accepted, an inducement was provided at the end of the experiment. They were selected and met in groups of two in order to conduct the experiment and were provided with the tools and the instructions needed to accomplish their assigned tasks.

6.2.3 Design

According to Cockburn, a Process Miniature is a technique that introduces the experimental team(s) to a relatively new and perhaps unfamiliar process. The aim is to shrink a methodology into a short time period (weeks to hours) depending on the type of experiment (Cockburn 2004). The goal in our case is to

be able to go through an iteration of the software development process in the chosen time period. The process itself could be a subset of activities normally accomplished during a complete methodology (XP in our case) that is part of a full software development lifecycle. This has been used throughout academia and the industry to conduct experiments (Cockburn 2004). The students were chosen and assigned based upon their level of knowledge and experience in software development to one of two groups: control and active groups which followed a popular Agile practice of pair programming. In a typical XP scenario, the development team consists of 2 or more members that develop the software during a variable-length iteration. Here we briefly describe a typical XP process:

1. Planning Game

During the planning, the team is given a number of requirements that can be accomplished in the time-span of an iteration from the stories (requirements) which contains a list of prioritized items. These items will be turned into use cases that can be turned into code within an iteration.

2. Development, Testing, Review, Adjustment

These are the main activities of XP in which Development, Testing, Review, and Adjustment of the items are done by the team members. In eXtreme Programming (XP), a developer is paired with another developer to write code and then test it. These stages are followed in order to produce a complete (deployable) software unit.

3. Deployment

At the end of an iteration, the software with the new functionality will be deployed into the production environment. As many tests are conducted, many features are added to the software bringing it closer to a release.

Roles

- The Team (team of developers, testers, etc.)
    - A typical team includes members from different specializations to complete the goal of the project.

- On-site Customer
    - Responsible for providing the project requirement to the team members. He is also responsible for creating and prioritizing user stories.

To ensure everyone understands their role, the researcher made sure that the students knew how the process worked before they were qualified to conduct the experiment for each of the active and control groups. The vision of the experiment was focused on the specific activities and requirements in the story cards (which were simple), therefore there was no need to have an on-site customer for the experimental teams. The following figure illustrates the general process of XP:

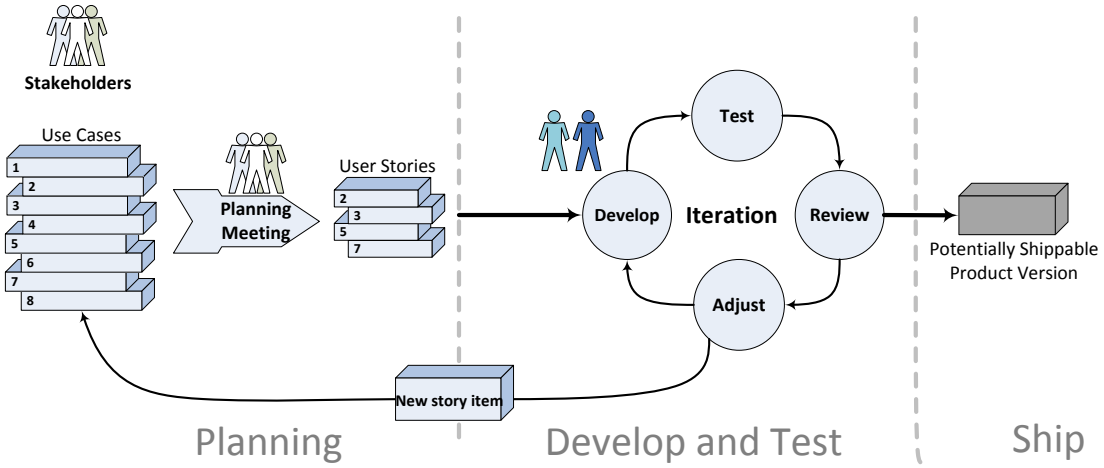


Figure 6-1 : Typical XP Process

6.2.3.1 Control Group

The control group would consist of 2 relatively inexperienced developers paired together and working as a unit. The group(s) would work as part of an Agile team following a subset of activities of XP methodology which were simplified to minimize variability of procedure. The activities that they can engage in this experiment include Integration Testing, Adding functionality, and Refactoring. They participated in a 2-3 hour iteration involving accomplishment of a set of specific task(s). They were asked mainly to develop a web based software. After

the completion of the iteration, they answered a questionnaire that collected information on their experience and interaction within the team as it relates to the process as well as the final product which is in creating an online submission mechanism for payment (HTML Form). As part of the simplified XP, the control and active groups were responsible for the following phases and activities:

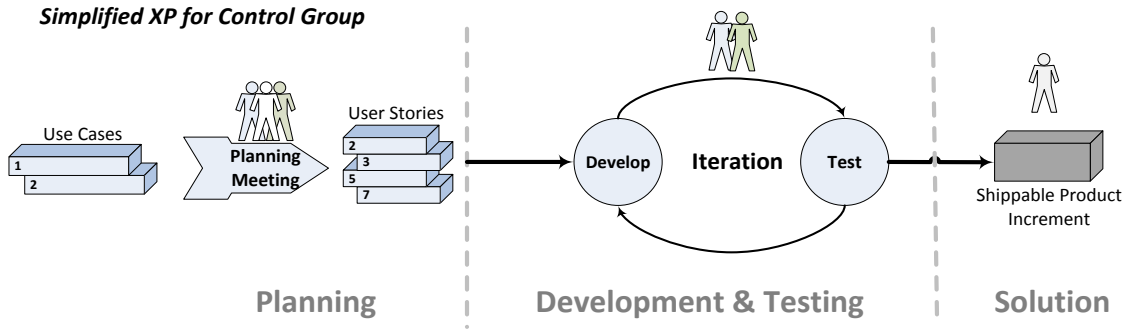


Figure 6-2 : Simplified Agile Framework for Control Group

Notable differences from normal XP include reduced number of stories, lack of on-site customer, and review and adjust activities. Multiple planning meetings were not needed because the team was working on specific functionality dictated by the researcher (acting as "offsite" customer). Everything else stayed the same as in a typical XP Agile group. Furthermore, there would not be any changes to the specifications (story cards) as a result of development and testing.

6.2.3.2 Active Group

The active group would consist of two relatively more experienced developers (compared to the control group) paired together. The groups worked as part of an Agile team following the activities of our simplified XP methodology. The activities that they can engage in this experiment include integration testing, adding functionality, and refactoring. They participate in 2-3 hour development

and testing iteration and would work in order to fulfill the requirements covering various aspects of the software development including quality and security of the final solution.

6.2.3.3 Experimental Hypotheses

- H1: Active Group is more aware of security than the base group doing the same programming tasks.
- H2: Active Group produces more secure software than the base group doing the same programming tasks.

6.2.3.4 Experimental Stages

The participants go through 3 stages:

1. Pre-Experiment Questionnaire
    - Some background questions and statements about participant experience in designing and developing software. This questionnaire will take approximately 5-10 minutes. Appendix H contains the pre-experiment questionnaire given to the students.
2. Experiment Instructions
    - As a student developer, participant(s) would be asked to participate in a simulated pair programming iteration (2-3 hours) along with another developer with equal or different knowledge and experience as a pair-programming team. Appendix H contains the experimental instructions given to the students.
    - The makeup of the teams is generally representative of some XP practitioners which typically work as pair programming units. The groups work on developing an online payment form that attempts to take credit card information from a fictitious customer through a payment form programmed in HTML and Javascript as well as server side technologies such as JSP. The tool used would be a pc capable of running an Integrated Development Environment which many Agile teams typically use to develop web-based applications. The practices that they can engage in with this tool include Integration Testing, Pair programming, and

interacting with various client side and server side web technologies.

3. Post-Experiment Questionnaire
   - After completion of the iteration, participants answer a questionnaire that collects metrics on their experience and interaction within the team as it relates to their contribution to the experiment and to the process as well as the final produced software. This questionnaire takes approximately 5-10 minutes. Appendix H contains the post-experiment questionnaire given to the students.

6.2.3.5 Software to be Written

The makeup of the teams is generally representative of some XP practitioners which typically work on software in 2-4 member teams as pair programming units. The groups work on developing an online payment form that attempts to take credit card information from a fictitious customer through a payment form programmed in HTML and Javascript as well as server side technologies such as JSP and Servlets. The tool that they use to make this software would be a workstation capable of running the NetBeans Integrated Development Environment which many Agile teams typically use to develop web-based applications. The practices that they can engage in with this tool include Integration Testing, Pair programming, and interacting with various client side and server side web technologies.

Prior to the start of the iteration, the student(s) were classified into two types: developer and experienced developer and then paired to constitute the control and active groups. They were not given any security or quality specific requirement as part of the iteration so any bias towards a particular requirement (namely quality and security) could be avoided. The members of the group were asked mainly to develop software and if they produced any tests or documentations during the iteration, we took that into account in the analysis.

After completion of the iteration, they completed a questionnaire that collected metrics on their experience and interaction within the team.

For the payment form itself, the groups were asked to gather the following information from the user who wishes to register:

- Credit Info
  - Name of card holder
  - Card Type (Visa, Master Card)
  - Card number
  - Expiration month
  - Expiration Year

6.2.3.6 The expected outcome

The expected outcome of the experiment would be to measure and observe differences between the Control and Active groups in how the choice of more experienced developer contributed to the process as well as to the final product that they were tasked to develop. While both the process (Agile) and the product (software) will be looked at and considered, the focus of the analysis would be on the effect of the experienced developer on the produced software. After all the data and observations, the final conclusion about the effectiveness of the Active group(s) compared with the Control group(s) would be the determining factor in understanding the impact of the experienced developer on the process of adding security to the XP flavor of Agile.

6.2.4 Influencing Attributes

In order to have a controlled experiment and minimize as many sources of variation as possible, the intended software to be written is chosen to be a subset of what a typical developer usually deals with on a daily basis. This is done to keep the experiment simple enough that the project can be done in the 2-3 hours that are set aside for it but is also challenging enough to demonstrate awareness of security and increased overall security of the produced software. Each team was responsible to write software based on the same specification and the Active teams included experienced developer(s) to help them with quality and/or security issues.

6.2.4.1 Experimental Unit

According to Fenton et al. (1997) "the experimental units are the objects to which the treatment is being applied" (Fenton and Pfleeger 1997). Our experimental unit for this experiment is the pair programming team that produces the solution (payment Form). As such, each developer that works on the software module that they develop could be considered for statistical purposes to determine the outcome of testing the hypothesis. In our case, multiple ways of data collection (Data gathering through surveys, unit testing, and static code reviews) are employed in order to get a large number of observations to use in statistical testing of the hypotheses under investigation. The following table depicts the metrics used and their types:

Metrics (Measurements or Response Variables)	Objective	Subjective	Type of Metric
Increased awareness of security within the Agile team (high degree of control)	Analysis by Researcher	Survey	Process
Increased security of the resulting software (low degree of control)	Analysis of Code	Survey	Product

-Overall degree of control over variables: High
-Control over design method: High
-Independent variable: Experienced Developer

Table 6-2: Type of Gathered Metrics and its related variables

According to Fenton (1997) process measures or metrics are "collections of software related activities" that are accomplished by the team members. Therefore, any metric related to the team and their behavior can be considered process metrics (Fenton and Pfleeger 1997). The manner and type of responses/reactions the developers give to a given security related question (on the survey) could indicate their level of awareness.

It is important to note that since the researcher was present at only the initial meeting during the iteration (where the subjects conducted their work), a clear set of steps on what needs to happen is provided and used to formalize the development process during the iteration. The manner and types of interactions

between team members was quantified (counted) and classified according to the above steps to gain an objective measure of the process of software development conducted by the team. Through the manner and frequency of these interactions, the relative differences between teams can be objectively identified and measured.

The project artifacts are another source of objective measures that can be taken from the project by the researcher. These metrics are easier to find and classify because the researcher has more time to analyze and look into any code, documentations, and meeting notes that can shed light on the manner and type of interactions that took place between team members. The most important and usable project artifact is the software that is produced.

These and other metrics and indicators form the basis for comparison of the control and active groups. According to Fenton (1997), both objective and subjective measures need to be taken in order for a comparison to be complete and to show a picture of a complete relationship (Fenton and Pfleeger 1997). As part of such comparison, the captured metrics would be looked into in terms of the number of issues identified and corrected, and other metrics that collectively form the basis for comparison of the groups.

6.2.4.2 Experimental Subjects

Those who apply the treatment designed by the researcher are called experimental subjects for the experiment which are the team members who do different actions depending on the type of group that they are in. As such, the characteristics and the identification mechanism for these individuals must be clearly specified (will be discussed in the next section). At the end of the iteration, the produced software (the payment form) was reviewed and the level of security it attained was decided according to the following scale.

6.3 Assessment of Security Issues in the Software

The first step in the assessment of the final produced software was to validate the correct functionality of the web application which is a standard Agile

practice done by practitioners to validate the correct functionality of the software. The test cases are acting as a kind of black box testing mechanism in order to check and make sure the software works as expected which is the most important aspect of the produced software. The assessment levels created by the researcher are partially based upon the mitigation of possible security threats that could affect the produced software. More specifically each level that the produced software attains helps in mitigation a possible security threat and/or vulnerability which indicates the relative security of the software produced. The list of threats are taken from the OWASP top 10 (OWASP 2010).

- Level 0: Failed unit/quality testing

- Level 1: No Input Validation (either on the client or the server side)
  Possible security threats: Invalid data submission, Injection flaws such as SQL injection, Broken Authentication, Cross Site Scripting, Cross Site Request Forgery, Insufficient Transport Layer Protection, Un-validated Redirects and Forwards

- Level 2a: Basic client side validation (empty field, etc.)
  Possible security threats: Injection flaws such as SQL injection, Broken Authentication, Cross Site Scripting, Cross Site Request Forgery, Insufficient Transport Layer Protection, Un-validated Redirects and Forwards

- Level 2b: Basic server side validation (non-null value, empty field, etc.)
  Possible security threats: Injection flaws such as SQL injection, Broken Authentication, Cross Site Scripting, Un-validated Redirects and Forwards , Insufficient Transport Layer Protection

- Level 3: Advanced client side validation (Regular expression, input formatting, other mathematical operations on inputted data etc.)
  Possible security threats: Cross Site Scripting, Broken Authentication, Cross Site Request Forgery, Un-validated Redirects and Forwards, Insufficient Transport Layer Protection

- Level 4a: Basic client side and basic server side validation
  Possible security threats: Broken Authentication, Un-validated Redirects and Forwards, Insufficient Transport Layer Protection

- Level 4b: Advanced client side and basic server side validation

Possible security threats: Broken Authentication, Un-validated Redirects and Forwards, Insufficient Transport Layer Protection

- Level 5a: No client side and Advanced server side validation (Regular expression, input formatting, mathematical operations on inputted data etc.)
  Possible security threats: Broken Authentication, Insufficient Transport Layer Protection, Un-validated Redirects and Forwards

- Level 5b: Basic client side and Advanced server side validation
  Possible security threats: Broken Authentication, Un-validated Redirects and Forwards, Insufficient Transport Layer Protection

- Level 5c: Advanced client side and advanced server side validation
  Possible security threats: Broken Authentication, Un-validated Redirects and Forwards, Insufficient Transport Layer Protection

- Level 6a: Basic client and basic server side validation plus encryption (HTTPS)
  Possible security threats: Broken Authentication

- Level 6b: Advanced client and advanced server side validation plus encryption (HTTPS)
  No major security threats!

Overall there are 12 distinguished levels of security that a team can attain throughout the course of developing the software. As is evident from the security levels, for each level that the team attains certain security threats and flaws are protected against and rendered neutral or at the very least those threats are minimized. The level that each Control group attains is compared with the level that the Active group attained in order to find out approximately what effect the team with the experienced developer (Active Group) had compared to the less experienced Agile team (Control Group).

6.4 Subjects

The experiment was designed to test the effects of having more experienced developer(s) within the Agile team. This is contrasted with just having two less

experienced developers which forms our control group as part of a comparative study.

6.4.1 Selection Criteria

In order for the experiment to be successful, the selection criteria for the participants, the assignment of them to various groups, and the number of participants per group has been identified. Undergraduate and postgraduate students of computer science are considered for the experiment because they are the most likely candidates to be practitioners working in the field and in fact some do work in the field as interns and in other similar capacities that makes them suitable candidates to conduct the experiment with. The selection procedure involves first emailing the students with an invitation asking them to participate and after they filled out the pre-experiment questionnaire (see Appendix H), qualified and willing candidates are contacted for the experiment.

Algorithm for the selection criteria:

1. Filter the candidates based on the following categorization, and group them based on their average responses (2.75 and above):
   - Category 1: (Java + JSP) and (HTML + Javascript)
   - Category 2: (PHP) and (HTML + Javascript)
   - Category 3: (C# + ASP .NET) and (HTML + Javascript)
   - Category 4: (Ruby + RubyOnRails) and (HTML + Javascript)
2. Then rank them based on their experience index (EI):
   - EI = (NY * DM) + PAS + YAE
     - NY: Number of Years
     - DM: Degree Multiplier
       - BA= 1.5
       - MA= 3
       - PHD=4
     - PAS: Previous Academic Score
     - YAE: Years of Additional Experience
3. Only people from categories 1 and 3 were considered because their knowledge most closely matched the requirements for our trial. Qualified students from other categories would have been considered

if categories 1 and 3 did not contain enough participants to conduct the experimental trail with.

4. The filtered list was then ranked by the Experience Index and the cutoff value of 13.125 (Median) was chosen to distinguish between the Control and Active groups
    - All potential candidates with EI of less than 13.125 were assigned randomly to control groups
    - All potential Candidates with EI of 13.125 or above were randomly assigned to active groups

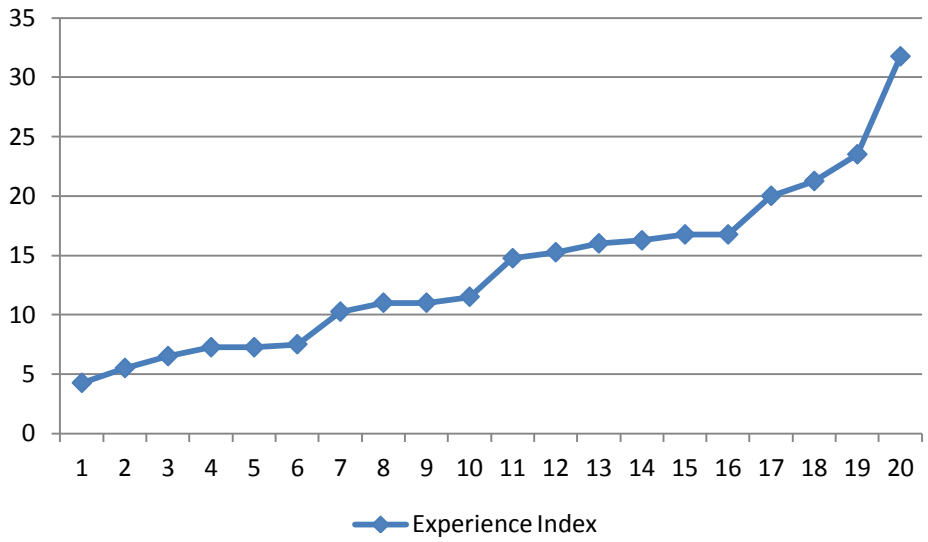


Figure 6-3: Experience Index distribution for Qualified Participants

6.4.2 Recruitment Strategy

The experiment was conducted in an academic setting involving students of the University of Southampton studying Computer Science. There was a 2-4 week period where the prospective students were notified of the existence of the experiment and their participation solicited through email in two rounds. It was necessary to offer an inducement particularly to attract experienced student developers which turned out to be a determining factor for most participants (based on their statements).

6.4.3 Assignment of Subjects to Groups

In order to make this simulate a typical Agile team, we assigned each group with 2 members and each of them acted as a stakeholder which is the same as in a typical Agile team. We assigned to each group two developers working as a pair programming team. For the control group, two relatively less experienced developers were chosen to accomplish the tasks. For the Active group, two of relatively more experienced developers were put in an Agile team as a two member pair-programming group to assess their effect on the process as well as the final product which is the software produced. The participants were taken through an iteration of the software development process in the chosen time period which was 2-3 hours. There was be five control groups and five active groups which resulted in a total of 20 students (minimum 16 needed) for participation in this experimental trial.

For this experiment, considering the level of statistical significance (or alpha) to be 0.05, the minimum amount of power needed is 0.25, and the effect size, the expected differences in the means between the control and active groups expressed in standard deviation units, is 1.20. Therefore, by looking-up in the power table the sample size needed for each group is 4 teams (Cohen 1988). This means 8 participants for each group was needed at minimum, and the minimum total number of participants required for this experiment is 16. However, we were able to conduct the experimental trial with 5 groups per condition which increased our probability to detect the effect to 0.5 (with effect size of 1.4) which means there is a now a 50% chance rather than a 25% chance for us to detect a tangible difference between groups.

To be able to carry out our experimental plan we contacted the Electronic and Computer Science School Ethics Committee at the University of Southampton and received the necessary approval (submission #4469).

6.5 Experimental Results

Out of 53 potential submissions, we chose 20 qualified candidates for our experimental trials whose results are presented for various stages of the experimental trial.

6.5.1 Pre-Experiment Questionnaire

Academic Level

Question	Bachelor	Master	PhD
1st Year	2	7	1
2nd Year	5	0	1
3rd Year	2	0	1
4th Year	1	0	0
Total:	10	7	3

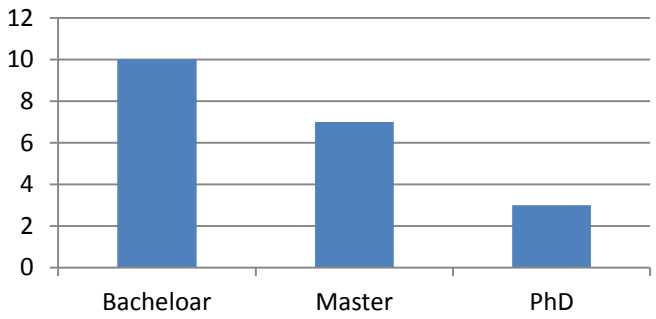


Figure 6-4 : Academic Level of Participants

Question	Control			Active		
	B.S.	M.S.	PhD	B.S.	M.S.	PhD
1st Year	2	3	0	0	4	1
2nd Year	4	0	0	1	0	1
3rd Year	1	0	0	1	0	1
4th Year	0	0	0	1	0	0
Total:	7	3	0	3	4	3

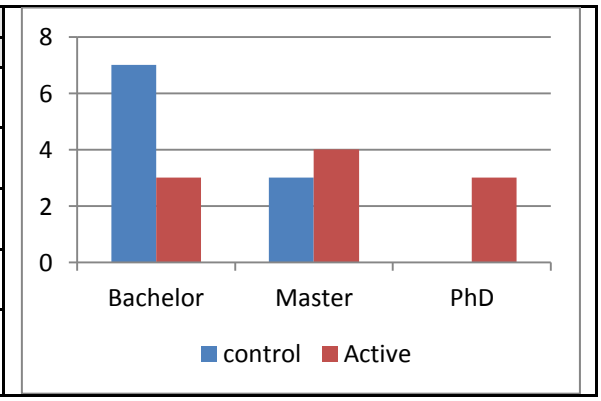


Figure 6-5: Academic Level of Participants (Control vs. Active)

As previously shown, the academic level of the participants for both Active and Control groups showed that the Active group participants had consistently more experience than the Control group participants. This means that the classification mechanism achieved the desired effect of having to include more experienced developers on the Active groups.

Concepts Learned

No	Statement		Expert	Proficient	Comfortable	Know little	No Knowledge
Pre1	**How proficient are you in Object Oriented Programming Languages:**						
Pre1a	Java	Freq.	3 (15%)	9 (45%)	7 (35%)	1 (5%)	0
Pre1b	C#	Freq.	3 (15%)	1 (5%)	4 (20%)	5 (25%)	7 (35%)
Pre1c	Ruby	Freq.	0	0	2 (10%)	2 (10%)	16 (80%)
Pre1d	PHP	Freq.	1 (5%)	2 (10%)	6 (30%)	8 (40%)	3 (15%)
Pre2	**How proficient are you in Web Design:**						
Pre2a	HTML	Freq.	4 (20%)	8 (40%)	8 (40%)	0	0
Pre2b	JavaScript	Freq.	1 (5%)	7 (35%)	9 (45%)	3 (15%)	0
Pre3	**How proficient are you in Web Development:**						
Pre3a	Servlet/JSP	Freq.	0	5 (25%)	4 (20%)	4 (20%)	7 (35%)
Pre3b	ASP/ASP.Net	Freq.	2 (10%)	2 (10%)	3 (15%)	5 (25%)	8 (40%)
Pre3c	PHP	Freq.	1 (5%)	3 (15%)	5 (25%)	9 (45%)	2 (10%)
Pre3d	RubyOnRails	Freq.	0	1 (5%)	1 (5%)	2 (10%)	16 (80%)
Pre4	**How proficient are you in IDEs:**						
Pre4a	Eclipse/NetBeans	Freq.	6 (30%)	5 (25%)	8 (40%)	1 (5%)	0
Pre4b	Visual Studio	Freq.	4 (20%)	5 (25%)	3 (15%)	5 (25%)	3 (15%)
Pre5	**How proficient are you in Databases:**						
Pre5a	MySQL	Freq.	2 (10%)	11 (55%)	5 (25%)	1 (5%)	1 (5%)
Pre5b	MS Access	Freq.	2 (10%)	2 (10%)	3 (15%)	9 (45%)	4 (20%)
Pre6	**How proficient are you in Software Development Process:**						
Pre6a	Waterfall	Freq.	1 (5%)	8 (40%)	6 (30%)	4 (20%)	1 (5%)
Pre6b	XP	Freq.	0	5 (25%)	0	7 (35%)	8 (40%)
Pre6c	Scrum	Freq.	0	8 (40%)	5 (25%)	4 (20%)	3 (15%)
Pre6d	TDD	Freq.	1 (5%)	1 (5%)	1 (5%)	4 (20%)	13 (65%)
Pre7	**How proficient are you in Security Related Topics:**						
Pre7a	Cryptography	Freq.	0	3 (15%)	5 (25%)	10 (50%)	2 (10%)
Pre7b	Network Security	Freq.	0	3 (15%)	6 (30%)	8 (40%)	3 (15%)

Table 6-3: Self-Reported Knowledge Level of the Participants

The participants on average had both server side and client side web based programming knowledge and experience which made them particularly suitable for the experimental trial. Most notably, the Proficiency and Expertise in Java and JSP were preferred which meant that whoever scored the most on those questions could be our suitable candidate to act as our experienced developer on the active groups.

Additional Experience

No	Statement		More than a year	6 months to year	3 - 6 months	Up to 3 months	No experience
Pre8	**Additional Experience in these categories:**						
Pre8a	Web Administration	Freq.	6 (30%)	0	1 (5%)	2 (10%)	11 (55%)
Pre8b	Web Application Development	Freq.	8 (40%)	1 (5%)	2 (10%)	5 (25%)	4 (20%)
Pre8c	Web Design	Freq.	6 (30%)	3 (15%)	3 (15%)	3 (15%)	5 (25%)
Pre8d	Network Administration	Freq.	3 (15%)	1 (5%)	0	2 (10%)	14 (70%)
Pre8e	Database Administration	Freq.	4 (20%)	3 (15%)	2 (10%)	5 (25%)	6 (30%)
Pre8f	Testing & Quality Assurance	Freq.	4 (20%)	2 (10%)	1 (5%)	5 (25%)	8 (40%)
Pre8g	Programming	Freq.	16(80%)	2 (10%)	2 (10%)	0	0
Pre8h	Software Engineer/Architecture	Freq.	9 (45%)	2 (10%)	5 (25%)	2 (10%)	2 (10%)

Table 6-4: Self-Reported Experience Level of the Participants

The additional experience of participants shows that the participants had more knowledge and experience beyond the academic setting which made them more suitable to be used for the experiment in the absence of the real practitioners working in the field because they are considered as candidates to get a real job in the industry (even so they may not have done so).

6.5.2 The Produced Software

At the start of the experiment, a Windows PC running NetBeans Integrated Development Environment was set up in a dedicated room for each group. When both developers were present in the location, a copy of the experimental instructions (see

Appendix H) was distributed to each developer. The researcher demonstrated how to execute and run the code for each team asking them to finish within the allotted 3 hours.

After the produced software was collected from the teams, the first step in the assessment of the final produced software was to validate the correct functionality of the web application. The test cases are a black box testing that checks to make sure the software worked as expected.

6.5.2.1 Testing Methodology

Field	Case 1	Case 2	Case 3	Case 4
Name	John Smith	john smith	JOHN SMITH	j smith
Card Type	Visa	Master Card	Master Card	Visa
Card Number	4012888888881881	5105105105105100	5105105105105100	4012888888881881
Expiration	05/2013	04/2014	11/2015	02/2022
Expected Result	Pass	Pass	Pass	Pass

Table 6-5: Data for Each Test Case

Group Number	Test Case 1	Test Case 2	Test Case 3	Test Case 4
1	✗	✗	✗	✗
2	✓	✗	✗	✗
3	✗	✗	✗	✗
4	✓	✓	✓	✓
5	✓	✓	✓	✓
6	✓	✓	✓	✓
7	✓	✓	✓	✓
8	✓	✓	✓	✓
9	✓	✓	✓	✓
10	✓	✓	✓	✓

Table 6-6: Outcome of Each Test Case Against Each Group's Produced Software

In order to ensure the software produced was functioning properly, we created and applied unit test cases (standard Agile practice) prior to static code reviews. If the produced software passed these tests, then that meant that the security assessment could proceed beyond the level 0 which was the failed state category that two Control and one Active group fell into. This failure rate shows that the control groups had more failure rate than the Active groups which means that having more experienced developers may have played a role in producing complete and functioning software.

6.5.2.2 Assessment Procedure

Each group's produced software would be analyzed in a white box manner (static code review) in order to assign a level of security to the final product. The assessment levels (described earlier in this chapter) represent the state of the resulting application as it relates to its employed security mechanism(s).

Assessment Level	Scale	Resulting score
0	0	0
1	0.5	0.5
2a	1.0	2.0
2b	1.5	3.0
3	2.0	6.0
4a	3.0	12.0
4b	3.5	14.0
5a	4.0	20.0
5b	4.5	22.5
5c	4.8	24.0
6a	5.0	30.0
6b	5.5	33.0

Table 6-7: Assessment Scale for Final Scoring

The scale goes up after each assessment level and for each minor stepping we add a nominal value to the scale (ex: a = 0.0, b = 0.5). Therefore, Assessment level 4b includes the base scale of 3.0 plus the 0.5 for the b stepping which results in 12.0 + 2.0 = 14.0 resulting score. We did this codifying scheme to be able to express in numbers the fact that the level of resulting security score jumps by a nominal factor each time the assessment level is increased. It is noteworthy to mention that each minor stepping only adds a few points to the score but a major level changes the resulting score with a much bigger increase to reflect the bigger change. We need this codifying scheme to allow us to explain the differences between each level of security as a relative jump rather than a simple linear increase in order to reflect reality much more closely.

6.5.2.3 Assessment Results

6.5.2.3.1 Client Side

Group Number	Client Side			
	Basic		Advanced	
	Empty field	Input Formatting	Regular Expression	Other math Opt
2			⊗	
3	✓	✓	⊗	
4	✓	✓		
9	✓	✓		
10	✓	✓		

Table 6-8: Client Side Security Assessment for the Control Groups

Group Number	Client Side			
	Basic		Advanced	
	Empty field	Input Formatting	Regular Expression	Other math Opt
1	✓		✗	
5	✓	✓	✓	✓
6	✓	✓		✓
7				
8	✓	✓	✓	✓

Table 6-9: Client Side Security Assessment for the Active Groups

6.5.2.3.2 Server Side

Group Number	Server Side		
	Basic	Advanced	
	Empty field Non-Null value	Regular Expression	Other math Opt
2		✗	✗
3	✓		✓
4	✓	✓	✓
9	✓		
10	✓		✓

Table 6-10: Server Side Security Assessment for the Control Groups

Group Number	Server Side		
	Basic	Advanced	
	Empty field Non-Null value	Regular Expression	Other math Opt
1	✓	✓	
5	✓	✓	
6	✓		✓
7	✓	✓	✓
8	✓		✓

Table 6-11: Server Side Security Assessment for the Active Groups

From amongst the groups that passed the initial testing phase, the client side assessment results show that the Active groups used more advanced techniques compared to the control groups which predominantly used basic techniques such as empty field checking. On the server side, the same situation also occurred which means the Active groups were consistently using more advanced server side techniques to increase the security of their software.

6.5.2.3.3 Final classification

Group Number	Group Type	Completion Time	Assessment Level
1	Active	2:45	0
2	Control	2:38	0
3	Control	2:02	0
4	Control	2:35	5b (Basic client, Adv. server)
5	Active	2:17	5c (Adv. client, Adv. server)
6	Active	2:09	5c (Adv. client, Adv. server)
7	Active	2:52	5a (None client, Adv. server)
8	Active	3:00	5c (Adv. client, Adv. server)
9	Control	2:41	4a (Basic client, Basic server)
10	Control	2:48	5b (Basic client, Adv. server)

Table 6-12: Overall Security Assessment

6.5.2.4 Group Scores

6.5.2.4.1 Control Group

Group Number	Assessment Level	Final Score
2	0	0
3	0	0
4	5b (basic client, Adv. server)	22.5
9	4a (Basic client, Basic server)	12
10	5b (Basic client, Adv. server)	22.5
	Total Group score:	**57**
	Mean Group score for Control group	**11.4**

Table 6-13: Overall Security Assessment for Control Group with Final Scores

6.5.2.4.2 Active Group

Group Number	Assessment Level	Final Score
1	0	0
5	5c (Adv. client, Adv. server)	24.0
6	5c (Adv. client, Adv. server)	24.0
7	5a (none client, Adv. server)	20.0
8	5c (Adv. client, Adv. server)	24.0
	Total Group score:	**92.0**
	Mean Group score for Active group	**18.4**

Table 6-14: Overall Security Assessment for Active Group with Final Scores

The results obtained through both the client side and server side aspects of the produced software are in agreement with our stated hypotheses which show that the Active groups (where the relative experience of the developers are higher) have a higher overall level of security than the control groups which speaks also to their higher level of awareness as well. This is evident from the final obtained group score for the Active and Control groups which scored 57 and 92 respectively. The aggregate score for the Active groups shows an almost two fold increase in security compared to the Control groups when it comes to the security of the software produced. Even though looking at the full data implies the final scores are within the measurement error, the overlap is rather small and by excluding Team1 score (an outlier) from the results, we get a separation beyond the measurement error. The graph shown in Figure 6-6 suggests the increase is indeed more visible and out of the measurement error.

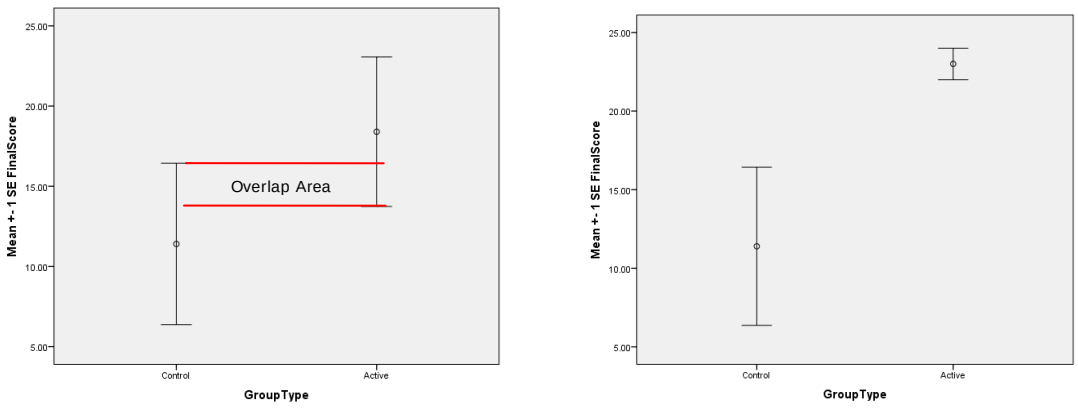


Figure 6-6: Assessment score means before and after outlier removal

The above result appears to support our hypotheses related to the increased level of security of the produced software when more experienced developers are present as part of the agile team but further analysis is needed to assess the significance of this difference.

6.5.3 Post-Experiment Questionnaire

A number of tables showing the actual results of the questionnaire for the actual participants (Control vs. Active) are as follows:

1a. Switch roles				
Categories	**%**		**Count**	
	C	**A**	**C**	**A**
Never	0	20	0	2
Few times (1-7 times)	80	50	8	5
Frequently (8-15 times)	10	30	1	3
All the time (16 or more)	10	0	1	0

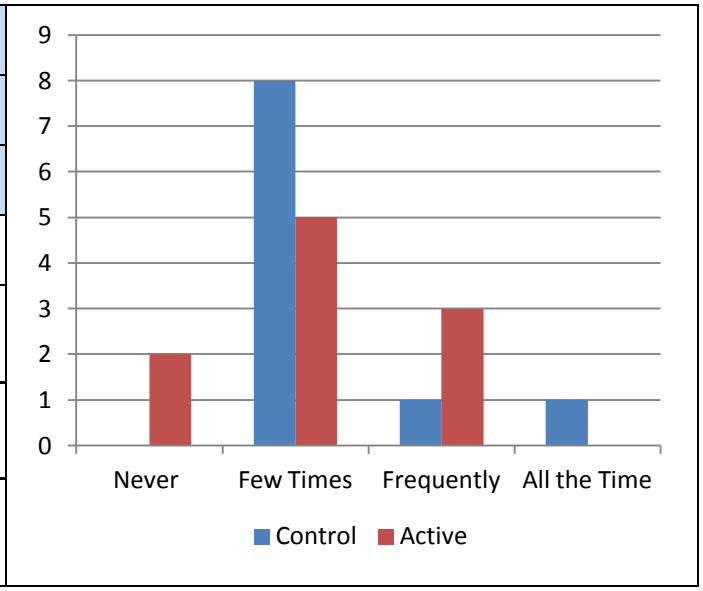


Figure 6-7: Switch roles between the team members in each group (control vs. active)

1b. Discovered bugs				
Categories	**%**		**Count**	
	C	**A**	**C**	**A**
Few times (1-4 times)	20	30	2	3
Frequently (5-8 times)	30	40	3	4
All the time (9 or more)	50	30	5	3

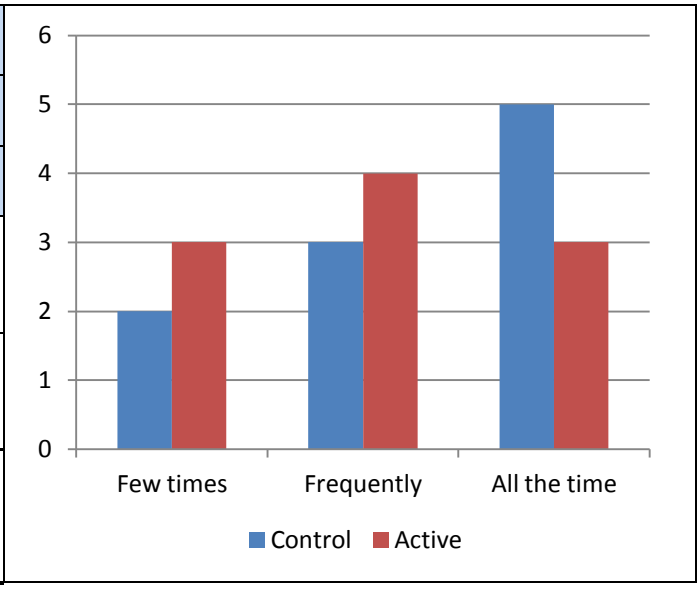


Figure 6-8: The number of Discovered Bugs in each group (control vs. active)

1c. Fixed bugs				
Categories	**%**		**Count**	
	C	**A**	**C**	**A**
Never (0 times)	0	10	0	1
Few times (1-3 times)	30	20	3	2
Frequently (4-7 times)	20	40	2	4
All the time (8 or more)	50	30	5	3

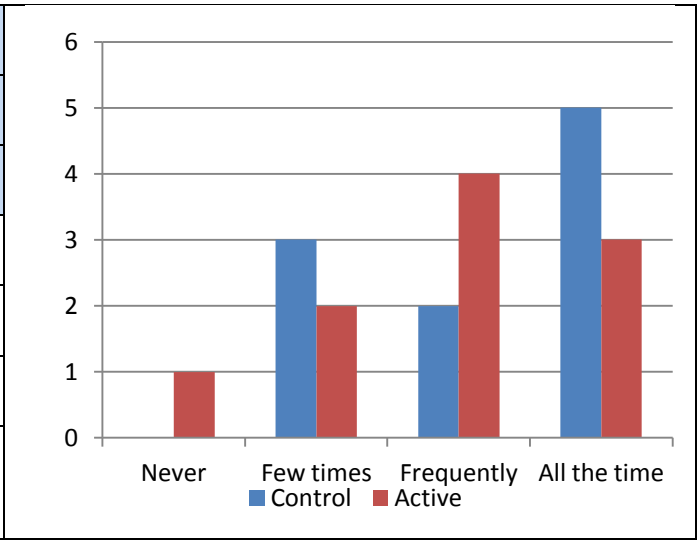


Figure 6-9: The number of Fixed Bugs in each group (control vs. active)

As far as switching roles is concerned the Control groups did more switching than the active groups which speaks to their relative lack of knowledge and awareness compared to the active groups. The Control groups also fixed more bugs which implies their lack of awareness of certain already established mechanisms to guard their produced software against vulnerabilities when writing code.

2a. Encounter Issues didn't get to fix				
Categories	**%**		**Count**	
	C	**A**	**C**	**A**
Never (0 times)	60	30	6	3
Once (1 times)	20	10	2	1
More than one (2 or more)	20	60	2	6

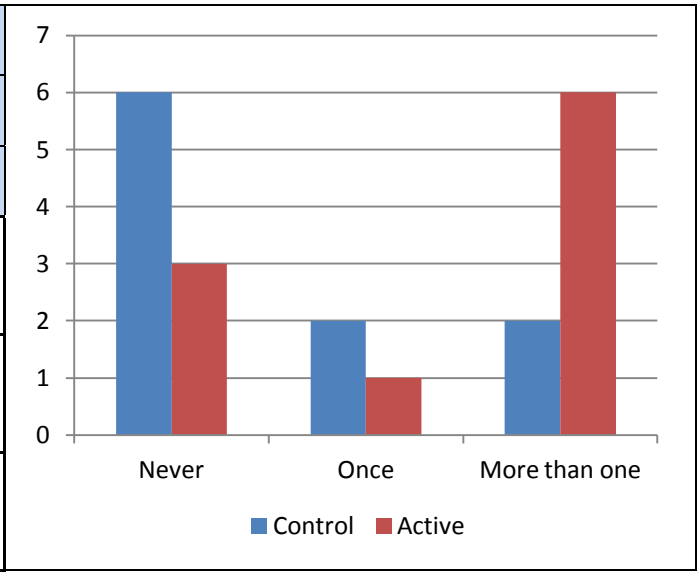


Figure 6-10: The number of Unresolved Issues in each group (control vs. active)

The differences between the knowledge between the Control and Active groups is shown in the number of issues (that they were aware of) that they did not (or were not able to) fix. In this case the Active groups were clearly aware of more issues present in their software that they did not get to fix whether because of time or other limitations. This is a good example of how having more knowledge and awareness leads to more secure software because more time could be requested to fix those issues for example.

2b. Security Discussions				
Categories	**%**		**Count**	
	C	**A**	**C**	**A**
Never (0 times)	10	0	1	2
rarely (1-2 times)	30	50	3	5
sometimes (3-4 times)	40	0	4	0
frequently (5 or more)	20	30	2	3

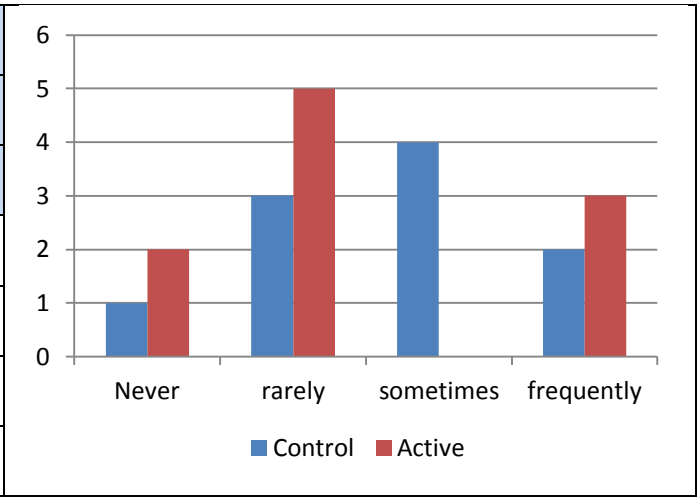


Figure 6-11: The number of Security Discussions in each group (control vs. active)

2c. Other discussions				
Categories	%		Count	
	C	A	C	A
Never (0 times)	0	10	0	1
rarely (1-2 times)	20	30	2	3
sometimes (3-4 times)	20	20	2	2
Frequently (5 or more)	60	40	6	4

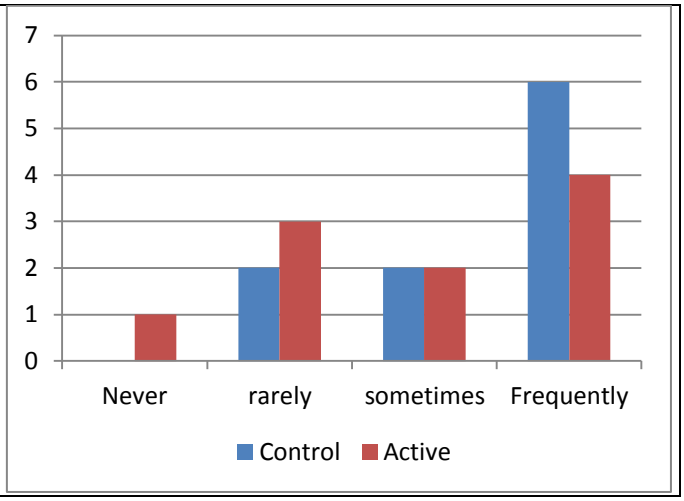


Figure 6-12: The number of Other Discussions in each group (control vs. active)

2d. Use of outside sources				
Categories	%		Count	
	C	A	C	A
Never (0 times)	20	10	2	1
occasionally (1-4 times)	50	60	5	6
Frequently (5-8 times)	20	10	2	1
All the time (9 or more)	10	20	1	2

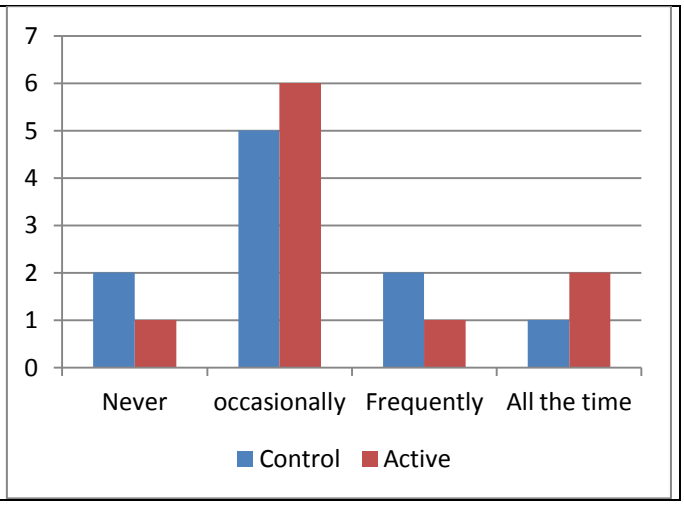


Figure 6-13: The number of other sources used in each group (control vs. active)

As far as having security and other discussions both Control and Active groups show roughly the same number of discussions across the board ranging from “never” to “Frequently”. One interesting point to note is that the Active groups had more frequent security related discussions. The Control groups on the other hand had about the same number of general discussions as the Active groups did. The situation is also the same with the use of outside sources which shows that the Active groups used about the same number of sources with a slight lead for the Active groups in checking more sources all the time.

No	Statement	Control			Active		
		Yes	Don't understand	No	Yes	Don't understand	No
3	**Did you and your teammate discuss or consider…?**						
3a	Client side basic Validation	10			10		
3b	Server side basic validation	9		1	9		1
3c	Client side Regular expression validation	7		3	9		1
3d	Server side Regular expression validation	9		1	7		3
3e	Combination of client side and server side validation	8		2	8		2
3f	Encryption		1	9	1		9
3g	Invalid data submission	9		1	9		1
3h	Cross site scripting		2	8	4		6
3i	Cross site request forgery	1	3	6	4		6

Table 6-15: Some security related activities in each group (control vs. active)

In answering specifically whether or not the team members discussed a certain specific security mechanism the Active groups reported that they mostly did discuss various security topics with the exception of encryption. However, the control group participants had discussions on various specific security mechanisms and some even stated that they did not understand certain specific security mechanisms to begin with which clearly shows their lack of knowledge, awareness, and experience in these areas.

No	Statement			Strongly Agree	Agree	Somewhat Agree	Neutral	Somewhat Disagree	Disagree	Strongly Disagree
4	**Would you say that your awareness of security increased as a result of this work in terms of…**									
4a	Design	Control	Freq.		3	3	4			
		Active	Freq.		1	3	2		2	2
4b	Development/Testing	Control	Freq.		4	4	2			
		Active	Freq.		1	4	1	1	1	2
5	**Would you say that your confidence in your ability to produce software increased as a result of this experiment in terms of…**									
5a	Security	Control	Freq.		4	3	1	2		
		Active	Freq.	1	3	1	2	2		1
5b	Other aspects of software Quality	Control	Freq.		5	4		1		
		Active	Freq.		5	3	1	1		

6	An important consideration for my team was…									
a	Development/Testing the software?	Control	Freq.	3	6	1				
		Active	Freq.	2	5	3				
b	Protecting the software against vulnerabilities?	Control	Freq.	1	3	4	2			
		Active	Freq.		8	2				
7	Would you say that being part of the pair programming team helped you…									
7a	Increase your security knowledge?	Control	Freq.	1	1	3	3		2	
		Active	Freq.		3	2	1	1	1	2
7b	Produce more secure software compared to not doing pair programming?	Control	Freq.		6	2		1	1	
		Active	Freq.	3	1	1	1	1	2	1
7c	Increase your awareness of potential bugs and vulnerabilities?	Control	Freq.	3	5	2				
		Active	Freq.	2	5	3				
7d	Discover security issues/vulnerabilities/bugs ?	Control	Freq.		4	3	2		1	
		Active	Freq.	2	4	2			1	1

Table 6-16: Opinions of participants on various aspects of software development

The answer to the question of whether the awareness of security increased for the Control and Active groups showed that the experiment made the Control group participants more aware of security issues while the Active groups were already aware. As a result, the Active groups did not score as high their increase in awareness answers which shows that they overall possessed more knowledge and awareness before the experiment.

Overall, the active group was considering protecting the software against bugs and vulnerabilities more than the control group while the control group was focused more on the development and testing aspect of the produced software. Not only that the active groups had a higher level of security on their software than the control groups, but also Active groups have less failure rate. This suggests that the experience and the level of knowledge of developers directly impacts the resulting software as it relates to security in addition to quality and functionality.

6.6 Analysis

The results of the experimental trial will be analyzed through a number of statistical procedures including means, standard deviations, and range of scores for the responses to pre and post experiment questionnaires and the produced software. SPSS will be used to conduct Multivariate Analysis of Variance (MANOVA) and t-test on the data.

6.6.1 Descriptive Statistics

This section focuses on using descriptive statistics on our data gathered from the pre-experiment questionnaire, as well as the post-experiment questionnaire. Responses to questionnaire items are used to assess and interpret the results for our experimental trial data. Descriptive statistics attempts to numerically describe the characteristics of the collected data and in our case the experimental data. The assessment of the mean, as a representative score for each item, will be based on the value of the standard deviation, while the interpretation of the mean values will be made on the basis of the various scales we used for each type of question (ex: 7-point Likert scale). The goal is to give an indication of the range of responses given the number of samples that were collected and to summarize the collected data in a meaningful way.

In order to ascertain whether or not a given mean is representative of the majority of the responses, we look at the standard deviation. If the standard deviation is small, it indicates that the individual responses are clustering closer to the mean which shows that the mean is an accurate representation of the majority of the responses (Field 2009). However, if the standard deviation is large, it indicates that individual responses are more scattered and therefore we need to look into additional information to be able to better interpret the nature of the results.

6.6.1.1 Pre-Experiment Questionnaire

Concepts Learned

The scale used to measure the participant's knowledge is as follows: (1) for I have no knowledge of it, (2) for I know very little of it, (3) for I am comfortable with it, (4) for I am proficient (skilled), and (5) for I am an expert.

No	Statement	Control		Active	
		Mean	Std. Deviation	Mean	Std. Deviation
Pre1	**How proficient are you in the following… Object Oriented Programming Languages:**				
Pre1a	Java	3.5	0.7	3.9	0.8
Pre1b	C#	1.7	1.0	3.1	1.4
Pre1c	Ruby	1.3	0.6	1.3	0.6
Pre1d	PHP	2.6	1.2	2.4	0.8
Pre2	**Web Design:**				
Pre2a	HTML	3.6	0.5	4.0	0.9
Pre2b	JavaScript	3.0	0.9	3.6	0.5
Pre3	**Web Development:**				
Pre3a	Servlet/JSP	1.9	1.1	2.8	1.2
Pre3b	ASP/ASP.Net	1.6	0.7	2.9	1.6
Pre3c	PHP	2.7	1.3	2.5	0.7
Pre3d	RubyOnRails	1.4	0.9	1.3	0.6
Pre4	**IDEs:**				
Pre4a	Eclipse/NetBeans	3.4	0.8	4.2	0.9
Pre4b	Visual Studio	2.5	1.0	3.7	1.5
Pre5	**Databases:**				
Pre5a	MySQL	3.1	1.0	4.1	0.5
Pre5b	MS Access	2.1	0.8	2.8	1.4
Pre6	**Software Development Process:**				
Pre6a	Waterfall	3.0	1.0	3.4	0.9
Pre6b	XP	2.0	1.1	2.2	1.3
Pre6c	Scrum	2.8	1.1	3.0	1.1
Pre6d	TDD	1.2	0.4	2.1	1.4
Pre7	**Security Related Topics:**				
Pre7a	Cryptography	2.1	0.7	2.8	0.9
Pre7b	Network Security	2.3	0.6	2.6	1.1

Table 6-17: Descriptive Statistics for Knowledge Level (Active vs. Control)

Even though programming and web related skills and knowledge was not part of the experience index used to assign subjects, the Active group had a higher mean than Control group for many of the skills including Java, C#, HTML, JavaScript, Servlet/JSP, and Eclipse/NetBeans. This means that the more experience the participant has, the more their skills reflect their level of experience.

Additional Experience

The scale used for the additional experience is as follows: (0) for I have no experience, (0.25) for I have up to 3 months of experience, (0.5) for I have between 3 and 6 months experience, (1.0) for I have between 6 months and a year experience, and (1.5) for I have more than a year of experience

No	Statement	Control		Active	
		Mean	Std. Deviation	Mean	Std. Deviation
Pre9	**Additional Experience in these categories:**				
Pre9a	Web Administration	0.2	0.4	0.7	0.7
Pre9b	Web Application Development	0.3	0.4	1.2	0.5
Pre9c	Web Design	0.5	0.5	0.9	0.6
Pre9d	Network Administration	0.0	0.1	0.5	0.7
Pre9e	Database Administration	0.2	0.4	0.8	0.5
Pre9f	Testing & Quality Assurance	0.2	0.4	0.7	0.6
Pre9g	Programming	1.2	0.4	1.5	0.0
Pre9h	Software Engineer/Architecture	0.5	0.4	1.2	0.4

Table 6-18: Descriptive Statistics for Experience Level (Active vs. Control)

Industrial Experience

No	Statement	Control		Active	
		Mean (Months)	Std. Deviation	Mean (Months)	Std. Deviation
Pre10	**Industrial Experience:**				
Pre10a	Internships	1.1	1.8	5.0	7.0
Pre10b	Real Job	1.0	2.0	11.3	11.5

Table 6-19: Descriptive Statistics for Industrial Experience Level (Active vs. Control)

Upon going over the descriptive statistics for the additional questions an important point to consider is the fact that the Active group participants are more experienced in all categories compared to the control group participants which is what was desired by our classification method. The Active group had a much higher mean than Control group when it comes to their industrial experience for both internships and real-jobs which is also in agreement with our strategy to divide participants based on their overall experience.

6.6.1.2 Software Security Metrics

6.6.1.2.1 Group Means

For software security metrics, the only descriptive statistic used here is the mean group score which we derived by applying the assessment scales to get the resulting scores for all the teams for each group type. In our case we had 5 control and 5 active groups and the mean is the sum of the aggregate score for each group assessment result divided by the total group number.

Group Type	**Mean**
Control group	**11.4**
Active group	**18.4**

Table 6-20: Mean Security Assessment Score (Active vs. Control)

The mean score for the Active group, in addition to the simple aggregate score, shows an almost two-fold increase in the security of the produced software compared to the Control groups when it comes to the security of the software produced which shows a higher overall level of security than the control groups which speaks also to their increased awareness as well. Again this suggests positive support for our hypotheses (in descriptive statistical terms) which stated that the Agile team with an experienced developer would have an increased level of security in their produced software.

6.6.1.3 Post-Experiment Questionnaire

Quantitative Questions

No	Statement	Control		Active	
		Mean	Std. Deviation	Mean	Std. Deviation
1	**How many times did you as an individual …**				
1a	Switch your places as the observer and driver?	5.6	5.6	4.8	4.0
1b	Discover bugs and/or vulnerabilities during development/testing?	7.6	3.1	7.9	6.2
1c	Fix bugs and/or vulnerabilities during development/testing?	6.6	2.9	5.6	3.7
2	**As a team, how many times you and your teammate…**				
2a	Encounter issues and/or vulnerabilities that you did not get to fix?	0.6	0.8	1.6	1.5
2b	Discuss issues regarding Security	2.9	1.5	2.8	3.2
2c	Discuss issues regarding other aspects of Quality	4.5	1.7	5.1	5.8
2d	Use outside sources to obtain security information in order to complete the task	3.3	3.8	3.6	3.6

Table 6-21: Descriptive Statistics for Activities in each group (control vs. active)

The scale used for the security related questions is as follows: (-1) for No, (0) for Don't understand, and (1) for Yes.

No	Statement	Control		Active	
		Mean	Std. Deviation	Mean	Std. Deviation
3	**Did you and your teammate discuss or consider…?**				
3a	Client side basic Validation	1.0	0.0	1.0	0.0
3b	Server side basic validation	0.8	0.6	0.8	0.6
3c	Client side Regular expression validation	0.4	0.9	0.8	0.6
3d	Server side Regular expression validation	0.8	0.6	0.4	0.9
3e	Combination of client side and server side validation	0.6	0.8	0.6	0.8
3f	Encryption	-0.9	0.3	-0.8	0.6
3g	Invalid data submission	0.8	0.6	0.8	0.6
3h	Cross site scripting	-0.8	0.4	-0.2	1.0
3i	Cross site request forgery	-0.5	0.7	-0.2	1.0

Table 6-22: Descriptive statistics for some security activities (control vs. active)

Likert Scale Questions

The used Likert scale rating is: (7) for Strongly Agree, (6) for Agree, (5) for Somewhat Agree, (4) for Neutral, (3) for Somewhat Disagree, (2) for Disagree and (1) for Strongly Disagree.

No	Statement	Control		Active	
		Mean	Std. Deviation	Mean	Std. Deviation
4	**Would you say that your awareness of security increased as a result of this work in terms of…**				
4a	Design	4.9	0.8	3.5	1.8
4b	Development/Testing	5.2	0.7	3.7	1.8
5	**Would you say that your confidence in your ability to produce software increased as a result of this experiment in terms of…**				
5a	Security	4.9	1.1	4.5	1.8
5b	Other aspects of software Quality	5.3	0.9	5.2	1.0
6	**An important consideration for my team was…**				
6a	Development/Testing the software?	6.2	0.6	5.9	0.7
6b	Protecting the software against bugs/vulnerabilities?	5.3	0.9	5.8	0.4
7	**Would you say that being part of the pair programming team helped you…**				
7a	Increase your security knowledge?	4.4	1.5	3.9	2.0
7b	Produce more secure software compared to not doing pair programming?	5.1	1.4	4.4	2.3
7c	Increase your awareness of potential bugs and vulnerabilities?	6.1	0.7	5.9	0.7
7d	Discover security issues/vulnerabilities/bugs?	4.9	1.2	5.1	2.0

Table 6-23: Descriptive Statistics for opinions of participants

By completing this experimental trail, the Control groups benefited more than the Active groups. This shows that the hands-on experience through developing software (along with the pre and post experiment questionnaires) increased the awareness of security of less experienced developer(s), allowing them to become more experienced (and aware) developer(s) than before the start of the experiment. This is however not the case for more experienced developer(s) where the aforementioned increase did not exists due to the fact that they were already aware and knowledgeable of many issues including security before the beginning of the experiment. Therefore, in terms of the awareness of security, the participants in the Active groups had more experience to start with and therefore the experiment did not add much more to their knowledge base in

terms of design, development, and testing of the software produced. When it came to the increased confidence in software security, the active group's confidence grew at pretty much the same rate as the control group after producing the software.

On the important consideration of the team, the active group was considering protecting the software against bugs and vulnerabilities more than the control group while the control group was focused more on the development and testing aspect of the produced software. Being part of the Agile team, the control group benefitted more from the experiment in terms of increased security knowledge and producing a more secure software.

6.6.2 Statistical Testing

6.6.2.1 T-test on Experience Index

As part of this experimental trail, participants have been assigned to various groups based on their self-reported experience on the pre-experiment questionnaire. In order to ensure that Control and Active groups are indeed differing based on each group's experience and since different participants have been used in each experimental group type, an independent t-test is performed to see if the mean score of the experience index for the Active group is higher than the mean score of the Control group. For this to be done, we will statistically test the experience index used to rank participants by comparing the experience mean score of Active group with the experience mean score of the Control group.

Group Statistics

	GroupType	N	Mean	Std. Deviation	Std. Error Mean
Experience Index	Active	10	19.2250	5.24067	1.65724
	Control	10	8.2000	2.55713	.80863

Table 6-24: Statistics for Experience Index (Active vs. Control)

Independent Samples Test										
		Levene's Test for Equality of Variances		t-test for Equality of Means						
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	95% Confidence Interval of the Difference	
									Lower	Upper
Experience Index	Equal variances assumed	2.580	.126	5.979	18	.000	11.02500	1.84400	7.15089	14.89911

Table 6-25: Independent Samples t-test on Experience Index (Active vs. Control)

There is an assumption of equal variances when performing the t-test that must hold in order for us to look at the significance which tells us that there is a statistical difference between the Control and Active groups if the value of the significance is less than 0.05.

To check whether this assumption holds, Levene's test was performed which showed that *p*-value is non-significant ($p = 0.126 > 0.05$) which means that the assumption of homogeneity of variances is met. Looking at t-test value when equal variances is assumed, the 2-tailed significance value indicates that there are significant differences between the mean scores of experience index between our experimental groups (indicated by p=.000).

Since we want to see if one group's mean score is higher than the other which in our case translates to checking to see if the Active Group's mean score is higher than the Control group's mean score one-tailed t-test is required rather than a normal two-tailed. SPSS has no option to specify one-tailed for t-test so we had to check this manually by looking up the value in the critical t-distribution table (Field 2009). For critical t-value with 18 degrees of freedom and α of 0.05 (one-tailed) the result is 1.7341. In a one-tailed t-test if the t-value of a solution is 5.979 which is higher than 1.7341 we can conclude that our mean experience score for Active group is significantly higher than the mean experience score for the Control group.

6.6.2.2 MANOVA on 2 Themes

Since we have used several indicators/dependent variables in our experimental trial, Multivariate Analysis of Variance (MANOVA) was chosen to test our experimental hypotheses (Field 2009). To perform MANOVA for each hypothesis, we have grouped each question in the post-experiment questionnaire to its more relevant theme.

6.6.2.2.1 Awareness of Security

Hypothesis	Survey Question
H1: Awareness of Security	1a
	1b
	2b
	2c
	2d
	4a
	4b
	7c
	7d

Table 6-26: H1 and its related Variables

Multivariate Tests						
Effect		Value	F	Hypothesis df	Error df	Sig.
GroupType	Pillai's Trace	.363	.633	9.000	10.000	.748

a. Exact statistic

Table 6-27: MANOVA on Awareness of Security (Active vs. Control)

Table 6-27 shows the MANOVA Pillai's Trace results for awareness of security theme. For the experimental trial, the Control and Active groups show us whether or not participants' experience had an effect on the awareness of security. To assess this, Pillai's Trace results are non-significant ($p = 0.748 > 0.05$) which means that there are no significant differences between the mean scores of these variables between our experimental groups.

6.6.2.2.2 Software Security

Hypothesis	Survey Question
H2: Software Security	1c
	2a
	5a
	5b
	6a
	6b
	7a
	7b

Table 6-28: H2 and its related Variables

Multivariate Tests						
Effect		Value	F	Hypothesis df	Error df	Sig.
GroupType	Pillai's Trace	.461	1.175[a]	8.000	11.000	.392

a. Exact statistic

Table 6-29: MANOVA on Software Security (Active vs. Control)

Table 6-28 shows MANOVA's Pillai's Trace results for software security theme. Again, the results are non-significant ($p = 0.392 > 0.05$) which means that there are no significant differences between the mean scores of these variables between our experimental groups.

6.6.2.3 MANOVA on Increased Security Awareness

From amongst the questions asked on the post-experiment questionnaire, questions 4a and 4b represent increased security awareness. We decided to perform MANOVA on these two questions in order to test the differences between the Active and Control groups when it comes to their awareness of security. The Active group had a smaller mean than the control group partially because of the fact that the participants had more experience to start with and therefore the experiment did not add much more to their knowledge base in terms of design, development, and testing of the software.

Multivariate Tests						
Effect		Value	F	Hypothesis df	Error df	Sig.
GroupType	Pillai's Trace	.241	2.700[a]	2.000	17.000	.096

a. Exact statistic

Table 6-30: MANOVA on Increased Security Awareness (Active vs. Control)

Table 6-30 shows Pillai's Trace results for increased awareness of security as a result of this experiment. The results are non-significant (p = 0.096 > 0.05) which shows that there are no significant differences between the mean scores of these variables associated to increased awareness of security between our experimental groups.

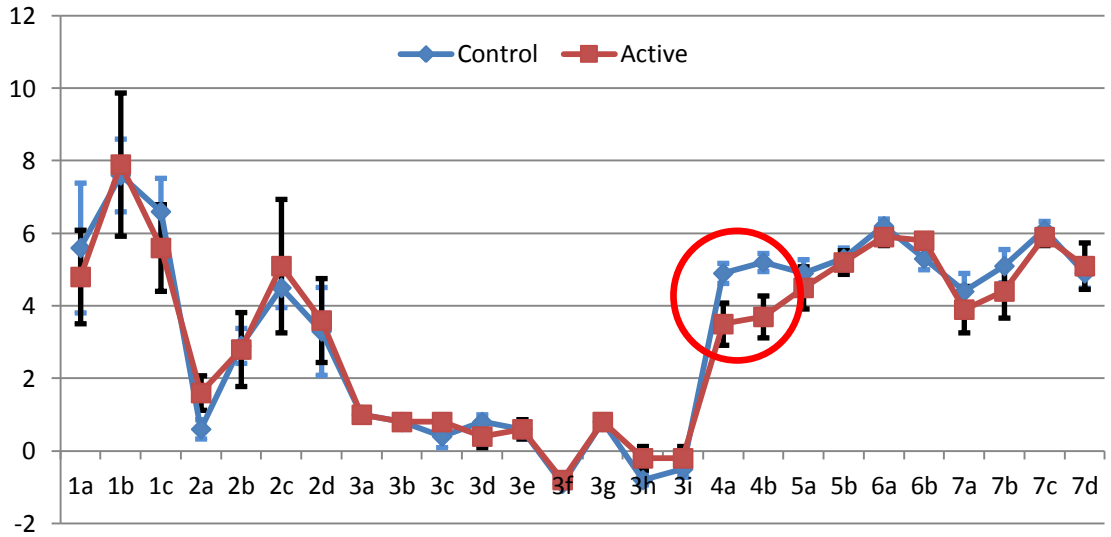


Figure 6-14: Means and its Standard Error for Active and Control

By plotting the mean and its standard error for all post-experiment questions for Active and Control groups as shown in Figure 6-14, we found that the mean standard errors for 4a and 4b question didn't overlap which means it's unlikely that these mean differences happened by chance. This is also corroborated with Uni-variate analysis for these two questions where we found that there is statistical significance for 4a ($p = 0.043 < 0.05$) and 4b ($p = 0.028 < 0.05$). However, since MANOVA takes the correlation between the dependent variables into account, the significance was not achieved in that case $p = 0.096$ (Field 2009).

6.6.2.4 T-test on Software Security Metrics Final Score

In this section, an independent t-test is performed to see if the mean final score of the produced software's security rating for the Active group is significantly different than the mean score of the Control group.

Group Statistics					
	GroupType	N	Mean	Std. Deviation	Std. Error Mean
FinalScore	Active	5	18.4000	10.43072	4.66476
	Control	5	11.4000	11.25500	5.03339

Table 6-31: Statistics for Security Assessment Final Score (Active vs. Control)

Independent Samples Test										
		Levene's Test for Equality of Variances		t-test for Equality of Means						
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	95% Confidence Interval of the Difference	
									Lower	Upper
Final Score	Equal variances assumed	.243	.635	1.020	8	.338	7.00000	6.86258	-8.82514	22.82514

Table 6-32: Independent Samples t-test on Assessment Scores (Active vs. Control)

To check whether equal variances assumption holds, Levene's test was performed which showed that *p*-value is non-significant ($p = 0.6 > 0.05$) which means that the assumption of homogeneity of variances is met. Looking at t-test value when equal variances is assumed, the 2-tailed significance value indicates that there are no significant differences between the mean final score of the software security rating between our experimental groups (indicated by $p = .338$).

6.6.2.5 Chi-square test on Software Security Assessment

Another way to compare the software produced from Active and Control groups is by analyzing the frequencies of the type of techniques used (Basic and Advanced) in the software. This can be done by using Chi-square to test the relationship between two categorical variables.

More Experienced or Less Experienced? * Techniques Crosstabulation

			Techniques		Total
			Basic	Advanced	
More Experienced or Less Experienced?	Control	Count	12	4	16
	Active	Count	12	11	23
Total		Count	24	15	39

Table 6-33: Crosstabulation Techniques used on each Combination of Categories

Table 6-33 shows that an equal number of basic techniques were used by both Active and Control groups. In terms of advanced techniques, out of 15 used by all teams, 11 of them were made by Active groups. While both Active and Control groups had the same number of basic techniques used in their software, the Active groups exceeded the number of advanced techniques used by almost three times the number for the control groups.

To ensure the accuracy of chi-square, the expected counts have to exceed certain values. For 2X2 contingency tables, expected value should be bigger than 5 (Field 2009). The minimum expected count for us is 6.15 which is bigger than 5. For chi-square, the results are non-significant ($p = 0.15 > 0.05$) which shows that group type doesn't significantly influence the number and type of techniques used in the software produced.

6.7 Interpretation of results

For the experimental phase of the research, we set out to further investigate two outstanding recommendations (that formed our hypotheses) from the results obtained through our qualitative and quantitative methods of in-depth interviews followed by a comprehensive questionnaire from practitioners in the field. The hypotheses were:

- H1: Active Group is more aware of security than the base group doing the same programming tasks.
- H2: Active Group produces more secure software than the base group doing the same programming tasks.

To check whether or not the experienced developer(s), as part of the Active Group, are indeed more aware of security than relatively less experienced developer(s) (control Group), we looked at the increased level of awareness of security as a result of the experimental trial. To examine whether the existence of the more experienced developer(s) increases the security of the software, a static code review assessment procedure was developed to see whether more experienced developers could produce more secure software as hypothesized.

To test our strategy of dividing participants based on their overall experience, we performed a t-test on the calculated experience index for both groups and found that there is statistical significance between the Active and Control groups. It showed that the Active group had a significantly higher mean than the Control Group which means they were significantly more experienced statistically.

When it comes to the awareness of security (H1), MANOVA showed non-significance possibly due to the fact that it takes the correlation between the dependent variables into account. However, the Uni-variate Analysis of our main indicators (4a and 4b) for awareness of security showed statistical significance.

The hypothesis regarding the security of the produced software (H2) did not reach the level of statistical significance according to our t-test even though the descriptive statistics on the level of the achieved security of the produced software showed an increased level of security between the Control and Active groups which were based on

the mean for each group's final security assessment score. A non-significant result indicates that the increase was within the measurement error but a more careful examination of the data reveals an interesting situation. We noticed a rather small overlap in the standard error confidence intervals for the two means and we decided to further refine the data to see if the difference between the two means could fall outside the confidence interval (see figure 6-15).

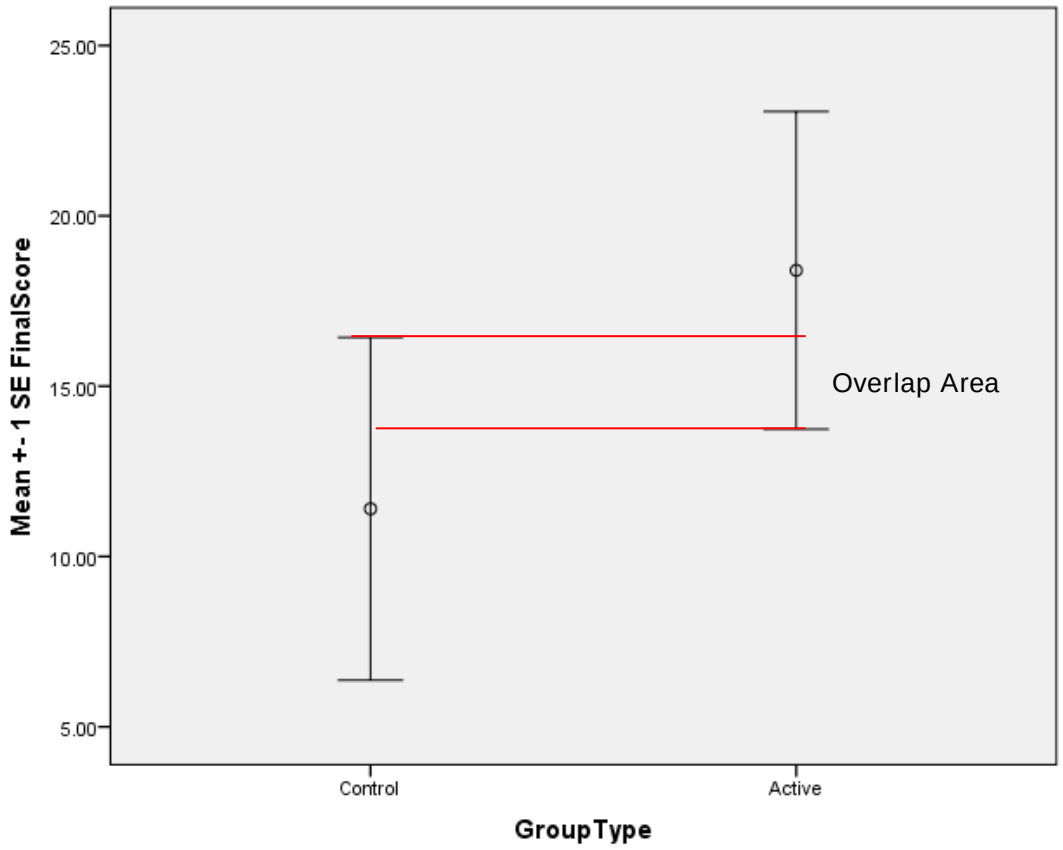


Figure 6-15: Assessment score means and their standard error with outlier

Looking back at the final security assessment scores and by excluding an outlier (Team1 score), the results now show an apparently significant difference between the two means. The graph shown in Figure 6-16 illustrates this.

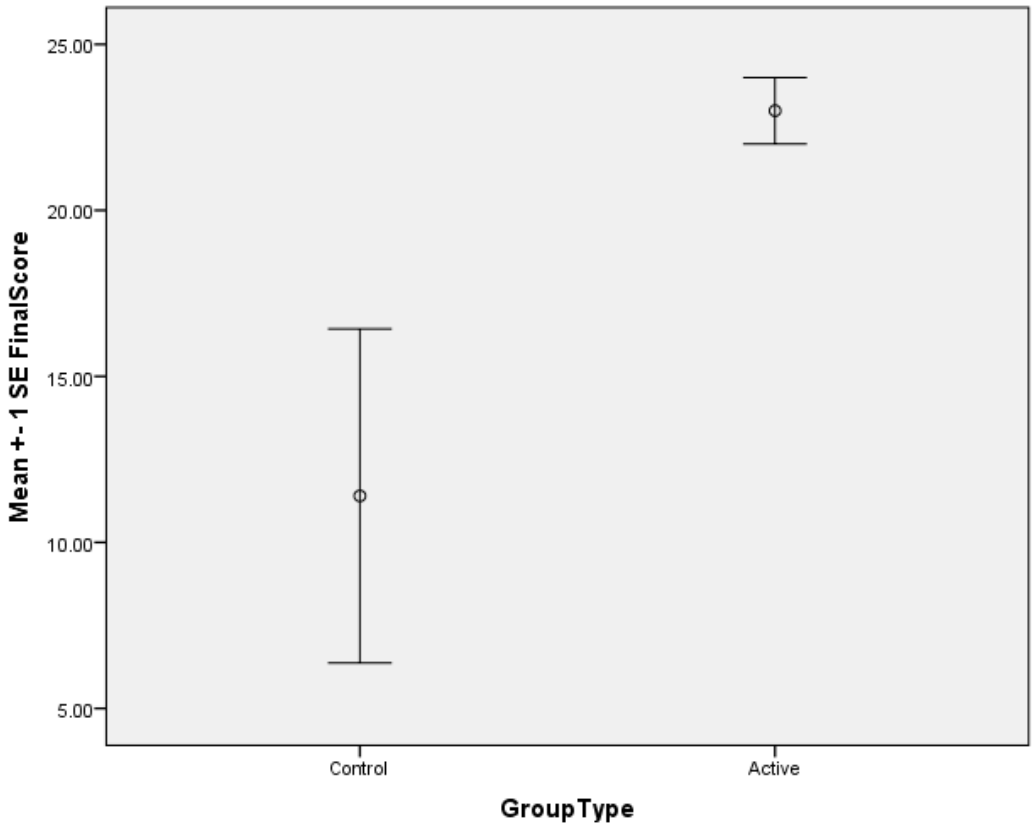


Figure 6-16: Assessment score means and their standard error without outlier

This result would normally suggest a statistical difference between the groups but performing a t-test on the results with the outlier removed still failed to show significance even though it decreased the probability to p=0.082 which is a modest decrease from our original value of p=0.332.

The above results, while promising, still did not produce the level of statistical significance even with the outlier removed in the case of H2. The outlier was chosen because its z-score for the active group was very close to the threshold of 1.96 (Field, 2009). However this was not true for the z-scores for the control groups which is why they were not excluded as well. There could be a variety of possible reasons why some differences are not significant while others were. One possibility could be because the hypothesis is false, but this conclusion seems improbable because we have indeed found affirmations of the hypotheses in some cases and also the practitioners on more than one occasion (interviews and surveys) affirmed it. Another possibility could be because of the sample size but a more careful examination reveals that it is not so much the number

of the participants but it is more the choice of participants. More specifically, the differences between participant's range of experiences, while significantly different, were not enough to produce the significant results on the final produced software. It is well known and understood both from the literature and from other published research that while using students as reasonable candidates in place of practitioners, this choice limits the range of experiences that we could have in terms of the experience of the participants which we believe is the most probable reason why the results did not reach the level of significance in some cases. The researcher's preference is therefore the choice of participants in terms of their relative experience that may have contributed to the above results.

Since our experimental trial was done in an academic setting, the participants were more homogeneous having similar experience that couldn't be discriminated further in our academic setting, therefore we did not get significant statistical differences between the Active and Control groups. Conducting this experiment with participants with more range of experiences (such as practitioners) would allow for increased experience gap between the participants and subsequently may increase these found differences to a level of statistical significance. Our academic findings are consistent with a number of other similar studies that also corroborate with our experience in conducting academic level experiments. According to Fenton (1997) "The problem is particularly noticeable in investigations of programmers, where evaluative and experimental research is the norm. Many findings reported in the literature involve small, student projects; because of constraints on cost and time, researchers refrain from doing large-scale, realistic studies" (Fenton and Pfleeger 1997). There are even more studies that also present a similar view as we have expressed here when it comes to using students (Sjoberg, Anda et al. 2002; Carver, Jaccheri et al. 2003). This indicates that even if we were to expand the experimental trial to include more students, or even a longer trial time span, it is unlikely to get participants with a wider experience gap than we already have. For the future work, the focus should now be put on conducting an empirical study with more experienced professionals in order to see if the desired outcome of knowing with certainty whether using more experienced developers impacts the security of the final produced software in a significant manner.

Overall, we have been able to demonstrate that through our experimental trial, there were differences between Control and Active groups in terms of both their awareness of security and also in terms of the security of the resulting software produced by them. In an academic setting this result is useful to show that the differences indeed exist. In order for us to know what the results would be in the real world, there must be a more elaborate multi-year experimental research performed with more experienced practitioners in order to make the results usable to the population of Agile developers working in the field. Our experimental work was intended to confirm or deny the recommendation made by practitioners which was increasing Agile security by adding more experienced developers on the team in an academic setting which we had time and budget for. Our experimental trial work affirmed statistically our hypothesis related to the awareness of security but failed to achieve statistical certainty for the security of the produced software even though we found interesting patterns in the data that indicate with more experience gap between participants significant results may be achievable.

6.8 Threats to Validity

On the threats to validity for matching, the expense of the experiment is one of the biggest concerns especially in an academic setting (Salkind 1990). Matching may also lead to non-equal groups if turnover occurs in one of the teams (Rosenthal and Rosnow 1991) and therefore we needed to have certain inducements for them as motivation for the completion of the work.

In order to keep the experimental conditions as realistic as possible, there was no control beyond the academic and the related experience level of the students that were chosen for this experiment. While it was possible to further isolate the knowledge of the participants (through their grades, etc.), it would have reduced the population size dramatically and would also increase other risks for the experiment such as turnover and the feasibility of conducting the experimental trials. Also in the industry, the majority of Agile teams have mixed knowledge which is the same as our experiment. Therefore, since similar experiments did not go any further in isolating the group based on grades (Williams, Kessler et al. 2000), we did not feel the need to minimize this kind of

variation further because we consider it to be realistic and acceptable to have mixed knowledge within the same academic level.

We used a process miniature which means that the team was only required to follow certain Agile practices that allowed for knowledge dissemination and transfer between team members. While this may pose a threat to validity in terms of not following all Agile practices, given the time and budget constraints for the experimental trials we chose to focus on the practices that maximized knowledge transfer while keeping the iteration length small to minimize the risk of turnover and reduce variability. For example, some team members did not switch roles during pair programming as part of the iteration for the active group which under normal circumstances would be considered as not following the proper Agile practice. However, considering the small time-frame of the iteration it makes sense for them to switch very few times given the fact that this process spanned only 2-3 hours. There were also other threat(s): for example should we have imposed a requirement for them to switch roles therefore introducing other biases (too much focus on procedure rather than focus on producing solutions) into the experiment.

We tried to simulate real working conditions for the group by assigning each participant with an hourly allowance for their time which was fully paid to them after the conclusion of the experimental period. To motivate participants to commit to the project and not quit in the middle of the experiment, we assigned a bonus for completing the post-experiment questionnaire. This minimized the risk of turnover which is an important threat to validity of the experiments in general and also simulates a real-world practice of payments for the job done on a smaller scale as in a real world setting. According to Dean and Voss (1999) the experiment needs to be as simple as possible and the time allotted should not typically be more than a few weeks if the success of the experiment is desired (Dean and Voss 1999). Furthermore, there are additional ethical issues (the ethics committee) involved in a long term experiment compared with the short term which may hinder the researcher from spending time on the important issues such as in affirming or denying the stated hypotheses. It is with the above reasons that multiple 2-3 hour iterations for the experiment was chosen to allow

for repeatability, lowest turnover, and a simpler and more focused experiment was chosen which received approval by the ethics committee.

The inducement was also a consideration since it could have introduced its own level of bias into the experiments which would persuade the students to commit to the trial only for the monetary reward involved. To minimize this threat we chose the amount small enough that only a truly interested individual would participate and the inducement was there to ensure they did not feel their time was wasted in any way. Thankfully the experimental trials were conducted successfully and with no turnover which was very important to the success and failure of the experimental trials and the people involved expressed satisfaction for having committed to the work overall.

6.9 Summary

This chapter concludes the experimental trial as the final part of our mixed method approach which allowed us to examine some of the benefits the experienced developers can bring to process as well as to the product compared to less experienced developers. Overall, we have been able to show that through our experimental trial, there were statistical differences between Control and Active groups in terms of their awareness of security but not in terms of the security of the produced software. In order to reach the level of statistical significance, there must be a more elaborate and lengthy multi-year experimental research performed with more experienced practitioners (empirically). Our experimental work was intended to confirm or deny the recommendation made by practitioners (from our survey) which was increasing Agile security argument by adding more experienced developers on the team. Our experimental trial work affirmed this only in terms of awareness of security but not for the security of the produced software.

Chapter 7 Conclusions and Future Work

7.1 Introduction

As Agile methods started to gain widespread acceptance and adoption throughout the industry, more and more companies started to replace the traditional methods (such as waterfall) in favor of the more Agile approaches which promised better software quality and faster development cycles. As the adoption progressed however, apparent areas where Agile seemed to have shortcomings became known. One such area is the domain of secure software development and mission critical systems where more careful consideration must be used to design and develop the software in order to successfully accomplish the task.

7.2 Conclusions

7.2.1 The Literature Review Conclusions

There are many studies that suggested additional steps for Agile in order to bolster its security assurance argument but there was little to no empirical evidence in support of how this could be accomplished. Our research attempted to fill this gap by conducting a mixed method empirical study of top security issues in Agile methods to gain a real world perspective of what could be done to improve or provide a security assurance argument for software developed using Agile methods. Specifically, in terms of empirical evaluation, our contribution is in filling the gap that was identified through the evaluation of the security issues and contributions in the past few years and identification and empirical classification of outstanding issues that were agreed upon by practitioners and experts in the field.

7.2.1.1 What steps were taken in order to fill this gap

The aim of the research was to conduct the investigation of security issues in Agile methods in multiple steps or phases. First ranking and then testing the top issue(s) based on the expert opinions of the practitioners and then further substantiating these findings through a wider range quantitative study aimed at gaining more widespread opinions from more practitioners. As a result of our investigation, we developed a procedure to answer significant questions regarding security issues in Agile methods and as a result discovered the most important issues and solutions that can be used in practice with a high degree of consensus and agreement from the industry. Chapter 3 (§3.10) lists the major security issues uncovered throughout the literature and classifies them into two major categories: The Software Engineering and the Security Engineering perspectives. We further added our own perspective to each uncovered issue which developed as the researcher gained more knowledge on these issues and their relevance to the overall Agile security investigation.

7.2.2 The Qualitative Phase Conclusions

The researcher's systematic and comprehensive literature review on the predominant security issues in Agile methodologies yielded a list of major security issues that were proposed by various experts in the field from various backgrounds and points of view. From the conclusions of those studies and related published material we created a set of questions designed to gain in-depth knowledge about the practitioner's opinions on those issues and/or topics. The result of the interviews was a number of findings that we ranked according to their occurrence amongst the practitioners as viable and/or important issues. We also realized that a number of practitioners were not much concerned with security aspects and issues in general and thus we realized our study would have been incomplete had we only elected to generate hypotheses based upon only the perspective of the practitioners. Therefore we employed a data driven approach to inquiry where we used the prior research in addition to the practitioner's opinions to identify major security issues. This method turned out to be most appropriate because we incorporated many issues that had come up in our process of systematic literature review on predominant security issues and could then gauge the practitioner's opinions about them in order to gain a deeper insight into what they thought of them in practical terms.

The results of the ranking and analysis of the responses provided answers to our research questions:

7.2.2.1 RQ1- Combining Security and Agility

In order to answer the question "How to seamlessly combine security and agility, and evaluate the suitability of security mechanisms for use in agile?" the best and most effective practice turned out to be the inclusion of the security engineer as shown by Table 4-4. It came at a relatively higher cost however, than the second most agreed upon solution which was developing software with security in mind. Note that this did not include the added cost of having to hire a dedicated security engineer for the team. The third solution involved the use of security controls in mitigating the need for security but the fact that Standard Security Controls are not yet present could make it relatively more costly than the other proposed solutions. Having informal security experts within the team underscored the fact that while there was no formal security engineer, such a skill was still sought after by various team members and more often than not such a person was unofficially utilized to help on certain security matters. Another major agreed upon solution was how practitioners viewed the relative experience of the people in the development team as having a direct impact on the overall security of the project and how this in and of itself could be used as a way to increase the security of the resulting software for a given project. For more details on the results and our findings refer to chapter 4 (§4.5.1).

7.2.2.2 RQ2 Changing Agile Practices for Security

To be able to better understand the overall issue of "Is changing one or more agile practice(s) for better security really necessary and what kind of projects would require security more than others?" the responses allowed us create a relative scale to interpret the findings. The majority of the practitioners (see Table 4-5) agreed on how the changes to Agile were not really necessary and also from the points of view of Agile vs. Traditional methods, Impact of accelerated schedules on Agile, and reduced security for internal projects, we know that they did not view modification of the Agile process as a viable solution overall. We also gained insight into which projects could use security more than others and the determining factor was to see if the project was for internal company use or it was public facing software which increased its security risk if

compromised. For more details on the results and our findings refer to chapter 4 (§4.5.2).

7.2.2.3 RQ3 Security Issues and Software Vulnerabilities

To answer the research question "To what extent do security issues stem from software vulnerabilities and/or defects?" we have found that the relative experience and knowledge of the developers accounted for almost half of the responses (see Table 4-6). However, since the number of respondents that discussed these topics accounted for as little as 6% of all the responses, we decided not to further investigate the issues surrounding this research question. For more details on the results and our findings refer to chapter 4 (§4.5.3).

7.2.2.4 RQ4 Customer's Control of Security

From the results and answers to research question "Should the customer be able to dictate the level of required security for a given project and how much control should they be given?" it is clear that the customer should indeed have the authority to dictate all requirements for the project including security (see Table 4-7), we saw no further need to continue our investigation into this issue because less than 6% of the respondents discussed it as an important factor overall. For more details on the results and our findings refer to chapter 4 (§4.5.4).

7.2.2.5 RQ5 Knowledge Dissemination and Diffusion

On the question of "What are the effects of knowledge dissemination and diffusion on a given agile project in terms of security awareness of its members, especially developers?", we found out (see Table 4-8) that the apparent lack of developer's security knowledge was most prominent when came to knowledge dissemination and diffusion which means that the number one factor in helping spread the knowledge of security within the team revolves around the relative knowledge and expertise of the developers. The second most contributing factor turned out to be pair programming which is a popular practice in certain flavors of Agile and used as a knowledge dissemination tool to increase the relative knowledge of security and other skills within the team. Almost equally important was achieving security through non-explicit forms of training that could result in increased security for the project. Overall, there were not enough respondents that touched on the subject to warrant more investigation of this

research question and therefore we closed the question to focus on other more popular topics under investigation. For more details on the results and our findings refer to chapter 4 (§4.5.5).

7.2.2.6 RQ6 Dedicated Security Iteration

For the question of "How beneficial is dedicated security iteration and under what circumstances should they be initiated?" we discovered that establishing periodic security was beneficial overall regardless of the circumstances (see Table 4-9). This was especially important to practitioners in projects that were being deployed publically and exposed outside of the organization. This research question garnered very clear answers in both aspects of whether or not to conduct periodic security iteration as well as the overall circumstances that one should consider when they would like to proceed with a project. While we could much more deeply investigate this topic further the relatively small number of responses indicates that we had gained as much as we could on the question and therefore we ceased to continue research on. For more details on the results and our findings refer to chapter 4 (§4.5.6).

7.2.2.7 RQ7 Testing and Verification for Security

In order to find out "To what extent do the testing and verification tools provide additional security assurance in addition to quality assurance?" we found that the majority of the respondents agreed that refactoring could contribute rather significantly towards the increased security of the resulting software if it's done throughout the project (see Table 4-10). Furthermore practicing security focused testing show how strongly testing and verification mechanisms can be used to prove the relative security of the software and contribute to increased security assurance argument for the Agile project. The subject of tools while acknowledged as effective and beneficial does not currently go far enough in terms of providing solid assurances that the resulting software would be secure but they could be used as a good augmenting practice to add to other approaches to further increase the security of the resulting software under development. For more details on the results and our findings refer to chapter 4 (§4.5.7).

7.2.2.8 RQ8 Documentation and Security

On the question of "Does the reduced documentation in agile projects lead to less secure software even with the advent of CASE tools?" we discovered that there are many forms of documentation that are now automatically generated and act as a replacement to the ones used before (waterfall) (see Table 4-11). In addition to this, they have also begun to fill in the blanks where traditional documentation came up short. Documenting the Agile way turned out be the most popular approach that developers and other stakeholders chose to follow and almost as important was the fact that they regarded Agile as not less secure than traditional methodologies in terms of documentation which gave them the confidence to continue innovating and finding new and creative ways to develop software faster. While Misuse and Abuser stories were acknowledged as a good way to look at security issues, the manner and the placement of those stories was a point of contention. A surprising theme that emerged from these interviews was the fact that many practitioners were choosing to forgo documentation in favor of spending the time in creating test cases which added to the overall security assurance and was used as part of the acceptance criteria for the software under development which was both creative and effective. For more details on the results and our findings refer to chapter 4 (§4.5.8).

7.2.3 Developing hypotheses

As we alluded to in earlier remarks in the interpretation of research questions, we chose to develop hypotheses for the top 4 research questions that had more than 9% occurrence based on our frequency analysis of the interview data. This allowed us to focus the research on the most important empirical aspects to continue our investigation on. Table 4-20 shows the overall occurrence percentage for each research question. For each of the top research questions, we developed hypotheses for themes that had more than 9% occurrence within that research question. If time and resources permitted, we would investigate all of the themes and the research questions but due to the limitations of time and resources, and the fact that we were looking to investigate the most important security issues in Agile empirically, the opinion of practitioners was important in this case in determining whether or not to continue investigating a theme or not. Therefore, we set the threshold at 9% or more because it indicates a rather considerable focus by the practitioners on a particular sub-topic or theme and allowed

us to focus more on them for the later stages of the research. Table 4-21 contains the list of hypotheses. From this point on we shifted focus to quantitative aspects where we continued the investigation by testing the hypotheses with questionnaires on various issues and topics raised in the qualitative study combined with the prior research issues that were similar to the one we discovered empirically.

7.2.4 The Quantitative Phase Conclusions

For the quantitative aspect of the research, the questionnaire focused on hypotheses for the top 2 research questions that had more than 20% occurrence (based on our frequency analysis results) in the interviews. The questions for the survey were derived from the results of the interview process which asked a more specific set of questions towards issues related to combining security and Agility and changing Agile practices for security. We chose the questionnaire method as a practical means to gauge more widespread opinions than were found during the in-depth interviews using a larger sample size (120 versus 16). This allowed us to have a more generalize-able answer for these questions that are important to larger audiences such as developers, managers, and other Agile stakeholders who wish to know and understand better how to implement security mechanisms into their software development lifecycles.

7.2.4.1 RQ1- Combining Security and Agility

The results of our factor analysis confirmed what we expected when we measured the suitability of including different proposed solutions to combine security and agility after the interview process. In order to be able to seamlessly combine security and agility and evaluate the suitability of security mechanisms for use in Agile, the addition of the Security Engineer was the best and most suitable solution as 60% of all participants were in support of this solution (§5.4.3). The second most suitable solution was the inclusion of experienced developers within the Agile team as evidenced by 37% of all participants.

Using ANOVA on factor scores (which implies suitability), we examined which solution was the most suitable solution for inclusion into Agile. Through the examination of the significance values and the differences of means, we found that although suitability of including the Security Engineer and Experienced Developers into

Agile team are not significantly higher from each other (p = 0.565), they are significantly higher than other proposed solutions such as the use of Software Security Controls (p = 0.000 for both) and building software with security in mind (p = 0.000 for both). We also found that the suitability of using Software Security Controls in Agile projects is significantly higher than building software with security in mind (p = 0.000). Based on inferential statistics, we can conclude that the most suitable security mechanisms into Agile methods are the inclusion of a dedicated Security Engineer and the use of more experienced developers in the Agile team. Using the one-sample t-test on suitability scores (which are represented by the factor scores) of including the Security Engineer (H1a), Experienced Developers (H1d), and Software Security Controls (H1c) into Agile showed significant values. This was not the case for building software with security in mind (H1b).

7.2.4.2 RQ2 Change Agile Practices for Security

The obtained factor analysis results for RQ2 were not consistent with our expectations when it came to changing Agile for sake of security and therefore, we decided to rely on the original scores of the questionnaire data for further analysis.

The result of two-tailed one-sample t-test rejected the null hypothesis (H2a) which indicated that there was a significant difference between Agile methods and Traditional methods in terms of the security of the produced software. This significant difference favors Agile because it shows that Agile is not worse than waterfall in terms of security as some interviewees had claimed. Moreover, as we have done on the earlier hypotheses, the results of one-tailed one-sample t-test showed that rating scores' mean is significant for awareness of security in Agile (H2b), impact of accelerated schedules in Agile projects (H2c), and reduced security in internal Agile projects (H2d).

The questionnaire represented the quantitative aspect of the investigation into security issues in Agile which allowed us to generalize and test hypotheses regarding various issues and topics raised in the qualitative study combined with the prior research issues that were similar to the one we discovered empirically with a much bigger sample than

what we were able to achieve in the qualitative phase (up to 148 survey respondents compared to 16 interviews).

Achieving a high degree of reliability for both research questions (RQ1 and RQ2) and correct factor structure for RQ1 indicates a high degree of replicability of the obtained results. It's also clear that since we were able to reach a bigger sample size (120-148 respondents from a broad geographical spectrum) with a broad number of industries involved, we can confidently say that the result of this questionnaire is more generalizable to the population of Agile practitioners than our interview results. For more detailed information refer to chapter 5.

7.2.5 The Experimental Trial Conclusions

The experimental trial was the final part of our mixed method approach which we conducted in order to gain a more practical perspective of the results we obtained earlier through our qualitative and quantitative techniques. With better practical knowledge and understanding of security issues and their impact when combined with Agile methodologies, the experiment served as the last perspective in the multiple perspectives on the important issues that were empirically identified and classified throughout our research. For the experimental trial, a subset of an Agile development process was studied in depth to see the effect of having more experienced developers on the resulting software and the awareness of security amongst the team members. The experiment attempted to get more targeted data based on the top issues concerning the suitability of including more experienced developer(s) to the Agile team (as were found through our quantitative survey analysis) which uncovered the most important and suitable security integration mechanisms in Agile today.

Awareness of security showed statistical significance which indicates that the more experienced experimental groups (Active Groups) were indeed more aware of security issues compared to the less experienced Agile groups (Control Groups). In terms of the produced software, the groups with more experienced developers produced more secure software compared to the groups with less experienced developers when it comes to the security of the software. Overall, the results affirmed our hypotheses which stated that

the Agile team with more experienced developer(s) would have an increased level of security awareness as a result which also affirms the practitioner recommendations to the same effect.

7.3 Research Contributions

A mixed method empirical study aimed at investigating security issues in Agile methods focusing on practitioner opinions and attitudes regarding the most predominant issues from the literature and from practitioners themselves. The data that has been collected is in the form of systematic and comprehensive literature review data, empirical interview data, empirical questionnaire responses, produced software code, and two academic surveys (pre and post experiment) which combined with our results, analysis, and interpretation forms the entirety of this research effort. We followed a mixed method approach in gaining a better practical knowledge and understanding of Security issues in Agile methodologies and our final experimental method served as the last perspective in the multiple perspectives on the important issues that were empirically identified and classified through our research.

I. A systematic and comprehensive literature review, identification of the gap in literature, and classification of the most predominant security and related issues in Agile methodologies based on academic and empirical sources.
    a. Classification of the all relevant security issues in the literature from both theoretical and practical perspectives on Agile teams and Agile methodologies (Chp.3 - P.52)
    b. A list of research questions (Chp.3 - P.50)

II. Empirical interviews aimed at gaining in-depth and specific information about the most predominant issues found throughout the literature
    a. Empirical classification and testing of the importance of the issues in terms of their real-world prospects and effects on Agile teams and the Agile methodology
        1. Uncovering of the importance of each issue based on the opinions and perspectives of researchers and experts in academia as well as the industry (Chp.4 - P.68)

b. A list of themes generated as a result of the thematic analysis on the interview data (Chp.4 - P.64)
c. A list of themes deemed most important by practitioners that were investigated further through quantitative survey:
    1. Dedicated Security Engineer (Chp.4 - P.88)
    2. Software with Security in mind (Chp.4 - P.89)
    3. Security Controls (Chp.4 - P.90)
    4. Experience of Developers (Chp.4 - P.90)
    5. Agile vs. Traditional Methods (Chp.4 - P.92)
    6. Awareness of Security to Agile (Chp.4 - P.92)
    7. Impact of Accelerated Schedule (Chp.4 - P.92)
    8. Reduced Security for Internal Projects (Chp.4 - P.93)

III. Empirical Survey aimed at validating the combined results of the literature plus the interview results to gain a widespread and generalize-able results from practitioners about the most important and beneficial solutions to add a security assurance argument to Agile methods
    a. Ranked proposed integration solutions using ANOVA based on their suitability of inclusion into Agile methodologies (Chp.5 - P. 191)
    b. Provided evidences to support hypotheses with statistical significance
        1. The addition of the dedicated security engineer is suitable for inclusion into Agile. (Chp.5 - P.193)
        2. Use of security controls is suitable for inclusion into Agile. (Chp.5 - P.193)
        3. The use of more experienced developers is suitable for inclusion in Agile. (Chp.5 - P.193)
        4. Adding practices aimed at increasing awareness of security is necessary for integration with Agile. (Chp.5 - P.195)
        5. Accelerated timeframes and schedules of Agile contribute to less secure software. (Chp.5 - P.195)

6. Internal Agile projects can ignore the security aspects of the software. (Chp.5 - P.195)

IV. An experiment involving students at various academic levels aimed at awareness of security in XP flavor of Agile and also to gauge the relative security of the final produced software

a. Experimentation on one of the themes based upon the results of the questionnaire which used the overall consensus on the software security in Agile today. (Chp.5 - P. 191)

1. Experience of Developers: The idea that the more knowledgeable and experienced the developer, the more the resulting software will be secure.

a. Pre-Experiment questionnaire to capture background and experience information from students

2. Classification and usage of candidates based on their experience index (Chp.6 - P.215)

b. Development of a secure online form to gage the security of the final produced software

c. Post-Experiment questionnaire to capture metrics related to their awareness and experiences working as part of an Agile team

d. List of hypotheses tested by the experiment

3. Active Group is more aware of security than the base group doing the same programming tasks. (Chp.6 - P.245)

a. Statistically Supported

4. Active Group produces more secure software than the base group doing the same programming tasks. (Chp.6 - P.249)

a. Found a difference but not statistically supported

As with any empirical research, each method carries with itself a number of threats to validity which if not carefully considered and guarded against may distort and weaken the results and the strength of the conclusions of the study. We made every effort to

minimize and mitigate these threats for each method in our mixed method approach and each chapter has a section dedicated to addressing and minimizing these threats to ensure a consistent and robust conclusion is reached which considered a number of threats in the design and in the data gathering and analysis stages.

7.4 Future Work

Our literature review identified the key issues related to security and Agile, and specifically looked for issues that are common to both disciplines. From there we chose a number of important issues that are currently being discussed in research and practice and prepared interviews, surveys, experiments and analysis schemes from this information as a basis for the research. As the available time and resources permitted, we outlined an approach that picked the most practical and empirically accepted path forward which meant we could not follow through with testing and experimentation on some less popular proposed solutions. As such, we will outline these here so that those looking to pursue other possible avenues may continue the areas where we did not pursue further due to time and resource limitations.

1. Empirical Survey(s) for the hypotheses related to RQ7 and RQ8
2. Experiments with practitioners for issues related to RQ1, RQ2, and possibly RQ7 and RQ8 should empirical surveys yield significance from practitioners in the field.
3. More elaborate and in-depth studies such as empirical case studies involving one of more proposed solutions in real Agile team(s).
4. More in-depth investigation into RQ2 in terms of the factor extracted in section 5.6.4.2

In order for us to know what the results would be in the real world, there must be a more elaborate multi-year experimental research performed with practitioners which can increase the experience gap beyond what we were able to achieve with our experimental trial.

References

Abbas, N., A. M. Gravell, et al. (2008). Historical roots of Agile methods: where did "Agile thinking" come from? Agile Processes in Software Engineering and Extreme Programming, Springer: 94-103.

Abbas, N., A. M. Gravell, et al. (2010). Using factor analysis to generate clusters of agile practices (a guide for agile process improvement). Agile Conference (AGILE), 2010, IEEE.

Abrams, M. D. (1998). Security engineering in an evolutionary acquisition environment. Proceedings of the 1998 workshop on New security paradigms. Charlottesville, Virginia, United States, ACM: 11-20.

Alexander, I. (2002). Initial industrial experience of misuse cases in trade-off analysis. Proceedings of the 10th IEEE Joint International Conference on Requirements Engineering, 2002. .

Amey, P. and R. Chapman (2003). Static verification and extreme programming. Proceedings of the 2003 annual ACM SIGAda international conference on Ada: the engineering of correct and reliable software for real-time & distributed systems using ada and related technologies. San Diego, CA, USA, ACM: 4-9.

Amoroso, E. G. (1994). Fundamentals of computer security technology, Prentice-Hall, Inc.

Atkinson, J. W. (1958). Motives in fantasy, action, and society: a method of assessment and study. New York, D. Van Nostrand.

Atkinson, L. (1988). "The measurement-statistics controversy: Factor analysis and subinterval data." Bulletin of the Psychonomic Society 26: 361–364.

Aydal, E., R. Paige, et al. (2006). Security Planning and Refactoring in Extreme Programming. Extreme Programming and Agile Processes in Software Engineering. P. Abrahamsson, M. Marchesi and G. Succi, Springer Berlin / Heidelberg. 4044: 154-163.

Babakus, E., C. E. Ferguson, Jr., et al. (1987). "The Sensitivity of Confirmatory Maximum Likelihood Factor Analysis to Violations of Measurement Scale and Distributional Assumptions." Journal of Marketing Research 24(2): 222-228.

Babbie, E. R. (1990). Survey research methods, Wadsworth Pub. Co.

Babbie, E. R. (2007). The practice of social research, Thomson Wadsworth.

Baca, D. (2010) "Developing Secure Software in a Agile environment."

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

Berg, C. and S. W. Ambler (2006) "Assurance & Agile Processes."

Beznosov, K. (2003). Extreme Security Engineering: On Employing XP Practices to Achieve "Good Enough Security" without Defining It. First ACM Workshop on Business Driven Security Engineering (BizSec).

Beznosov, K. and P. Kruchten (2004). Towards agile security assurance. Proceedings of the 2004 workshop on New security paradigms. Nova Scotia, Canada, ACM: 47-54.

Bishop, M. (2003). Computer security: art and science, Addison-Wesley.

Boehm, B. and R. Turner (2003a). "Using risk to balance agile and plan-driven methods." Computer 36(6): 57-66.

Boehm, B. and R. Turner (2003b). Balancing agility and discipline: a guide for the perplexed, Addison-Wesley.

Bogdan, R. and S. K. Biklen (1992). Qualitative research for education: an introduction to theory and methods, Allyn and Bacon.

Borg, W. R., J. P. Gall, et al. (1993). Applying educational research: a practical guide, Longman.

Boström, G., J. Wäyrynen, et al. (2006). Extending XP practices to support security requirements engineering. Proceedings of the 2006 international workshop on Software engineering for secure systems. Shanghai, China, ACM: 11-18.

Boyatzis, R. E. (1998). Transforming qualitative information: thematic analysis and code development, Sage Publications.

Briggs, S. R. and J. M. Cheek (1986). "The role of factor analysis in the development and evaluation of personality scales." Journal of Personality 54(1): 106-148.

Bryman, A. (2001). Quantitative Data Analysis with SPSS Release 10 for Windows: A Guide for Social Scientists, Taylor & Francis.
Byrne, B. M. (2010). Structural equation modeling with AMOS: basic concepts, applications, and programming, Routledge.
Campbell, D. T. and D. W. Fiske (1959). "Convergent and discriminant validation by the multitrait-multimethod matrix." Psychological Bulletin 56(2): 81-105.
Carver, J., L. Jaccheri, et al. (2003). Issues in using students in empirical studies in software engineering education. Software Metrics Symposium, 2003. Proceedings. Ninth International.
CC (1999). Common Criteria for Information Technology Security Evalution. Version 2.1.
Chao, J. and G. Atli (2006). Critical personality traits in successful pair programming. Agile Conference, 2006.
Chivers, H., R. F. Paige, et al. (2005). Agile Security Using an Incremental Security Architecture. Extreme Programming and Agile Processes in Software Engineering. H. Baumeister, M. Marchesi and M. Holcombe, Springer Berlin / Heidelberg. 3556: 57-65.
Chong, L. (2009). "Forrester: From Agile Development To Agile Engagement." Retrieved 01/03/2010, from http://blogs.msdn.com/b/lchong/archive/2009/05/21/forrester-from-agile-development-to-agile-engagement.aspx.
Chung, M.-W. and B. Drummond (2009). Agile at Yahoo! From the Trenches. 2009 Agile Conference.
Churchill, G. A., Jr. (1979). "A Paradigm for Developing Better Measures of Marketing Constructs." Journal of Marketing Research 16(1): 64-73.
Cliff, N. (1996). Ordinal methods for behavioral data analysis. Mahwah, NJ, Lawrence Erlbaum.
Cockburn, A. (2002). Agile software development, Addison-Wesley.
Cockburn, A. (2004). Crystal clear: a human-powered methodology for small teams, Addison-Wesley.
Coffey, A. and P. Atkinson (1996). Making sense of qualitative data: complementary research strategies. Thousand Oaks, CA, Sage Publications.
Cohen, J. (1988). Statistical Power Analysis for the Behavioral Sciences, L. Erlbaum Associates.
Coleman, K. (2009). "How to Achieve More 'Agile' Application Security." Retrieved 29/04/2010, from http://www.computerworld.com/s/article/9128668/How_to_Achieve_More_Agile_Application_Security.
Collier Jr, R. O., F. B. Baker, et al. (1967). "Estimates of test size for several test procedures based on conventional variance ratios in the repeated measures design." Psychometrika 32(3): 339-353.
Corbin, J. M. and A. L. Strauss (2007). Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory, SAGE-USA.
Cortina, J. M. (1993). "What is coefficient alpha? An examination of theory and applications." Journal of Applied Psychology 78(1): 98-104.
Creswell, J. W. (2007). Qualitative inquiry & research design: choosing among five approaches, Sage Publications.
Creswell, J. W. (2008). Educational research: planning, conducting, and evaluating quantitative and qualitative research, Pearson/Merrill Prentice Hall.
Creswell, J. W. (2009). Research Design: Qualitative, Quantitative, and Mixed Methods Approaches, SAGE Publication, Inc. .
Creswell, J. W. and M. L. Brown (1992). "How Chairpersons Enhance Faculty Research: A Grounded Theory Study." The Review of Higher Education 16(1): 41-62.
Creswell, J. W. and V. L. P. Clark (2007). Designing and conducting mixed methods research, SAGE Publications.
Curran, J. and R. A. Blackburn (2001). Researching the small enterprise, SAGE Publications.
De Vaus, D. (2002). Surveys in Social Research 5th Edition, Taylor & Francis.
Dean, A. M. and D. Voss (1999). Design and analysis of experiments, Springer.

Denzin, N. K. and Y. S. Lincoln (2005). The SAGE handbook of qualitative research, Sage Publications.
Di Lucca, D., A. Fasolino, et al. (2002). Testing Web Applications. 18th IEEE International Conference on Software Maintenance (ICSM'02). Montreal, Quebec, Canada
Di Lucca, G. A. and A. R. Fasolino (2006). "Testing Web-based applications: The state of the art and future trends." Information and Software Technology 48(12): 1172-1186.
Dingsøyr, T., T. Dybå, et al. (2008). A Preliminary Roadmap for Empirical Research on Agile Software Development. Proceedings of the Agile 2008, IEEE Computer Society: 83-94.
DiStefano, C., M. Zhu, et al. (2009). "Understanding and using factor scores: Considerations for the applied researcher." Practical Assessment, Research & Evaluation 14(20): 1-11.
Doshi, C. and D. Doshi (2009). A Peek into an Agile Infected Culture. Agile Conference, 2009. AGILE '09.
DSB (1999). Final Report of the Defense Science Board Task Force on Globalization and Security.
Eisner, E. W. (1991). The enlightened eye: qualitative inquiry and the enhancement of educational practice. New York, Macmillan.
Erdogan, G. and E. T. Baadshaug (2008). Extending SeaMonster to Support Vulnerability Inspection Modeling, NTNU, Department of Computer and Information Science.
Erdogan, G., P. H. Meland, et al. (2010). Security Testing in Agile Web Application Development - A Case Study Using the EAST Methodology. Agile Processes in Software Engineering and Extreme Programming. A. Sillitti, A. Martin, X. Wang and E. Whitworth, Springer Berlin Heidelberg. 48: 14-27.
Fenton, N. E. and S. L. Pfleeger (1997). Software metrics: a rigorous and practical approach, International Thomson Computer Press.
Field, A. (2005). Discovering Statistics Using SPSS, SAGE Publications.
Field, A. (2009). Discovering Statistics Using SPSS, SAGE Publications.
Garson, D. (2012). Testing Statistical Assumptions, Statistical Associates Publishing.
Ge, X., R. F. Paige, et al. (2007). Extreme programming security practices. Proceedings of the 8th international conference on Agile processes in software engineering and extreme programming. Como, Italy, Springer-Verlag: 226-230.
Ge, X., R. F. Paige, et al. (2006). Agile development of secure web applications. Proceedings of the 6th international conference on Web engineering. Palo Alto, California, USA, ACM: 305-312.
Gibbs, G. R. (2007). Analyzing qualitative data. London, Sage.
Goertzel, K. M., T. Winograd, et al. (2007). Software Security Assurance, State-of-the-Art Report (SOAR). D. o. D. IATAC and DACS, United States Government.
Goodin, D. (2010). "Microsoft security dev tools go 'Agile'." Retrieved 25/02/2010, from http://www.theregister.co.uk/2010/02/02/microsoft_sdl_expands/.
Gorsuch, R. L. (2013). Factor Analysis, Taylor & Francis.
Green, P. E., D. S. Tull, et al. (1988). Research for marketing decisions, Prentice Hall.
Greene, J. C. and V. J. Caracelli (1997). Advances in mixed-method evaluation: the challenges and benefits of integrating diverse paradigms. San Francisco, CA, Jossey-Bass Publishers.
Gregoire, J., K. Buyens, et al. (2007). On the Secure Software Development Process: CLASP and SDL Compared. Proceedings of the Third International Workshop on Software Engineering for Secure Systems, IEEE Computer Society: 1.
Guadagnoli, E. and W. F. Velicer (1988). "Relation to sample size to the stability of component patterns." Psychological Bulletin 103(2): 265-275.
Hair, J. F., B. Black, et al. (2006). Multivariate Data Analysis. Upper Saddle River, N.J., Pearson Prentice Hall.
Hatch, J. A. (2002). Doing qualitative research in education settings, State University of New York Press.
Hearn, J. (2004). "Does the Common Criteria Paradigm Have a Future?" IEEE Security and Privacy 2(1): 64-65.

Heckman, R. (2005). "Is Agile development secure?" Retrieved 5/5/2010, from http://www.techrepublic.com/article/is-agile-development-secure/.
Hieatt, E. and R. Mee (2002). "Going faster: testing the Web application." Software, IEEE 19(2): 60-65.
Holsti, O. R. (1968). Content analysis. Handbook of social psychology. G. Lindsey and E. Aronson. Reading, MA, Addison-Wesley. 2: 596-692.
Hubley, A. M. and B. D. Zumbo (1996). "A Dialectic on Validity: Where We Have Been and Where We Are Going." Journal of General Psychology 123: 207-215.
Huo, M., J. Verner, et al. (2004). Software quality and agile methods. Computer Software and Applications Conference, 2004. COMPSAC 2004. Proceedings of the 28th Annual International.
Huynh, H. and L. S. Feldt (1976). "Estimation of the box correction for degrees of freedom from sample data in randomized block and split-plot designs." Journal of Educational and Behavioral Statistics 1(1): 69-82.
Jick, T. D. (1979). "Mixing Qualitative and Quantitative Methods: Triangulation in Action." Administrative Science Quarterly 24(4): 602-611.
Jolliffe, I. T. (1972). "Discarding Variables in a Principal Component Analysis. I: Artificial Data." Journal of the Royal Statistical Society. Series C (Applied Statistics) 21(2): 160-173.
Jolliffe, I. T. (1986). Principal Component Analysis. New York, Springer.
Kaiser, H. (1970). "A second generation little jiffy." Psychometrika 35(4): 401-415.
Kaiser, H. (1974). "An index of factorial simplicity." Psychometrika 39(1): 31-36.
Kaiser, H. F. (1960). "The application of electronic computers to factor analysis." Educational and Psychological Measurement 20: 141-151.
Kaplan, B. and D. Duchon (1988). "Combining Qualitative and Quantitative Methods in Information Systems Research: A Case Study." MIS Quarterly 12(4): 571-586.
Keith, C. (2010). Agile Game Development with Scrum, Addison Wesley.
Keramati, H. and S.-H. Mirian-Hosseinabadi (2008). Integrating software development security activities with agile methodologies. ACS/IEEE International Conference on Computer Systems and Applications.
Kirk, J. and M. L. Miller (1986). Reliability and validity in qualitative research. Beverly Hills, CA, Sage.
Kongsli, V. (2006). Towards agile security in web applications. Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications. Portland, Oregon, USA, ACM: 805-808.
Kongsli, V. (2007). Security testing with Selenium. Companion to the 22nd ACM SIGPLAN conference on Object-oriented programming systems and applications companion. Montreal, Quebec, Canada, ACM: 862-863.
Kotulic, A. G. and J. G. Clark (2004). "Why there aren't more information security research studies." Information & Management 41(5): 597-607.
Lane, A. (2010). "FireStarter: Agile Development and Security." Retrieved 22/02/2010, from https://securosis.com/blog/agile-development-and-security.
Lather, P. and P. A. Lather (1991). Getting smart: feminist research and pedagogy with/in the postmodern, Routledge.
LeCompte, M. D. and J. J. Schensul (1999). Designing and conducting ethnographic research, AltaMira Press.
Li, H., R. Rosenthal, et al. (1996). "Reliability of Measurement in Psychology: From Spearman-Brown to Maximal Reliability." Psychological Methods 1(1): 98-107.
Likert, R. (1932). A technique for the measurement of attitudes, Archives of psychology.
Lincoln, Y. S. and E. G. Guba (1985). Naturalistic inquiry. Beverly Hills, CA, Sage Publications.
Lu, C.-S., K.-h. Lai, et al. (2007). "Application of structural equation modeling to evaluate the intention of shippers to use Internet services in liner shipping." European Journal of Operational Research 180(2): 845-867.
Marshall, C. and G. B. Rossman (2006). Designing qualitative research, Sage Publications.
McClelland, D. C. (1961). The Achieving Society. Princeton, NJ, D. Van Mostrand.
McClelland, D. C. (1985). Human motivation. Glenview, IL, Scott, Foresman.
McGraw, G. (2006). Software security: building security in, Addison-Wesley.

Meneely, A. and L. Williams (2010). Strengthening the empirical analysis of the relationship between Linus' Law and software security. Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, ACM.
Miles, M. B. and A. M. Huberman (1994). Qualitative data analysis: a sourcebook of new methods. Thousand Oaks, CA, Sage.
Muthén, B. and D. Kaplan (1985). "A comparison of some methodologies for the factor analysis of non-normal Likert variables." British Journal of Mathematical and Statistical Psychology 38: 171-189.
Nesbary, D. (2000). Survey research and the World Wide Web, Allyn and Bacon.
Norman, G. (2010). "Likert scales, levels of measurement and the "laws" of statistics." Advances in Health Sciences Education 15(5): 625-632.
Nunnally, J. C. and I. H. Bernstein (1994). Psychometric theory, McGraw-Hill.
Oppenheim, A. N. (2001). Questionnaire Design, Interviewing and Attitude Measurement, Bloomsbury.
OWASP. (2008). "Security In Agile Development." Retrieved April 20, 2011, from http://video.google.co.uk/videoplay?docid=-8287209466278543377&ei=USqdS_HYLIyC-AappqiZDQ&q=security+in+agile&hl=en#.
OWASP. (2010). "OWASP Top Ten 2010." Retrieved April 20, 2011, from http://vimeo.com/9006276.
OWASP. (2011, 2/2/2009). "The OWASP Testing Framework." Retrieved 1/5/2011, from https://www.owasp.org/index.php/The_OWASP_Testing_Framework.
Paige, R., J. Cakic, et al. (2004). Towards agile re-engineering of dependable grid applications. Proceeding of 17th International Conference of Software and System Engineering and Their Application (ICSSEA), CNAM. Paris. vol.1 à 3: 1.1-1.7.
Pallant, J. (2007). SPSS Survival Manual: A Step by Step Guide to Data Analysis Using SPSS, Allen & Unwin.
Peterson, R. A. (1994). "A Meta-Analysis of Cronbach's Coefficient Alpha." Journal of Consumer Research 21(2): 381-391.
Pfleeger, C. P. and S. L. Pfleeger (2007). Security in computing, Prentice Hall.
Poppendieck, M. and R. Morsicato (2002). "Using XP for Safety-Critical Software." Cutter IT Journal 15(9): 12-16.
Rico, D. F. (2008). Effects of Agile Methods on Website Quality for Electronic Commerce. Hawaii International Conference on System Sciences, Proceedings of the 41st Annual.
Rosenthal, R. and R. L. Rosnow (1991). Essentials of behavioral research: methods and data analysis. New York, McGraw-Hill.
Rossman, G. B. and B. L. Wilson (1985). "Numbers and Words: Combining Quantitative and Qualitative Methods in a Single Large-Scale Evaluation Study." Evaluation Review 9(5): 627-643.
Rostaher, M. and M. Hericko (2002). Tracking Test First Pair Programming—An Experiment, Springer.
Ryan Jr, J. and R. Scudiere (2008). The Price of Agile Is Eternal Vigilance. Proceedings of the Agile 2008, IEEE Computer Society: 125-128.
Salkind, N. (1990). Exploring research. New York, MacMillan.
Sars, C. (2010). "Agile Security?", 2010.
Saunders, M., P. Lewis, et al. (2007). Research Methods for Business Students, Pearson Education.
Schwandt, T. A. (2000). Three epistemological stances for qualitative inquiry. Handbook of qualitative research. N. K. Denzin and Y. S. Linclon, Thousands Oaks: 189-213.
Sfetsos, P., I. Stamelos, et al. (2006). Investigating the Impact of Personality Types on Communication and Collaboration-Viability in Pair Programming – An Empirical Study. Extreme Programming and Agile Processes in Software Engineering. P. Abrahamsson, M. Marchesi and G. Succi, Springer Berlin / Heidelberg. 4044: 43-52.
Sindre, G. and A. Opdahl (2001). Templates for Misuse Case Description. 7th International Workshop on Requirements Engineering, Interlaken, Switzerland.

Singh, S. (2009). "Agile Security Threats." Retrieved 29/04/2010, from http://singhsanjeev.com/2009/06/agile-security-threat.html.
Siponen, M. (2001). An Analysis of the Recent IS Security Development Approaches: Descriptive and Prescriptive Implications. Information Security Managment - Global Challenges in the Next Millennium, Idea Group.
Siponen, M., R. Baskerville, et al. (2005). Integrating Security into Agile Development Methods. Proceedings of the 38th Annual Hawaii International Conference on System Sciences, 2005. HICSS '05. .
Sjoberg, D. I. K., B. Anda, et al. (2002). Conducting realistic experiments in software engineering. Empirical Software Engineering, 2002. Proceedings. 2002 International Symposium n.
Smith, C. P. (1992). Reliability issues. Motivation and personality: handbook of thematic content analysis. C. P. Smith, J. W. Atkinson, D. C. McClelland and J. Veroff. New York, Cambridge University Press.
Standish (1995). The Chaos Report. West Yarmouth, MA, The Standish Group.
Stevens, J. P. (2002). Applied Multivariate Statistics for the Social Sciences. Hillsdale, NJ, L. Erlbaum Associates Inc.
Straub, D. W. and R. J. Welke (1998). "Coping with Systems Risk: Security Planning Models for Management Decision Making." MIS Quarterly 22(4): 441-469.
Strauss, A. L. and J. M. Corbin (1990). Basics of qualitative research: grounded theory procedures and techniques. Newbury Park, CA, Sage Publications.
Strauss, A. L. and J. M. Corbin (1998). Basics of qualitative research: grounded theory procedures and techniques. Thousand Oaks, CA, Sage Publications.
Sue, V. M. and L. A. Ritter (2007). Conducting online surveys, Sage Publications.
Tabachnick, B. G. and L. S. Fidell (2007). Using Multivariate Statistics, Pearson/Allyn & Bacon.
Tashakkori, A. and C. Teddlie (1998). Mixed methodology: combining qualitative and quantitative approaches, Sage.
Tashakkori, A. and C. Teddlie (2003). Handbook of mixed methods in social & behavioral research, SAGE Publications.
Tesch, R. (1990). Qualitative research: analysis types and software tools, Falmer Press.
Theunissen, W. H. M., D. G. Kourie, et al. (2003). Standards and agile software development. Proceedings of the 2003 annual research conference of the South African institute of computer scientists and information technologists on Enablement through technology, South African Institute for Computer Scientists and Information Technologists: 178-188.
Vaha-sipila, A. (2010). "Software Security Management for Large Agile Projects." Retrieved 29/04/2010.
Viega, J. and G. McGraw (2002). Building Secure Software: How to Avoid Security Problems the Right Way. Boston, MA, Addison-Wesley Longman Publishing Co., Inc.,.
Wäyrynen, J., M. Bodén, et al. (2004). Security Engineering and eXtreme Programming: An Impossible Marriage? Extreme Programming and Agile Methods - XP/Agile Universe 2004. C. Zannier, H. Erdogmus and L. Lindstrom, Springer Berlin / Heidelberg. 3134: 152-195.
West, D. and T. Grant (2010). Agile Development: Mainstream Adoption Has Changed Agility, Forrester Research, Inc.
Wichers, D. (2008). "Breaking the Waterfall Mindset of the Security Industry." Retrieved April 20, 2011, from www.owasp.org/images/b/b8/AppSecEU08-Agile_and_Secure.ppt.
Wichers, D. (2009). "OWASP Top 10 - 2010." Retrieved April 20, 2011, from http://www.owasp.org/images/a/a1/AppSec_DC_2009_-_OWASP_Top_10_-_2010_rc1.pptx.
Williams, L., R. R. Kessler, et al. (2000). "Strengthening the case for pair programming." Software, IEEE 17(4): 19-25.
Williams, L., A. Meneely, et al. (2010). "Protection Poker: The New Software Security "Game"." Security & Privacy, IEEE 8(3): 14-20.
Wolcott, H. F. (2001). Writing up qualitative research, Sage Publications.
Yin, R. K. (2003). Case study research: design and methods. Thousand Oaks, CA, Sage Publications.

Zyzanski, S. J., I. R. McWhinney, et al. (1992). Qualitative research: Perspectives on the future. Doing qualitative research: Research methods for primary care. B. F. Crabtree and W. L. Miller. Newbury Park, CA, Sage Publications. 3.

Appendixes

Appendix A: 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	

http://agilemanifesto.org/

Appendix B: Semi-Structured Interviews Questions

General	Specific
General Questions	• 1. What are your project or code level security practices that are in use today in your organization? • 2. Is security as important to agile projects as many would like to have us believe? • 3. Which specific practice in your project contributed the most to increased security of the resulting software? • 4. Which factor(s) are most responsible for causing vulnerabilities and weaknesses in software in terms of security? • 5. Is there an observed or apparent trend towards increased security of software in agile projects? • 6. How can security related aspects such as risk analysis be effectively made to become part of iterative and incremental processes? • 7. Does every project inevitably end up with some security vulnerabilities and what can be done to mitigate this risk? • 8. What is the best way to model Security issues as part of agile projects?
Traditional Security Mechanisms	• 1. Do you think traditional security mechanisms are adequate for inclusion into agile without any modification? • 2. Are you implementing a form of traditional security mechanism in your projects? If yes elaborate, if no why
Security in FDD	• 1. Does including risk assessment as a part of the feature design process be beneficial to the overall security of the software produced?
Security in XP	• 1. Would it be beneficial to go through a number of security iterations after all functional iterations are completed for a given release? Why or why not • 2. Would the use of refactoring play a positive role in implementing security requirements towards the end of the release? • 3. Would it be more beneficial to spread the security iterations throughout the project rather than deferring them until the last number of iterations before the release? • 4. Would solely relying on refactoring towards the end of the project enable better security assurance or handling security requirements throughout the project?
Misuse or Abuse Stories	• 1. Are Misuse or Abuser stories the best ways to describe and document security requirements? If yes why if no what could be a better replacement? • 2. Are Misuse or Abuser stories easily translatable to good security focused unit tests? Why or why not? • 3. Does misuse or abuser stories more considered as built-

	in security or bolted-on security? Please elaborate
Modeling security Issues	• 1. Are using attack trees the best way to model security issues as part of agile projects? If yes why, if not what are the alternatives? • 2. How do you model security issues and/or threats as part of your project(s)?
Resource Utilization	• 1. Do you think adding security requirements to agile projects would be time/cost prohibitive in some cases? • 2. Do you think the customer should have the ability to dictate security requirements or should it reside with the project managers?
Extreme Security Engineering	• 1. Is having the customer provide security requirements for the project considered to be "Good Enough Security" for the project? • 2. Would the addition of a number of security iterations consisting of one or more security requirements provide a good security assurance argument? • 3. To what degree should the customer be involved in creating and/or deciding on security requirements and/or features? • 4. Should customers be allowed to be in charge of creating functional test cases while security engineers develop unit tests when it comes to security?
Security Testing	• 1. Do you think that having two security engineers working as one on a given security problem produces more high-quality results compared to just one security engineer? What about in terms of cost? • 2. What form of security testing do you use on agile projects? Why or why not
Extending Agile Methodology	• 1. Do you think extending agile to include security specific techniques provides the necessary assurance that the resulting software would be secure against most threats and vulnerabilities? • 2. Is extending the agile process for the sake of security really necessary? If yes why, of not what are the alternatives? • Do you agree with the claim that "the addition of security should not hinder or lengthen the agile process"?
Security through Documentation	• 1. Does increasing documentation in agile provide increased security in your opinion? Why or why not • 2. Do you think the lack of formal documentation mechanisms that have been replaced in agile with more informal practices has translated to less security for the resulting software? • 3. Do you agree or disagree with the statement: "exploration of security issues must be done from the agile perspective rather than from a pure security perspective which is deeply biased towards documentation and early planning to accomplish security"

	If you agree why, if not why not? • 4. Are the CASE tools mature enough to be able to extract many forms of documentation from the agile projects that in the traditional projects used to be created as part of the design phase of the project? If yes, does this contribute to agile's increased security assurance argument?
Security Standards: SSE-CMM or CC	• 1. Do you use any open security standards and/or initiatives such as CVE, OVAL, CCE, CPE, CWE, CAPEC, CVSS, CRF in your projects? • 2. Do you think the traditional security models such as SSE-CMM and CC are too slow and ill-suited to be used in conjunction with agile? What alternative(s) can you suggest? • 3. What part of CC do you think would be best suited to include into Agile in an un-obtrusive manner?
Security Assurance Argument	• 1. What is the best practice that could lead to the "Security Assurance" argument if implemented as part of the agile project? • 2. In your professional opinion, do you agree or disagree with the statement that "accelerated schedules of agile hinder security"? Why or why not? • 3. Do you think that having automatic testing and pair programming (especially if one of the team members is a dedicated security engineer or expert) could be used to test for security specific cases to provide security assurance argument for the project and satisfy the overall security requirement? • 4. Would you agree that having a security engineer (which could help spread the tacit knowledge of security) on-board in agile projects would have the same or similar effect as having a thorough documentation which is aimed at providing security assurance?
Security or Agile Perspective	• 1. Do you agree with the statement that in order to keep the agile benefits intact we must strictly safeguard the process and instead introduce activities that fit within the framework of agile specifically dedicated to security in order to bolster the project to provide better security assurance argument?
Planning for security	• 1. Would it be advantageous to try to establish collective ownership over security matters through extending the planning meetings to include time for security? Why or why not
Formalizing security requirements	• 1. Would be advantageous to formalize security requirements and create security test cases for them to establish a formal security testing mechanism? How would this affect the overall security assurance argument? • 2. Do you agree that security issues need to be prioritized in terms of threat level and risk and voted on by stakeholders to implement or not in order to keep costs to

	a reasonable level and to establish a more formal assurance procedure for establishing security?
Verification tools	• 1. Do you know of any security verification tool(s) that aid in finding vulnerabilities and/or security threats that can be used for agile projects? • 2. Do you think the testing and verification tools used by agile teams provide adequate security assurance for the software in addition to quality assurance? • 3. Do you think that having an automated set of tools to better assist with finding security issues would be beneficial to overall security of the software?
Security and quality assurance	• 1. What is the best way to increase the security assurance argument in agile? • 2. Which is more beneficial to overall security assurance and why: -The addition of a security engineer to the team -The extension of agile with specific security related tasks to be included in every iteration -Dedication of one or more security iteration(s) for every x number of normal functional iterations
Agile principles on Security	• 1. Which of the agile manifesto principles in your opinion impacts security negatively if any?
New Ideas for better security	• 1. Would the addition of a security engineer be beneficial to the team? • 2. Would the extension of agile with specific security related tasks to be included in every iteration useful? • 3. Would the dedicated security iteration(s) for every x number of normal functional iterations plausible in your professional opinion?
Security Metrics	• 1. What Metric(s) if any are you capturing and/or using to monitor security practices for the project? • 2. Do you use any tools and/or techniques to access, analyze, and document vulnerabilities that are discovered throughout the duration of the project? If yes, elaborate…. • 3. Can the measure of the number of defects and vulnerabilities found be the basis for measuring relative security of the software?
Education and Awareness	• 1. How important is Education and Awareness of security issues by developers to your projects?

Table B-1: Semi-Structured Interview Questions

Appendix C: Detailed Themes from Semi-Structured Interviews

RQ1 Combining Security & Agility

Theme RQ1A

Label Dedicated Security Engineer

Definition The addition of a highly experienced Developer and/or Software Engineer/Architect to the team to act as the Formal, Dedicated Security Engineer for the project.

Indicators/Flags The mention of the existence or desirability of the existence of a Security Engineer within a sentence.

Examples "I think security specialist would be useful. For developers, there is a certain level of knowledge on how to write secure code." (Interview_02)

Exclusions Informal security expert not coded for this theme.

Theme RQ1B

Label Software with Security in mind

Definition The claim that consideration of security and having security specific knowledge will help in creating more secure software.

Indicators/Flags Any mention of the perceived and/or direct benefit that consideration of security would bring to the team and/or project within a sentence.

Examples "So writing code in a responsible fashion and they not leaving code in a way they could be exploited. Therefore, they should develop with security in mind and think about how is going to be used." (Interview_11)

Exclusions Explicit steps to increase awareness of security are excluded from coding.

Theme RQ1C

Label Security Controls

Definition The use of Standard or 3rd party Security Specific controls within software as an alternative to provide added security to the resulting software.

Indicators/Flags The mention of existing or proposed 3rd party security controls as part of the software development within a sentence.

Examples "We should have standard control that allows us to have similar results from any software developer." (Interview_15)

Exclusions Traditional security mechanisms not coded.

Theme RQ1D

Label Informal Security Experts

Definition Sometimes, if the team lacks the formal security expert as a source for guidance, they turn to the most experienced person in the team with known or implied security experience.

Indicators/Flags The mentioned or implied existence of a person informally in charge of security or answering security questions and/or concerns of the team within a sentence.

Examples "In our project, there is someone who is known as expert on security and there are some other people who are expert on different aspect." (Interview_01)

Exclusions The formal security engineer is not included

Theme RQ1E

Label Integration Risk

Definition The notion that adding or attempting to add security practices and/or procedures to Agile projects will affect Agile in undesirable ways.

Indicators/Flags When a proposed addition and/or integration mechanism or process change will result in added risk to the project.

Examples "I'm sure adding it [security requirements] to any project could be time or cost prohibitive." (Interview_09)

Exclusions Proposals that do not impact the agile process are not included in the coding.

Theme RQ1F

Label Experience of Developers

Definition The idea that the more knowledgeable and experienced the developer, the more the resulting software will be secure.

Indicators/Flags The mentioned or implied existence of more experienced developers within the team that help in identifying security issues and/or concerns within a sentence.

Examples "If you hire intelligent, best developer they will be more than 10 times more productive and more than 10 times more secure than the average developer" (Interview_01)

Exclusions Does not include QA or other roles such a manager

Theme RQ1G

Label Security in planning

Definition Dedication of some time to discuss security issues with the software team during planning stages as part of the process.

Indicators/Flags Any mention of the inclusion of issues about security during planning or the beginning stages of the project and/or iteration within a sentence.

Examples "Again once you got that figured out that higher level things, then you need to do essentially risk analysis on every single user story and again this is not super complicated. But you need to think about security for every user story."(Interview_14)

Exclusions Security in later stages not included.

Theme RQ1H

Label Static Analysis Tools

Definition Tools and/or software programs that aid in finding various issues in terms of security of the resulting software.

Indicators/Flags Any mention or implied use and/or apparent benefit of static analysis tools in providing added security assurance within a sentence.

Examples "Static analysis tools can scan the code and generate a report to tell you where is what member [variable residing in memory] is leaking and your code should be changed to avoid some security hole." (Interview_01)

Exclusions Other analysis tools such as dynamic analysis tools and other tools not coded.

Theme RQ1I

Label Static Code Reviews

Definition Techniques and/or practices that allows developers to find weaknesses and vulnerabilities within the code base.

Indicators/Flags Any mention of the apparent or perceived benefit of conducting static code reviews on the resulting security of the software being written as part of a sentence.

Examples "Our senior engineers review our code whenever we check-in code into the repository and this is very critical. Code review and inspection will be helpful on security of the code. Senior developers can see what we cannot see." (Interview_01)

Exclusions QA not included in the coding.

RQ2 Change Agile Practices for Security

Theme RQ2A

Label Agile vs. Traditional Methods

Definition The opinion that Agile either in part or as a whole produces software that is different in terms of security than more traditional upfront-designed and developed software development mechanisms.

Indicators/Flags Any mention of a comparison and/or contrast between the two methodologies in terms of security within a sentence.

Examples "In terms of security, I don't think there is any difference between waterfall and agile to be honest. I am not seeing a lot of thought outside of the regulated industry where it is legal requirement to do it."(Interview_11)

Exclusions Other comparisons not related to security not coded.

Theme RQ2B

Label	Awareness of Security to Agile
Definition	The concern that the stakeholders of the project may not recognize the importance and may not even be aware of the potential impact security has on the software.
Indicators/Flags	Any mention of the apparent and/or perceived impact of awareness of security on the resulting software under development within a sentence.
Examples	"Developing team might not be aware of the security threats or risk to the project." (Interview_08)
Exclusions	Awareness of other issues not coded.

Theme RQ2C

Label	Impact of Accelerated Schedules
Definition	The notion that doing things faster and being given a shorter timeframe to get tasks done adds an element of risk to various stages of software development lifecycle in terms of security.
Indicators/Flags	Any mention of the potential and/or consequential impact of the accelerated pace of development on the team and the resulting software in terms of security within a sentence.
Examples	"If you are in hurry then you will have more chances to write a less secure code. This is common in all projects." (Interview_01)
Exclusions	Any level higher than project level is not coded.

Theme RQ2D

Label Reduced Security for Internal Projects

Definition The recognition that software that is going to be only available internally within a given organization can have less security requirements.

Indicators/Flags Any mention of the state of practice within an organization in terms of various levels of security specifically project level and system level security within a sentence.

Examples "This is an internal application and as part of the infrastructure it will protect it from the outside world and therefore we don't have to do risk analysis." (Interview_3)

Exclusions Projects where the software is to be deployed outside of the organization is not coded.

Theme RQ2E

Label No change Necessary

Definition The opinion that Agile is better off not being changed in favor of security by the addition of certain security specific practices and/or processes.

Indicators/Flags Any mention of the opinion and/or experience that states or implies the changes to the agile process for the sake of security are not really necessary within a sentence.

Examples "You don't extend the agile process to say ok it need to be concurrent. You shouldn't extend the process to say it is secure however, things you do in order to make sure that you don't have concurrency bugs the same way there should be things to do to make sure they are secure."(Interview_9)

Exclusions Changes to Agile not related to security not coded.

RQ3 Security Issues and Software Vulnerabilities

Theme RQ3A

Label Security through resolving vulnerabilities

Definition The recognition that in order to achieve better security of the resulting software all vulnerabilities must be taken care of before release.

Indicators/Flags Any mention of the relationship between vulnerabilities and the level of security of the software under question within a sentence.

Examples "To avoid vulnerabilities if necessary is assess the risk. What do you want to achieve in the system from the security point of view. What are your security needs? Once you know that you can take action? If you don't know that then of course you cannot succeed." (Interview_12)

Theme RQ3B

Label Relationship between defects and Security

Definition Based on the notion that there is a relationship between software bugs, defects, and vulnerabilities and the resulting security of the software.

Indicators/Flags Any mention of the cases where bugs and/or vulnerabilities are associated with security of the resulting software within a sentence.

Examples "For high severity bugs such as security issues and stability issues, we have to put it in highest priority in waiting list." (Interview_01)

Exclusions Bugs not classified as security related not coded.

Theme RQ3C

Label Impact of Experienced Developers

Definition The point that having more experienced and knowledgeable developers on the team will contribute to the resulting security of the software.

Indicators/Flags Any mention of the impact of a more experienced developer in helping to find security related issues and/or solutions within a sentence.

Examples "If you are experienced developer, even thought the requirement doesn't say it, if you don't think about that there will be a problem later." (Interview_04)

Exclusions The impact of low level developers on security not coded.

Theme RQ3D

Label The role of security training

Definition The claim that the knowledge of security contributes to creating a more secure software compared to the case where there is no specific security training for the team members.

Indicators/Flags Any mention of the relative importance and/or benefit of education would contribute to the overall security assurance of the software within a sentence.

Examples "Developer education is probably number one." (Interview_09)

Exclusions Awareness not coded for this theme.

RQ4 Customer's Control of Security

Theme RQ4A

Label Customer must Dictate all requirements

Definition The opinion that the customer should be responsible for and provide all needed requirements even though they might not be in the best position to understand certain issues.

Indicators/Flags Any mention of the opinion that the customer should make decisions about creating security requirements for the project within a sentence.

Examples "Security requirements should come from the customer, generally, because it's their business at the end of the day." (Interview_09)

Exclusions Code level security requirements not coded.

Theme RQ4B

Label Customer needs help with security

Definition The notion that the customer however knowledgeable they may still need more technical help in understanding security issues in order to understand the importance of security to the project.

Indicators/Flags Any mention of the apparent need of the customer for added council and advice on security specific issues with their project by relevant stakeholders within a sentence.

Examples "The development team has professional responsibility to include that the customer are not be able to explicitly express the need they will know that it is inadequate if they don't get it and there is responsibility on the team whether it's project manager, architect or other team member to include a appropriate requirements or at least to advocate for them to the customer and explain the economic value of including them." (Interview_13)

RQ5 Knowledge Dissemination and Diffusion

Theme RQ5A

Label Developers lack of Security knowledge

Definition The recognition that developers are not trained on security specific issues and may not be in the best position to make security related decisions.

Indicators/Flags Any mention of lack of developer's security knowledge and expertise or having no knowledge of security within a sentence.

Examples "Some developers have lack of knowledge regarding security issues and sometimes they might have conflict of interest in the project and therefore depending just on them wouldn't be a good idea." (Interview_15)

Exclusions Other team members lack of security knowledge such as managers and QA not coded.

Theme RQ5B

Label Tacit knowledge Dissemination and Sharing

Definition The notion that as it relates to agile the most effective form of informal training and making various stakeholders aware of security issues is through tacit knowledge sharing and dissemination.

Indicators/Flags Any mention of the effectiveness of knowledge sharing within the team on security and related issues within a sentence.

Examples "When I work in agile team I liked to pair program on everything we do and that is an example of being able to spread the knowledge and being able to get I think I can teach other people things and they can teach me things." (Interview_07)

Exclusions Formal training not coded.

Theme RQ5C

Label Security through Training

Definition The obvious choice that in order to learn security issues one must be specifically trained for it.

Indicators/Flags Any mention of the training as a solution to make developers aware of security or the need for the developers to be educated within a sentence.

Examples "As far as developers are concerned, anybody who can program can learn just about anything he needs about the software and if they don't certainly have that knowledge [of security], then they need to gain it." (Interview_06)

RQ6 Dedicated Security Iteration

Theme RQ6A

Label Periodic Security

Definition The idea that for every few normal functional iterations there should be one or more steps focused on security issues taken as part of the iteration.

Indicators/Flags Any mention of an iteration focused on certain security related practices within a sentence.

Examples "You could dedicate any iteration of the software to concentrating on one particular feature of the application. So you could have something like SQL performance iteration or any iteration base another around other things just like refactoring or whatever. As security iteration, yes sure." (Interview_7)

Exclusions Security steps that need to be a permanent part of Agile not coded.

Theme RQ6B

Label More public needs more Security

Definition The recognition that the more exposed the software is going to be to the public the more uncertainty in terms of security it would face and therefore more attention should be placed on the security aspect.

Indicators/Flags Any mention of the need for increased security when the deployment is in a more public environment within a sentence.

Examples "Whether you need to have code security or not, it's more to do whether the system is exposed publically whether it could be hacked" (Interview_13)

Exclusions Reduced security for internal project not coded.

RQ7 Testing and Verification for Security

Theme RQ7A

Label Refactoring

Definition The idea that refactoring helps with security in general.

Indicators/Flags Any mention of the use of refactoring as a good practice towards increasing security within a sentence.

Examples "Yes it would simplify you code base. So simple code base is generally easily to secure and easily to understand and refactor is tend to reduce what you have got. Refactoring should be done continuously." (Interview_11)

Theme RQ7B

Label Testing for security

Definition The notion that QA in addition to developers should be conducting unit testing and other security focused tests to increase the security assurance.

Indicators/Flags Any mention of the use of testing in order to prove the relative security of the resulting software within a sentence.

Examples "You want to demonstrate that all the test that they have provided run. All security tests that they are provided or has developed for the project you have to prove that they run and they all pass. That's all what you need to do prove it." (Interview_06)

Exclusions Testing for quality is not coded.

Theme RQ7C

Label Tools provide added Security Assurance

Definition The recognition that tools in general add to the level of assurance for a given purpose such as for security focused testing and verification, and other security aspects.

Indicators/Flags Any mention of the fact that using tools to increase security of the resulting software would be beneficial to the project within a sentence.

Examples "Tools are good, someone scanned the code weekly for vulnerabilities and we also need automated tools to help us with code security." (Interview_01)

Exclusions Non-security related tools not coded.

RQ8 Documentation and Security

Theme RQ8A

Label Documenting the Agile way

Definition The idea that the traditional forms of documentation used in waterfall and other traditional methods are no longer effective and could be replaced by more modern tools and/or practices.

Indicators/Flags Any mention of the way agile does documentation or practices that are closely related to it within a sentence.

Examples "In agile project, actually, everybody is still learning. But, in the very first agile releases, we attempted to discard most of the documentation. And finally we found we need some documentation, and technically we were thinking about what kind of documentation we really need? and essential for each iteration. Release by release, we have some documentation defined as essential. For example, the design documentation and we always do some light documentation. I called light because we can modify and edit it and revise the content at any time I want." (Interview_01)

Theme RQ8B

Label Agile not less secure than traditional methods

Definition The idea that just because agile produces less documentation that it somehow results in less secure software produced.

Indicators/Flags Any mention of the use of documentation in agile compared to traditional approaches and vice versa within a sentence.

Examples "We don't have code design documentation and the reason for that in my experience any such documentation is wrong immediately." (Interview_09)

Exclusions User level documentation not coded.

Theme RQ8C

Label Abuse or Misuse Stories

Definition The idea that the use of security focused user stories such as misuse or abuser stories would increase the security assurance argument.

Indicators/Flags Any mention of the perceived or apparent benefit of using misuse or abuser stories within a sentence.

Examples "You find the issue through the misuse or abuse stories and then write a test case for that." (Interview_04)

Theme RQ8D

Label Reducing Documentation Risky in Agile

Definition The recognition that while reducing documentation is desired, too much reduction of documentation could add more risk to the project.

Indicators/Flags Any mention of the risk associated with reduced documentation in agile within a sentence.

Examples "If you trying just to deliver results and the project doesn't have any concern about security too much, then you can say I don't want to do too much about documentation. If I'm going to develop online banking system then definitely the security is very important. So I don't think that this kind of product is quite simple for agile in this case." (Interview_04)

Exclusions Other effects of reduced documentation not coded.

Theme RQ8E

Label Testing over Documentation

Definition The idea that forgoing writing of unnecessary documentation could be replaced with more effective practices such as more Testing.

Indicators/Flags Any mention of the ways in which testing could be used to replace the need for more documentation in Agile within a sentence.

Examples "Those documents could in a way have security flaws they could be incorrect or they could be talking about things that those don't actually happen or don't actually work where in agile software, might be certain acceptance tests that have been written or all the acceptance test that have been written out all the executable specification of the software." (Interview_07)

Exclusions Other replacements for documentation not coded.

Theme RQ8F

Label Simpler code better than more documentation

Definition The idea that writing simpler, self documenting code is preferred over documentation that is hard to maintain and update.

Indicators/Flags Any mention of the apparent benefit of simpler coding over documentation within a sentence.

Examples "I would think that by writing readable code which is clean and simple code will make it tested as well in which they are basis of the agile practices will limit the amount of bugs that might be exploited." (Interview_03)

Appendix D: Survey Questions

UNIVERSITY OF Southampton
School of Electronics and Computer Science

Dear Prospective Participant,
I am Ahmed Alnatheer, a PhD candidate at University of Southampton, working toward a doctorate degree in Computer Science. You are invited to participate in a questionnaire which focuses on *Security Adoption in Agile Methodologies*. To take this questionnaire, please read the following:

Title
An Investigation into Security Issues in Agile Methodologies

Purpose

1. Capture metrics from the agile team members regarding security issues.
2. Measure the degrees of consensus practitioners have towards the major security issues in Agile.

Procedure
To complete this questionnaire, you need to answer some background questions and statements about your perception and attitude towards some security issues. This questionnaire will take approximately 10-15 minutes.

Potential benefits
Your participation will help in finding the best ways to integrate security practices with Agile methods. In addition, it will help in addressing some of the concerns facing practitioners in software development field today.

Confidentiality
No personal information identifying you will be asked or recorded. The results of this survey will only be used for the study. You can also withdraw from the survey at any time.

Consent of Research Participant
You are considered to be giving your explicit consent for participation which is obtained by your participation in this questionnaire.

Your participation in this questionnaire is deeply appreciated.

Yours sincerely,
Ahmed Alnatheer
PhD Candidate
School of Electronics and Computer Science
University of Southampton, UK
Phone: +44 (0)2380-581965
aaa1v09@ecs.soton.ac.uk

Section 1: Background Questions

Q1: Where are you located?

- ☐ Europe
- ☐ North America
- ☐ South America
- ☐ Asia
- ☐ Africa
- ☐ Australia

Q2: Type of Organization (please choose the one that most closely applies)

- ☐ Information Technology, Security, Telecommunications
- ☐ Finance, Banking, Insurance
- ☐ Government
- ☐ Consulting/Professional Services
- ☐ Manufacturing/Retail
- ☐ Education, Training
- ☐ Other (please specify)

_____

Q3: What is the total number of employees in your organization?

- ☐ Less than 300
- ☐ Between 301 – 2,000
- ☐ Between 2,001 – 5,000
- ☐ Between 5,001 – 20,000
- ☐ More than 20,001

Q4: What is your level of education?

- ☐ Less than Bachelor's Degree
- ☐ Bachelor's Degree
- ☐ Master's Degree
- ☐ Ph.D

Q5: What is your role within the organization?

- ☐ Programmer
- ☐ Software Engineer/Architect
- ☐ Systems Engineer/Analyst
- ☐ Tester/Quality Assurance
- ☐ Senior Manager/Project Manager
- ☐ Consultant
- ☐ Security Engineer/ Security Auditor/etc.
- ☐ Customer/Stakeholder/Owner/Partner
- ☐ Other (please specify)

_____

Q6: How many years of overall experience do you have?

- ☐ Less than 1 Year
- ☐ 2-3 Years
- ☐ 4-6 Years
- ☐ 6-10 Years
- ☐ 10+ Years

Q7: How many years of experience do you have in your current role?

- ☐ Less than 1 Year
- ☐ 2-3 Years
- ☐ 4-6 Years
- ☐ 6-10 Years
- ☐ 10+ Years

Q8: Which Agile methodology was most recently used in your organization? (pick the one that most closely applies)

- ☐ Scrum
- ☐ eXtreme Programming (XP)
- ☐ Test Driven Development (TDD)
- ☐ Feature Driven Development (FDD)
- ☐ Other (please specify)

_____

Q9: What was the team size?

- ☐ 1-5
- ☐ 6-10
- ☐ 11-20
- ☐ 21-50
- ☐ 51-100
- ☐ 101+ people

Q10: How many years of experience do you have working with the current methodology?

- ☐ Less than 1 Year
- ☐ 2-3 Years
- ☐ 4-6 Years
- ☐ 6-10 Years
- ☐ 10+ Years

Q11: If you used an Iterative methodology, how many iterations did it take to produce a major release? (please pick the closest answer)

- ☐ 1 Iteration
- ☐ 2-3 Iterations
- ☐ 4-5 Iterations
- ☐ 6-10 Iterations
- ☐ 11+ Iterations

Q12: What is the length of one iteration? (if it varies choose the average)

- ☐ 1 Week or less
- ☐ 2 Weeks
- ☐ 3 Weeks
- ☐ 4-6 Weeks
- ☐ 7+ Weeks

Section 2: Combining Security & Agility	Strongly Agree	Agree	Somewhat Agree	Neutral	Somewhat Disagree	Disagree	Strongly Disagree
Having a Security Engineer within the agile team can...							
Increase each team member's security knowledge							
Help in producing more secure software							
Help solve security issues faster							
Increase awareness of security issues							
Uncover security issues earlier in the project							
Help the customer make better security decisions							
Lengthen the iteration							
Help Establish/Increase the overall security assurance of Agile							
Help the project achieve compliance with security standards faster							
Lower the security risk of the produced software further than developers alone can							
Decrease the need for highly experienced developers/architects within the team							
Alleviate other team members from having to focus on specific security issues							
Maximize the precision of security requirement assessment							
Put more importance on security for the project and the team							
Mitigate the lack of mature security vulnerability assessment tools							
Allow him/her to serve as an independent on-site security auditor for the project							
Building Software with Security in mind in agile projects can...							
Establish/Increase the Security Assurance for the project							
Help in writing more secure code							
Help in achieving security focused testing							
Expand the awareness of security throughout the project							
Put more overall focus on security							
Promote the collective ownership of security by all stakeholders							
Help in decreasing vulnerabilities in the software							
Reduce security risk to the project and the resulting software							
Using Software Security Controls (pre-made security modules) in the agile project can...							
Alleviate developers from having to focus on security							
Be beneficial throughout the project							
Be reused on multiple projects							
Become a part of the framework/infrastructure							
Be available as part of the infrastructure before the next project begins							
Reduce costs if used on multiple projects							
Help the resulting software to have reduced weaknesses/vulnerabilities							
Reduce reliance on individuals within the team for security expertise							
Experienced Developers within the agile team can...							
Create/write more secure user stories							
Discover more security issues/vulnerabilities/bugs than regular developers							
Help implement security requirements faster							
Add to the team's overall security awareness							
Mitigate more security issues than regular developers							
Be more effective than using tools on security							
Increase the overall security of the software							

Be more beneficial than QA when it comes to security focused testing							
Play a key role in avoiding to introduce security vulnerabilities into software							
Predict security vulnerabilities at earlier stages of the project							
Section 3: Change Agile Practices for Security	Strongly Agree	Agree	Somewhat Agree	Neutral	Somewhat Disagree	Disagree	Strongly Disagree
Compared to Traditional Waterfall, Agile methodologies ...							
Impact security more negatively overall							
Do less on security							
Do not deal with security directly							
Produce more secure software overall							
Perform equally overall in terms of security							
Are more secure because of testing							
Provide team members more opportunities to mitigate security issues							
Are better on security because of their iterative nature							
Can discover security related issues faster							
Awareness of security in Agile Projects...							
Allows for increased consideration of security throughout the project							
Requires elaboration steps to be included in the iteration							
Contributes more to security for the project than training can							
Allows team members to voice their security concerns openly							
Reduces security risks overall							
Adds more focus/emphasis on security within the project							
Should include upfront planning for security specific issues							
Makes stakeholders more considerate of potential security issues							
Requires organizational maturity							
Requires organizational support							
Should be required in certain projects only							
Is important to all projects overall							
Is important to some projects only							
Is only important if security is required for the project							
Accelerated Schedules of Agile Projects ...							
Does not hinder security							
Affect security less than additional requirements affect security							
Affect security less than more functionality affect security							
Do not affect security substantially							
Are affected by security requirements							
Affect every aspect of the project including security							
Affect security practices negatively							
Increases the overall risk including security risk to the project							
Put major focus on achieving functionality and not much else							
Contributes to less secure software							
Internal Agile Projects...							
Can rely on the infrastructure for security							
Are less concerned with security							
Typically have less security requirements							
Do not need security in some cases							
Can have less security if they're not exposed publically							

Figure D-1: Survey Questions

Appendix E: Questionnaire Statistics Data

No	Number of Participants	Missing Values	Case with |z| > 3.29	Mean	5% Trimmed Mean	Δ Mean*	Skewness	Kurtosis
1	**148**	**0**	**0.7**	**5.76**	**5.76**	**0.0**	**-0.84**	**0.64**
2	**148**	**0**	**0.7**	**5.74**	**5.72**	**0.02**	**-0.96**	**1.42**
3	**148**	**0**	**0.7**	**5.76**	**5.75**	**0.01**	**-0.8**	**0.45**
4	**148**	**0**	**0.7**	**5.89**	**5.91**	**-0.02**	**-0.9**	**1.25**
5	**148**	**0**	**0.7**	**5.72**	**5.73**	**-0.01**	**-1.04**	**1.69**
6	148	0	0.0	5.29	5.32	-0.03	-0.59	-0.14
7	148	0	0.0	3.80	3.7	0.1	-0.02	-0.73
8	148	0	0.0	4.99	4.99	0.0	-0.62	0.24
9	148	0	0.0	5.3	5.31	-0.01	-0.79	0.18
10	148	0	0.0	5.27	5.37	-0.1	-0.98	0.91
11	148	0	0.0	2.61	2.35	0.26	0.78	-0.28
12	148	0	0.0	3.71	3.62	0.09	0.02	-1.18
13	148	0	0.0	5.04	5.07	-0.03	-0.55	-0.05
14	**148**	**0**	**0.7**	**5.49**	**5.57**	**-0.08**	-1.04	1.47
15	148	0	0.0	4.63	4.48	0.15	-0.31	-0.49
16	148	0	0.0	4.41	4.39	0.02	-0.51	-0.47
17	**144**	**4**	**0.7**	**5.81**	**5.84**	**-0.03**	**-1.03**	**1.68**
18	**144**	**4**	**0.7**	**5.74**	**5.81**	**-0.07**	**-0.93**	**1.45**
19	**144**	**4**	**0.7**	**5.74**	**5.77**	**-0.03**	**-0.77**	**1.1**
20	**144**	**4**	**0.7**	**5.89**	**5.96**	**-0.07**	**-1.1**	**2.0**
21	**144**	**4**	**0.7**	**5.82**	**5.87**	**-0.05**	**-1.06**	**1.42**
22	**144**	**4**	**0.7**	**5.41**	**5.46**	**-0.05**	**-1.11**	**1.51**
23	**144**	**4**	**0.7**	**5.67**	**5.67**	**0.0**	**-0.71**	**0.77**
24	144	4	0.0	5.58	5.63	-0.05	-1.12	1.68
25	138	10	0.0	3.99	3.88	0.11	-0.06	-1.03
26	138	10	0.0	4.99	4.96	0.03	-0.37	-0.11
27	138	10	0.0	5.18	5.22	-0.04	-0.6	-0.06
28	138	10	0.0	5.13	5.17	-0.04	-0.57	0.12
29	138	10	0.0	5.14	5.11	0.03	-0.35	-0.56
30	138	10	0.0	5.05	5.07	-0.02	-0.58	-0.13
31	138	10	0.0	5.22	5.27	-0.05	-0.55	0.19
32	138	10	0.0	4.47	4.4	0.07	-0.31	-0.65
33	130	18	0.0	5.24	5.3	-0.06	-0.86	0.32
34	**130**	**18**	**0.8**	**5.38**	**5.42**	**-0.04**	**-0.97**	**1.0**
35	130	18	0.0	5.41	5.41	0.0	-0.76	0.48
36	**130**	**18**	**0.8**	**5.45**	**5.46**	**-0.01**	**-0.63**	**0.21**
37	**130**	**18**	**0.8**	**5.35**	**5.35**	**0.0**	**-0.82**	**1.44**
38	130	18	0.0	4.81	4.81	0.0	-0.08	-0.62
39	**130**	**18**	**0.8**	**5.45**	**5.44**	**0.01**	**-0.67**	**0.62**
40	130	18	0.0	4.41	4.36	0.05	-0.34	-0.55

No	Number of Participants	Missing Values	Case with |z| > 3.29	Mean	5% Trimmed Mean	Δ Mean*	Skewness	Kurtosis
41	130	18	0.0	5.31	5.33	-0.02	-0.87	0.33
42	130	18	0.0	5.25	5.27	-0.02	-0.79	0.2

Table E-1: Analysis of Responses for each item related to RQ1

No	Number of Participants	Missing Values	Case with |z| > 3.29	Mean	5% Trimmed Mean	Δ Mean*	Skewness	Kurtosis
43	129	19	0.0	4.7	4.84	-0.14	-0.42	-0.18
44	129	19	0.0	4.66	4.8	-0.14	-0.26	-0.62
45	129	19	0.0	4.4	4.5	-0.1	-0.16	-0.63
46	129	19	0.0	4.06	4.02	0.04	0.03	0.64
47	129	19	0.0	4.56	4.57	-0.01	-0.2	-0.46
48	129	19	0.0	4.38	4.39	-0.01	-0.41	-0.08
49	129	19	0.0	4.82	4.87	-0.05	-0.38	-0.27
50	129	19	0.0	4.50	4.52	-0.02	-0.28	-0.26
51	129	19	0.0	5.15	5.24	0.09	-0.62	0.0
52	124	24	0.0	5.65	5.67	-0.02	-0.6	-0.06
53	124	24	0.0	4.65	4.61	0.04	-0.2	-0.33
54	124	24	0.0	4.65	4.63	0.02	-0.15	-0.4
55	124	24	0.0	5.44	5.47	-0.03	-0.77	0.61
56	124	24	0.0	5.21	5.23	-0.02	-0.42	0.05
57	124	24	0.0	5.47	5.51	-0.04	-0.9	1.21
58	124	24	0.0	5.15	5.19	-0.04	-0.62	-0.18
59	124	24	0.0	5.27	5.28	-0.01	-0.78	0.44
60	124	24	0.0	5.06	5.09	-0.03	-0.67	-0.1
61	**124**	**24**	**0.8**	**5.59**	**5.67**	**-0.08**	**-1.21**	**1.93**
62	124	24	0.0	3.73	3.63	0.1	0.09	-0.93
63	124	24	0.0	5.1	5.14	-0.04	-1.03	0.67
64	124	24	0.0	3.64	3.54	0.1	0.42	-0.72
65	124	24	0.0	4.15	4.11	0.04	-0.13	-1.18
66	122	26	0.0	3.89	3.93	-0.04	-0.1	-1.05
67	122	26	0.0	4.28	4.25	0.03	-0.01	0.12
68	122	26	0.0	4.35	4.32	0.03	-0.04	0.03
69	122	26	0.0	3.7	3.73	-0.03	-0.03	-0.92
70	122	26	0.0	4.94	5.0	-0.06	-0.89	1.28
71	**122**	**26**	**0.8**	**5.27**	**5.31**	**-0.04**	**-0.6**	**0.62**
72	122	26	0.0	4.09	4.06	0.03	-0.23	-0.45
73	122	26	0.0	4.1	4.06	0.04	-0.29	-0.83
74	122	26	0.0	4.07	4.04	0.03	-0.34	-0.75

No	Number of Participants	Missing Values	Case with |z| > 3.29	Mean	5% Trimmed Mean	Δ Mean*	Skewness	Kurtosis
75	122	26	0.0	3.74	3.65	0.09	0.04	-0.72
76	120	28	0.0	3.62	3.62	0.0	0.07	-0.97
77	120	28	0.0	4.42	4.44	-0.02	-0.36	-0.75
78	120	28	0.0	4.73	4.78	-0.05	-0.62	-0.57
79	120	28	0.0	3.96	3.95	0.01	0.13	-0.87
80	120	28	0.0	4.29	4.32	-0.03	-0.35	-0.91

Table E-2: Analysis of Responses for each item related to RQ2

RQ1-Combining Security & Agility

RQ1A Dedicated Security Engineer

No	Statement		Strongly Agree	Agree	Somewhat Agree	Neutral	Somewhat Disagree	Disagree	Strongly Disagree	Mean	Std. Deviation	Rank
Having a Security Engineer within the agile team can												
1	Increase each team member's security knowledge	Freq.	35	**66**	28	16	2	1	0	5.76	1.04	2
		%	23.6	**44.6**	18.9	10.8	1.4	0.7	0			
2	Help in producing more secure software	Freq.	38	**60**	28	19	2	0	1	5.74	1.09	4
		%	25.7	**40.5**	18.9	12.8	1.4	0	0.7			
3	Help solve security issues faster	Freq.	39	**59**	31	15	3	1	0	5.76	1.07	3
		%	26.4	**39.9**	20.9	10.1	2.0	0.7	0			
4	Increase awareness of security issues	Freq.	38	**70**	27	12	0	1	0	5.89	0.93	1
		%	25.7	**47.3**	18.2	8.1	0	0.7	0			
5	Uncover security issues earlier in the project	Freq.	42	**50**	36	16	1	2	1	5.72	1.15	5
		%	28.4	**33.8**	24.3	10.8	0.7	1.4	0.7			
6	Help the customer make better security decisions	Freq.	29	**45**	31	33	3	7	0	5.29	1.32	8
		%	19.6	**30.4**	20.9	22.3	2.0	4.7	0			
7	Lengthen the iteration	Freq.	6	16	26	**45**	16	27	12	3.8	1.58	14
		%	4.1	10.8	17.6	**30.4**	10.8	18.2	8.1			
8	Help Establish/Increase the overall security assurance of Agile	Freq.	19	**42**	33	39	6	6	3	4.99	1.38	11
		%	12.8	**28.4**	22.3	26.4	4.1	4.1	2.0			
9	Help the project achieve compliance with security standards faster	Freq.	26	**54**	29	24	9	5	1	5.30	1.34	7
		%	17.6	**36.5**	19.6	16.2	6.1	3.4	0.7			
10	Lower the security risk of the produced software further than developers alone can	Freq.	27	**45**	44	16	8	5	3	5.27	1.38	9
		%	18.2	**30.4**	29.7	10.8	5.4	3.4	2.0			
11	Decrease the need for highly experienced developers architects within the team	Freq.	1	10	8	21	27	34	**47**	2.61	1.56	16
		%	0.7	6.8	5.4	14.2	18.2	23.0	**31.8**			
12	Alleviate other team members from having to focus on specific security issues	Freq.	6	21	**32**	20	21	30	18	3.71	1.77	15
		%	4.1	14.2	**21.6**	13.5	14.2	20.3	12.2			
13	Maximize the precision of security requirement assessment	Freq.	14	**52**	29	39	8	5	1	5.04	1.27	10
		%	9.5	**35.1**	19.6	26.4	5.4	3.4	0.7			
14	Put more importance on security for the project and the team	Freq.	27	**58**	36	20	2	4	1	5.49	1.24	6
		%	18.2	**39.2**	24.3	13.5	1.4	2.7	0.7			
15	Mitigate the lack of mature security vulnerability assessment tools	Freq.	13	30	**39**	38	12	15	1	4.63	1.42	12
		%	8.8	20.3	**26.4**	25.7	8.1	10.1	0.7			
16	Allow him/her to serve as an independent on-site security auditor for the project	Freq.	12	32	33	**35**	13	10	13	4.41	1.67	13
		%	8.1	21.6	22.3	**23.6**	8.8	6.8	8.8			
Mean for total										**4.96**		

Table E-3: Frequency and descriptive statistics for 'dedicated security engineer' items

RQ1B Software with Security in Mind

No	Statement		Strongly Agree	Agree	Somewhat Agree	Neutral	Somewhat Disagree	Disagree	Strongly Disagree	Mean	Std. Deviation	Rank
Building Software with Security in mind in agile projects can												
17	Establish/Increase the Security Assurance for the project	Freq.	32	**69**	31	8	3	1	0	5.81	0.98	3
		%	22.2	**47.9**	21.5	5.6	2.1	0.7	0			
18	Help in writing more secure code	Freq.	28	**68**	35	9	3	1	0	5.74	0.96	4
		%	19.4	**47.2**	24.3	6.3	2.1	0.7	0			
19	Help in achieving security focused testing	Freq.	31	**61**	40	9	2	1	0	5.74	0.95	5
		%	21.5	**42.4**	27.8	6.3	1.4	0.7	0			
20	Expand the awareness of security throughout the project	Freq.	38	**66**	31	5	3	1	0	5.89	0.97	1
		%	26.4	**45.8**	21.5	3.5	2.1	0.7	0			
21	Put more overall focus on security	Freq.	34	**70**	25	11	3	1	0	5.82	0.99	2
		%	23.6	**48.6**	17.4	7.6	2.1	0.7	0			
22	Promote the collective ownership of security by all stakeholders	Freq.	20	**61**	37	16	5	4	1	5.41	1.19	8
		%	13.9	**42.4**	25.7	11.1	3.5	2.8	0.7			
23	Help in decreasing vulnerabilities in the software	Freq.	29	**58**	42	11	3	1	0	5.67	1.01	6
		%	20.1	**40.3**	29.2	7.6	2.1	0.7	0			
24	Reduce security risk to the project and the resulting software	Freq.	25	**64**	36	12	3	4	0	5.58	1.10	7
		%	17.4	**44.4**	25.0	8.3	2.1	2.8	0			
Mean for total										**5.70**		

Table E-4: Frequency and descriptive statistics for ‘software with security in mind’ items

RQ1C Security Controls

No	Statement		Strongly Agree	Agree	Somewhat Agree	Neutral	Somewhat Disagree	Disagree	Strongly Disagree	Mean	Std. Deviation	Rank
Using Software Security Controls (pre-made security modules) in the agile project can												
25	Alleviate developers from having to focus on security	Freq.	5	23	**32**	20	28	23	7	3.99	1.60	8
		%	3.6	16.7	**23.2**	14.5	20.3	16.7	5.1			
26	Be beneficial throughout the project	Freq.	12	38	**41**	35	7	5	0	4.99	1.19	6
		%	8.7	27.5	**29.7**	25.4	5.1	3.6	0			
27	Be reused on multiple projects	Freq.	17	**47**	36	25	8	5	0	5.18	1.24	2
		%	12.3	**34.1**	26.1	18.1	5.8	3.6	0			
28	Become a part of the framework/infrastructure	Freq.	18	**43**	34	31	7	4	1	5.13	1.27	4
		%	13.0	**31.2**	24.6	22.5	5.1	2.9	0.7			
29	Be available as part of the infrastructure before the next project begins	Freq.	15	**47**	32	32	10	2	0	5.14	1.19	3
		%	10.9	**34.1**	23.2	23.2	7.2	1.4	0			
30	Reduce costs if used on multiple projects	Freq.	21	**38**	33	30	6	9	1	5.05	1.41	5
		%	15.2	**27.5**	23.9	21.7	4.3	6.5	0.7			
31	Help the resulting software to have reduced weaknesses/vulnerabilities	Freq.	16	**46**	40	28	4	4	0	5.22	1.15	1
		%	11.6	**33.3**	29.0	20.3	2.9	2.9	0			
32	Reduce reliance on individuals within the team for security expertise	Freq.	10	28	**36**	28	18	15	3	4.47	1.51	7
		%	7.2	20.3	**26.1**	20.3	13.0	10.9	2.2			
Mean for total										**4.89**		

Table E-5: Frequency and descriptive statistics for ‘security controls’ items

RQ1F Experience of Developers

No	Statement		Strongly Agree	Agree	Somewhat Agree	Neutral	Somewhat Disagree	Disagree	Strongly Disagree	Mean	Std. Deviation	Rank
Experienced Developers within the agile team can												
33	Create/write more secure user stories	Freq.	16	**54**	25	21	9	4	1	5.24	1.30	8
		%	12.3	**41.5**	19.2	16.2	6.9	3.1	0.8			
34	Discover more security issues/vulnerabilities/bugs than regular developers	Freq.	18	**55**	30	17	7	2	1	5.38	1.20	4
		%	13.8	**42.3**	23.1	13.1	5.4	1.5	0.8			
35	Help implement security requirements faster	Freq.	16	**55**	34	18	5	2	0	5.41	1.09	3
		%	12.3	**42.3**	26.2	13.8	3.8	1.5	0			
36	Add to the team's overall security awareness	Freq.	14	**60**	31	22	2	1	0	5.45	0.99	1
		%	10.8	**46.2**	23.8	16.9	1.5	0.8	0			
37	Mitigate more security issues than regular developers	Freq.	15	**50**	39	21	3	1	1	5.35	1.08	5
		%	11.5	**38.5**	30.0	16.2	2.3	0.8	0.8			
38	Be more effective than using tools on security	Freq.	8	**35**	29	43	12	3	0	4.81	1.18	9
		%	6.2	**26.9**	22.3	33.1	9.2	2.3	0			
39	Increase the overall security of the software	Freq.	12	**59**	38	18	2	1	0	5.45	0.94	2
		%	9.2	**45.4**	29.2	13.8	1.5	0.8	0			
40	Be more beneficial than QA when it comes to security focused testing	Freq.	6	**32**	25	**32**	21	9	5	4.41	1.49	10
		%	4.6	**24.6**	19.2	**24.6**	16.2	6.9	3.8			
41	Play a key role in avoiding to introduce security vulnerabilities into software	Freq.	17	**53**	31	16	8	5	0	5.31	1.25	6
		%	13.1	**40.8**	23.8	12.3	6.2	3.8	0			
42	Predict security vulnerabilities at earlier stages of the project	Freq.	16	**50**	33	18	8	5	0	5.25	1.24	7
		%	12.3	**38.5**	25.4	13.8	6.2	3.8	0			
Mean for total										**5.20**		

Table E-6: Frequency and descriptive statistics for 'experience of developers' items

RQ2-Change Agile Practices for Security

RQ2A Agile vs. Traditional Waterfall

No	Statement		Strongly Agree	Agree	Somewhat Agree	Neutral	Somewhat Disagree	Disagree	Strongly Disagree	Mean	Std. Deviation	Rank
Compared to Traditional Waterfall, Agile methodologies												
43	Impact security less negatively overall*	Freq.	23	18	27	**40**	8	5	8	4.7	1.63	3
		%	17.8	14.0	20.9	**31.0**	6.2	3.9	6.2			
44	Do more on security*	Freq.	21	21	26	**33**	14	10	4	4.66	1.61	4
		%	16.3	16.3	20.2	**25.6**	10.9	7.8	3.1			
45	Deal with security directly*	Freq.	15	21	22	**37**	16	12	6	4.4	1.62	7
		%	11.6	16.3	17.1	**28.7**	12.4	9.3	4.7			
46	Produce more secure software overall	Freq.	8	8	19	**64**	15	9	6	4.06	1.33	9
		%	6.2	6.2	14.7	**49.6**	11.6	7.0	4.7			
47	Perform equally overall in terms of security	Freq.	14	26	21	**40**	17	7	4	4.56	1.51	5
		%	10.9	20.2	16.3	**31.0**	13.2	5.4	3.1			
48	Are more secure because of testing	Freq.	8	18	**38**	**38**	10	12	5	4.38	1.44	8
		%	6.2	14.0	**29.5**	**29.5**	7.8	9.3	3.9			
49	Provide team members more opportunities to mitigate security issues	Freq.	15	26	**40**	27	12	8	1	4.82	1.38	2
		%	11.6	20.2	**31.0**	20.9	9.3	6.2	0.8			
50	Are better on security because of their iterative nature	Freq.	13	19	33	**38**	11	11	4	4.50	1.49	6
		%	10.1	14.7	25.6	**29.5**	8.5	8.5	3.1			
51	Can discover security related issues faster	Freq.	20	**39**	31	25	8	5	1	5.15	1.35	1
		%	15.5	**30.2**	24.0	19.4	6.2	3.9	0.8			
Mean for total										**4.58**		

*items are originally negative statements that were reversed

Table E-7: Frequency and descriptive statistics for ‘Agile vs. Waterfall’ items

RQ2B Awareness of Security to Agile

No	Statement		Strongly Agree	Agree	Somewhat Agree	Neutral	Somewhat Disagree	Disagree	Strongly Disagree	Mean	Std. Deviation	Rank
Awareness of security in Agile Projects												
52	Allows for increased consideration of security throughout the project	Freq.	20	**60**	27	15	2	0	0	5.65	0.94	1
		%	16.1	**48.4**	21.8	12.1	1.6	0	0			
53	Requires elaboration steps to be included in the iteration	Freq.	10	26	28	**40**	11	8	1	4.65	1.34	10
		%	8.1	21.0	22.6	**32.3**	8.9	6.5	0.8			
54	Contributes more to security for the project than training can	Freq.	9	29	24	**42**	13	6	1	4.65	1.31	11
		%	7.3	23.4	19.4	**33.9**	10.5	4.8	0.8			
55	Allows team members to voice their security concerns openly	Freq.	18	**52**	28	22	1	3	0	5.44	1.11	4
		%	14.5	**41.9**	22.6	17.7	0.8	2.4	0			
56	Reduces security risks overall	Freq.	17	**37**	34	31	1	4	0	5.21	1.17	6
		%	13.7	**29.8**	27.4	25.0	0.8	3.2	0			
57	Adds more focus/emphasis on security within the project	Freq.	17	**52**	35	15	2	3	0	5.47	1.07	3
		%	13.7	**41.9**	28.2	12.1	1.6	2.4	0			
58	Should include upfront planning for security specific issues	Freq.	18	**44**	21	27	9	4	1	5.15	1.36	7
		%	14.5	**35.5**	16.9	21.8	7.3	3.2	0.8			
59	Makes stakeholders more considerate of potential security issues	Freq.	13	**49**	33	20	5	4	0	5.27	1.16	5
		%	10.5	**39.5**	26.6	16.1	4.0	3.2	0			
60	Requires organizational maturity	Freq.	20	**40**	19	28	9	5	3	5.06	1.50	9
		%	16.1	**32.3**	15.3	22.6	7.3	4.0	2.4			
61	Requires organizational support	Freq.	26	**52**	26	14	2	3	1	5.59	1.20	2
		%	21.0	**41.9**	21.0	11.3	1.6	2.4	0.8			
62	Should be required in certain projects only	Freq.	8	16	15	**33**	15	21	16	3.73	1.76	13
		%	6.5	12.9	12.1	**26.6**	12.1	16.9	12.9			
63	Is important to all projects overall	Freq.	13	**52**	21	23	6	5	4	5.10	1.45	8
		%	10.5	**41.9**	16.9	18.5	4.8	4.0	3.2			
64	Is important to some projects only	Freq.	9	13	12	27	25	**29**	9	3.64	1.68	14
		%	7.3	10.5	9.7	21.8	20.2	**23.4**	7.3			
65	Is only important if security is required for the project	Freq.	12	**28**	16	19	20	17	12	4.15	1.86	12
		%	9.7	**22.6**	12.9	15.3	16.1	13.7	9.7			
Mean for total										**4.91**		

*items are originally negative statements that were reversed

Table E-8: Frequency and descriptive statistics for ‘awareness of security to Agile’ items

RQ2C Impact of Accelerated Schedule

No	Statement		Strongly Agree	Agree	Somewhat Agree	Neutral	Somewhat Disagree	Disagree	Strongly Disagree	Mean	Std. Deviation	Rank
Accelerated Schedules of Agile Projects												
66	Hinder security*	Freq.	8	12	**34**	22	8	26	12	3.89	1.75	6
		%	6.6	9.8	**27.9**	18.0	6.6	21.3	9.8			
67	Affect security less than additional requirements affect security	Freq.	7	19	15	**55**	16	6	4	4.28	1.35	5
		%	5.7	15.6	12.3	**45.1**	13.1	4.9	3.3			
68	Affect security less than more functionality affect security	Freq.	8	19	18	**54**	12	8	3	4.35	1.35	3
		%	6.6	15.6	14.8	**44.3**	9.8	6.6	2.5			
69	Affect security substantially*	Freq.	4	10	**30**	27	14	26	11	3.70	1.60	4
		%	3.3	8.2	**24.6**	22.1	11.5	21.3	9.0			
70	Are affected by security requirements	Freq.	9	**37**	34	32	4	2	4	4.94	1.29	2
		%	7.4	**30.3**	27.9	26.2	3.3	1.6	3.3			
71	Affect every aspect of the project including security	Freq.	18	**39**	32	28	2	2	1	5.27	1.21	1
		%	14.8	**32.0**	26.2	23.0	1.6	1.6	0.8			
72	Affect security practices negatively	Freq.	7	16	26	**35**	18	10	10	4.09	1.57	8
		%	5.7	13.1	21.3	**28.7**	14.8	8.2	8.2			
73	Increases the overall risk including security risk to the project	Freq.	9	17	**31**	27	9	16	13	4.10	1.74	7
		%	7.4	13.9	**25.4**	22.1	7.4	13.1	10.7			
74	Put major focus on achieving functionality and not much else	Freq.	7	18	**31**	27	13	12	14	4.07	1.70	9
		%	5.7	14.8	**25.4**	22.1	10.7	9.8	11.5			
75	Contributes to less secure software	Freq.	8	10	20	**37**	13	19	15	3.74	1.69	10
		%	6.6	8.2	16.4	**30.3**	10.7	15.6	12.3			
Mean for total										**4.24**		

Table E-9: Frequency and descriptive statistics for ‘impact of accelerated schedule’ items

RQ2D Reduced Security for Internal Projects

No	Statement		Strongly Agree	Agree	Somewhat Agree	Neutral	Somewhat Disagree	Disagree	Strongly Disagree	Mean	Std. Deviation	Rank
Internal Agile Projects												
76	Can rely on the infrastructure for security	Freq.	1	17	20	22	**28**	21	11	3.62	1.56	5
		%	0.8	14.2	16.7	18.3	**23.3**	17.5	9.2			
77	Are less concerned with security	Freq.	8	24	**35**	19	15	16	3	4.43	1.55	2
		%	6.7	20.0	**29.2**	15.8	12.5	13.3	2.5			
78	Typically have less security requirements	Freq.	8	**39**	30	15	14	12	2	4.73	1.51	1
		%	6.7	**32.5**	25.0	12.5	11.7	10.0	1.7			
79	Do not need security in some cases	Freq.	7	20	17	22	**32**	15	7	3.96	1.62	4
		%	5.8	16.7	14.2	18.3	**26.7**	12.5	5.8			
80	Can have less security if they're not exposed publically	Freq.	6	**28**	**28**	20	15	17	6	4.29	1.64	3
		%	5.0	**23.3**	**23.3**	16.7	12.5	14.2	5.0			
Mean for total										**4.20**		

Table E-10: Frequency and descriptive statistics for 'reduced security for internal projects' items

Appendix F: Questionnaire Factor Analysis

Section 1: Combining Security & Agility

Total Variance Explained							
Component	Initial Eigenvalues			Extraction Sums of Squared Loadings			Rotation Sums of Squared Loadings[a]
	Total	% of Variance	Cumulative %	Total	% of Variance	Cumulative %	Total
1	14.571	37.362	37.362	14.571	37.362	37.362	10.336
2	4.549	11.663	49.025	4.549	11.663	49.025	7.769
3	2.839	7.278	56.304	2.839	7.278	56.304	7.491
4	2.324	5.958	62.262	2.324	5.958	62.262	8.933
5	1.684	4.319	66.581	1.684	4.319	66.581	4.108
6	1.099	2.819	69.399	1.099	2.819	69.399	1.333
7	.969	2.484	71.884				
8	.861	2.207	74.091				
9	.845	2.166	76.257				
10	.792	2.030	78.287				
11	.769	1.972	80.258				
12	.720	1.845	82.103				
13	.630	1.614	83.718				
14	.551	1.414	85.132				
15	.511	1.309	86.441				
16	.482	1.236	87.677				
17	.460	1.179	88.856				
18	.400	1.026	89.882				
19	.372	.954	90.836				
20	.351	.899	91.735				
21	.320	.821	92.556				
22	.294	.753	93.309				
23	.278	.714	94.023				
24	.263	.674	94.696				
25	.234	.599	95.296				
26	.217	.557	95.853				
27	.209	.535	96.388				
28	.184	.472	96.860				
29	.174	.446	97.306				
30	.163	.417	97.723				

31	.153	.392	98.116				
32	.129	.332	98.447				
33	.120	.307	98.754				
34	.111	.286	99.040				
35	.101	.258	99.298				
36	.095	.244	99.541				
37	.071	.182	99.723				
38	.058	.150	99.873				
39	.050	.127	100.000				
Extraction Method: Principal Component Analysis.							
a. When components are correlated, sums of squared loadings cannot be added to obtain a total variance.							

Table F-1: Total Variance Explained without specifying number of factors (Combining Security & Agility)

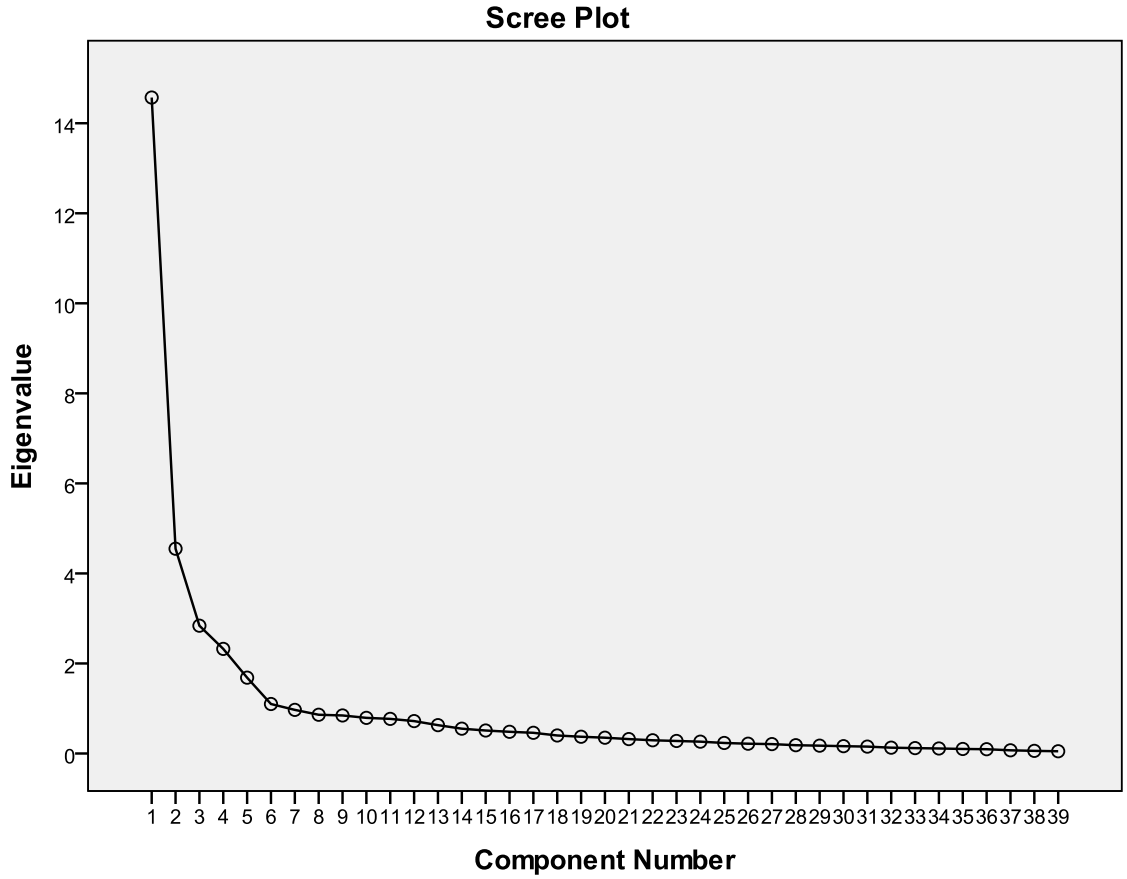


Figure F-1: Scree Plot (Combining Security & Agility)

Pattern Matrix[a]						
	Component					
	1	2	3	4	5	6
SecEng5	.856					
SecEng4	.812					
SecEng1	.784					
SecEng2	.750					
SecEng3	.710					
SecEng9	.695					
SecEng10	.688					
SecEng14	.687					.332
SecEng13	.675					
SecEng8	.650					
SecEng6	.617					
ExpDev5		.867				
ExpDev2		.814				
ExpDev3		.808				
ExpDev7		.807				
ExpDev9		.793				
ExpDev4		.757				
ExpDev10		.741				
ExpDev1		.642				
ExpDev6		.503				.411
SecCon3			.891			
SecCon4			.869			
SecCon6			.832			
SecCon5			.830			
SecCon2			.650			
SecCon7			.474			
SecInMind6				-.769		
SecInMind3				-.716		
SecInMind5				-.707		
SecInMind2				-.703		
SecInMind1				-.696		
SecInMind4				-.688		
SecInMind8				-.664		
SecInMind7				-.599		
SecCon1					.755	
SecEng12					.734	
SecCon8			.354		.611	
SecEng15					.599	.324

ExpDev8		.460			.480	

Extraction Method: Principal Component Analysis.
Rotation Method: Oblimin with Kaiser Normalization.

a. Rotation converged in 16 iterations.

Table F-2: Pattern Matrix of Oblimin Rotation without specifying number of factors (Combining Security & Agility)

Structure Matrix						
	Component					
	1	2	3	4	5	6
SecEng5	.851		.329	-.349		
SecEng4	.845		.339	-.468		
SecEng2	.835	.313	.423	-.490		
SecEng1	.831			-.464		
SecEng14	.763		.331	-.432		.354
SecEng3	.751			-.417		
SecEng8	.745		.372	-.462		
SecEng13	.745		.380		.347	.308
SecEng10	.744		.396	-.305		
SecEng9	.734		.318			
SecEng6	.684		.303	-.390		
ExpDev7		.875		-.466		
ExpDev5	.306	.873		-.312		
ExpDev2	.395	.814		-.315		
ExpDev3		.813		-.354		
ExpDev9		.804		-.365		
ExpDev4		.801		-.392		
ExpDev10		.792		-.373		
ExpDev1		.639	.324			
ExpDev6		.587				.493
ExpDev8		.568		-.360	.504	
SecCon3	.416		.922	-.336		
SecCon4	.355		.894	-.375		
SecCon5	.404		.886	-.385		
SecCon6	.368		.862	-.313		
SecCon2	.501		.806	-.434	.445	
SecCon7	.515		.680	-.476	.455	
SecInMind2	.515	.338	.421	-.817		
SecInMind3	.461	.403		-.811		

SecInMind5	.437	.384	.389	-.805		
SecInMind1	.518	.330		-.798		
SecInMind4	.493		.449	-.779		
SecInMind8	.432	.373	.371	-.772		
SecInMind6				-.743		
SecInMind7	.450	.369	.372	-.732	.321	
SecCon1			.401		.786	
SecEng12	.341				.771	
SecCon8			.498		.684	
SecEng15	.436				.669	.344

Extraction Method: Principal Component Analysis.
Rotation Method: Oblimin with Kaiser Normalization.

Table F-3: Structure Matrix of Oblimin Rotation without specifying number of factors (Combining Security & Agility)

Total Variance Explained							
Component	Initial Eigenvalues			Extraction Sums of Squared Loadings			Rotation Sums of Squared Loadings[a]
	Total	% of Variance	Cumulative %	Total	% of Variance	Cumulative %	Total
1	14.571	37.362	37.362	14.571	37.362	37.362	11.556
2	4.549	11.663	49.025	4.549	11.663	49.025	8.567
3	2.839	7.278	56.304	2.839	7.278	56.304	9.258
4	2.324	5.958	62.262	2.324	5.958	62.262	4.640
5	1.684	4.319	66.581				
6	1.099	2.819	69.399				
7	.969	2.484	71.884				
8	.861	2.207	74.091				
9	.845	2.166	76.257				
10	.792	2.030	78.287				
11	.769	1.972	80.258				
12	.720	1.845	82.103				
13	.630	1.614	83.718				
14	.551	1.414	85.132				
15	.511	1.309	86.441				
16	.482	1.236	87.677				
17	.460	1.179	88.856				
18	.400	1.026	89.882				
19	.372	.954	90.836				
20	.351	.899	91.735				
21	.320	.821	92.556				
22	.294	.753	93.309				
23	.278	.714	94.023				
24	.263	.674	94.696				
25	.234	.599	95.296				
26	.217	.557	95.853				
27	.209	.535	96.388				
28	.184	.472	96.860				
29	.174	.446	97.306				
30	.163	.417	97.723				
31	.153	.392	98.116				
32	.129	.332	98.447				
33	.120	.307	98.754				

34	.111	.286	99.040				
35	.101	.258	99.298				
36	.095	.244	99.541				
37	.071	.182	99.723				
38	.058	.150	99.873				
39	.050	.127	100.000				
Extraction Method: Principal Component Analysis.							
a. When components are correlated, sums of squared loadings cannot be added to obtain a total variance.							

Table F-4: Total Variance Explained with 4 Factors (Combining Security & Agility)

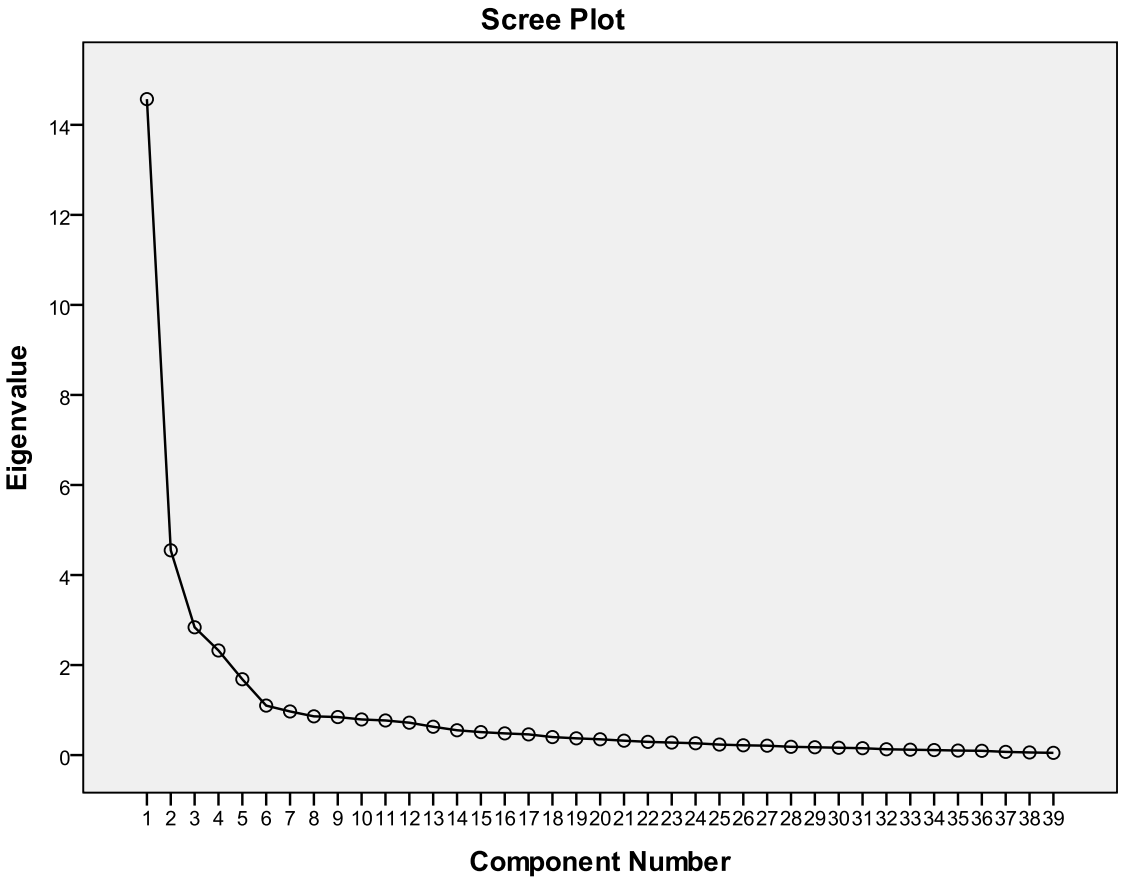


Figure F-2: Scree Plot (Combining Security & Agility)

Pattern Matrix[a]				
	Component			
	1	2	3	4
SecEng5	.890			
SecEng4	.862			
SecEng1	.859			
SecEng14	.784			
SecEng3	.754			
SecEng9	.752			
SecEng2	.752			
SecEng13	.738			
SecEng10	.717			
SecEng8	.706			
SecEng6	.646			
SecEng15	.471			
ExpDev7		.879		
ExpDev5		.861		
ExpDev9		.827		
ExpDev10		.810		
ExpDev4		.779		
ExpDev3		.745		
ExpDev2		.740		
ExpDev6		.703		
ExpDev8		.662		
ExpDev1		.590		
SecCon4			.877	
SecCon3			.860	
SecCon5			.854	
SecCon6			.851	
SecCon2			.788	
SecCon8			.672	.317
SecCon7			.656	
SecCon1			.654	.313
SecInMind6				-.601
SecInMind4				-.594
SecInMind3	.301			-.562
SecEng12	.344		.355	.558
SecInMind2	.312			-.556
SecInMind5				-.501
SecInMind8				-.473

SecInMind1	.381			-.463
SecInMind7			.311	-.370
Extraction Method: Principal Component Analysis. Rotation Method: Oblimin with Kaiser Normalization.				
a. Rotation converged in 9 iterations.				

Table F-5 : Pattern Matrix of Oblimin Rotation with 4 Factors (Combining Security & Agility)

Structure Matrix				
	Component			
	1	2	3	4
SecEng1	.838		.393	
SecEng5	.835		.362	
SecEng4	.831		.376	-.306
SecEng2	.825	.333	.473	-.305
SecEng14	.772		.336	
SecEng13	.761	.331	.430	
SecEng3	.755		.385	
SecEng8	.748		.399	
SecEng9	.741	.314	.353	
SecEng10	.739		.410	
SecEng6	.689		.377	
SecInMind1	.597	.404	.422	-.589
SecInMind7	.530	.428	.521	-.495
SecEng15	.530		.416	
ExpDev7		.884		
ExpDev5	.348	.865		
ExpDev10		.806		
ExpDev9		.801		
ExpDev2	.415	.790	.302	
ExpDev4		.789		-.315
ExpDev3	.306	.782		
ExpDev6		.642		
ExpDev8		.640		
ExpDev1		.614		
SecCon2	.551		.869	
SecCon5	.437		.862	
SecCon3	.432		.861	

SecCon4	.375		.839	
SecCon6	.399		.834	
SecCon7	.571		.780	
SecCon8	.350		.682	
SecCon1	.312		.647	
SecEng12	.416		.473	.436
SecInMind4	.545	.309	.470	-.688
SecInMind3	.535	.471	.350	-.683
SecInMind2	.583	.403	.475	-.674
SecInMind6	.318	.335		-.663
SecInMind5	.526	.461	.472	-.621
SecInMind8	.507	.428	.484	-.588
Extraction Method: Principal Component Analysis. Rotation Method: Oblimin with Kaiser Normalization.				

Table F-6: Structure Matrix of Oblimin Rotation with 4 Factors (Combining Security & Agility)

Weighted Average Factor Scores for Four Extracted Factors

Participant	Factor Score			
	1	2	3	4
1	68.32	47.71	49.15	20.76
2	73.12	56.18	52.11	24.05
3	66.4	38.71	42.83	28.48
4	65.96	49.28	48.36	26.75
5	62.92	53.57	52.42	29.42
6	62.76	43.51	37.85	23.23
7	50.38	53.98	40.5	24.23
8	69.01	54.38	42.09	22.55
9	41.15	34.66	25.63	10.86
10	30.23	34.88	37.6	25.87
11	74.58	51.43	29.38	29.23
12	45.79	36.26	32.99	15.76
13	70.8	50.6	37.65	20.15
14	65.23	42.21	43.45	21.11
15	59.11	54.46	36.5	20.65
16	51.55	52.87	33.15	22.76
17	65.79	43.63	50.82	19.41
18	68.82	54.37	49	23.67
19	77.44	59.63	57.19	30.42
20	64.35	44.4	28.09	28.22
21	65.12	51.09	40.33	27.59
22	61.96	53.65	49.03	20.45
23	62.51	34.22	47.73	21.92
24	66.33	59.28	41.34	28.96
25	68.2	53.73	45.65	23.41
26	46.5	58.58	28.56	23.19
27	80.84	53.99	56.55	27.06
28	67.83	50.88	46.98	22.91
29	55.55	42.28	40.07	18.83
30	40.55	50.15	45.67	25.66
31	80.53	50.82	57.72	27.65
32	65.29	59.52	35.61	25.87
33	65.7	38.12	38.92	23.99
34	43.1	52.53	22.98	23.81
35	77.16	63.19	38.91	27.13
36	69.85	50.76	41.37	29.28
37	80.74	63.52	58.35	26.6

38	43.79	20.95	27.85	12.45
39	57.68	54.97	36.59	24.56
40	68.24	48.71	50.04	23.52
41	69.26	53.85	49.85	22.8
42	62.93	46.32	45.07	19.11
43	72.2	54.78	48.1	35.08
44	64.22	48.51	46.51	20.93
45	46.14	36.3	33.34	15.2
46	54.89	34.73	36.6	23.22
47	53.61	41.97	36.01	24.59
48	60.22	39.64	43.41	17.07
49	77	52.71	50.52	27.34
50	63.45	59.62	52.31	24.15
51	65.95	54.44	49.27	22
52	67.83	42.61	41.98	19.56
53	62.38	48.05	28.95	26.2
54	60.32	38.64	40.73	23.59
55	61.06	52.91	43.98	30.03
56	66.49	51.28	41.73	22.25
57	64.08	44.29	46.78	22.09
58	70.02	51.46	43.16	24.9
59	61.5	39.52	40.58	24.39
60	67.39	43.62	45.67	25.47
61	48.77	39.35	34.64	20.25
62	74.58	56.5	56.68	26.61
63	60.65	41.74	39.84	21.33
64	61.84	43.68	45.88	27.46
65	58.32	41.03	33.71	22.07
66	57.92	44.18	34.13	24.56
67	80.21	50.38	50.93	28.3
68	76.43	60.88	48.47	34.14
69	73.88	48.16	55.24	29.12
70	78.77	39.77	58.42	25.99
71	66.29	49.98	44.81	25.75
72	46.7	46.43	36.87	24.11
73	59.95	43.34	40.9	20.86
74	61.68	53.48	43.31	26.56
75	73.35	48.75	48.37	26.57
76	58.79	51.97	43.66	20.73
77	63.66	50.03	46.32	20.07
78	47.82	39.69	22.56	15.21

79	53.01	46.31	40.83	22.44
80	62.54	48.04	43.94	23.74
81	66.35	45.47	34.77	23.41
82	75	47.61	47.9	32.67
83	65.7	49.69	47.63	25.74
84	53.59	44.39	37.87	21.73
85	66.73	51.26	47.02	24.28
86	51.84	37.62	37.41	19.26
87	47.07	45.87	42.94	26.68
88	69.06	53.01	47.91	23.08
89	59.5	50.47	49.47	23.61
90	54.37	49.83	35.36	25.03
91	60.8	49.88	42.85	20.74
92	69.34	55.25	54.78	23.22
93	47.94	44.04	41.67	25.09
94	77.27	40.85	35.53	27.3
95	63.79	38.41	45.33	21.22
96	72.72	48.31	39.92	27.96
97	64.67	54.49	50.33	21.71
98	68	53.74	44.99	30.21
99	60.67	57.96	38.89	28.5
100	59.46	54.46	43.16	32.96
101	57.83	53.54	42.51	23.77
102	67.46	52.69	48.97	24.52
103	70.59	52.27	47.76	26.01
104	68.81	52.69	46.3	23.29
105	51.1	52.92	36.87	22.11
106	51.23	32.06	38.15	25.96
107	53.4	36.05	32.79	22.65
108	51.22	42.14	31.33	18.16
109	65.76	47.63	43.07	25.93
110	55.01	38.46	39.8	26.59
111	76.81	52.37	48.56	28.76
112	63.01	41.3	40.07	21.46
113	57.82	34.21	37.9	20.73
114	64.77	47.39	40.76	20.83
115	62.89	54.18	20.99	23.85
116	77.34	54.04	51.06	25.22
117	60.49	42.77	32.85	24.18
118	44.41	34.45	24.53	21.6
119	72.15	55.57	51	25.15

120	63.72	44.22	36.18	21.18
121	63.16	40.51	37.18	22.23
122	50.29	32.67	29.7	14.17
123	46.09	51.49	33.28	15.53
124	47.92	50.75	27.29	19.45
125	52.71	36.54	32.74	18.18
126	47.18	52.41	33.85	21.27
127	62.51	48.63	38.16	17.57
128	59.72	47.05	41.98	23.72
129	68.08	51.45	44.42	23.9
130	54.28	37.18	40.36	19.58

Table F-7: Weighted Average Factor Scores for Four Extracted Factors

Normalized Weighted Average Factor Scores for Four Extracted Factors

Participant	Normalized Factor Score			
	1	2	3	4
1	5.47	4.87	5.11	2.6
2	5.85	5.73	5.41	3.01
3	5.31	3.95	4.45	3.56
4	5.28	5.03	5.02	3.34
5	5.04	5.46	5.44	3.68
6	5.02	4.44	3.93	2.9
7	4.03	5.51	4.21	3.03
8	5.52	5.55	4.37	2.82
9	3.29	3.53	2.66	1.36
10	2.42	3.56	3.91	3.23
11	5.97	5.25	3.05	3.65
12	3.66	3.7	3.43	1.97
13	5.67	5.16	3.91	2.52
14	5.22	4.31	4.51	2.64
15	4.73	5.56	3.79	2.58
16	4.13	5.39	3.44	2.85
17	5.26	4.45	5.28	2.43
18	5.51	5.55	5.09	2.96
19	6.2	6.08	5.94	3.8
20	5.15	4.53	2.92	3.53
21	5.21	5.21	4.19	3.45
22	4.96	5.47	5.09	2.56
23	5	3.49	4.96	2.74
24	5.31	6.05	4.29	3.62
25	5.46	5.48	4.74	2.93
26	3.72	5.97	2.97	2.9
27	6.47	5.51	5.87	3.38
28	5.43	5.19	4.88	2.86
29	4.45	4.31	4.16	2.35
30	3.25	5.12	4.74	3.21
31	6.44	5.18	6	3.46
32	5.22	6.07	3.7	3.23
33	5.26	3.89	4.04	3
34	3.45	5.36	2.39	2.98
35	6.17	6.45	4.04	3.39
36	5.59	5.18	4.3	3.66
37	6.46	6.48	6.06	3.33

38	3.5	2.14	2.89	1.56
39	4.62	5.61	3.8	3.07
40	5.46	4.97	5.2	2.94
41	5.54	5.49	5.18	2.85
42	5.04	4.73	4.68	2.39
43	5.78	5.59	5	4.39
44	5.14	4.95	4.83	2.62
45	3.69	3.7	3.46	1.9
46	4.39	3.54	3.8	2.9
47	4.29	4.28	3.74	3.07
48	4.82	4.04	4.51	2.13
49	6.16	5.38	5.25	3.42
50	5.08	6.08	5.43	3.02
51	5.28	5.55	5.12	2.75
52	5.43	4.35	4.36	2.45
53	4.99	4.9	3.01	3.28
54	4.83	3.94	4.23	2.95
55	4.89	5.4	4.57	3.76
56	5.32	5.23	4.33	2.78
57	5.13	4.52	4.86	2.76
58	5.6	5.25	4.48	3.11
59	4.92	4.03	4.21	3.05
60	5.39	4.45	4.74	3.18
61	3.9	4.01	3.6	2.53
62	5.97	5.76	5.89	3.33
63	4.85	4.26	4.14	2.67
64	4.95	4.46	4.77	3.43
65	4.67	4.19	3.5	2.76
66	4.63	4.51	3.55	3.07
67	6.42	5.14	5.29	3.54
68	6.12	6.21	5.04	4.27
69	5.91	4.91	5.74	3.64
70	6.3	4.06	6.07	3.25
71	5.3	5.1	4.65	3.22
72	3.74	4.74	3.83	3.01
73	4.8	4.42	4.25	2.61
74	4.94	5.45	4.5	3.32
75	5.87	4.97	5.02	3.32
76	4.7	5.3	4.53	2.59
77	5.09	5.1	4.81	2.51
78	3.83	4.05	2.34	1.9

79	4.24	4.72	4.24	2.81
80	5	4.9	4.56	2.97
81	5.31	4.64	3.61	2.93
82	6	4.86	4.98	4.08
83	5.26	5.07	4.95	3.22
84	4.29	4.53	3.93	2.72
85	5.34	5.23	4.88	3.04
86	4.15	3.84	3.89	2.41
87	3.77	4.68	4.46	3.34
88	5.53	5.41	4.98	2.89
89	4.76	5.15	5.14	2.95
90	4.35	5.08	3.67	3.13
91	4.87	5.09	4.45	2.59
92	5.55	5.64	5.69	2.9
93	3.84	4.49	4.33	3.14
94	6.18	4.17	3.69	3.41
95	5.1	3.92	4.71	2.65
96	5.82	4.93	4.15	3.5
97	5.18	5.56	5.23	2.71
98	5.44	5.48	4.67	3.78
99	4.85	5.91	4.04	3.56
100	4.76	5.55	4.48	4.12
101	4.63	5.46	4.42	2.97
102	5.4	5.37	5.09	3.07
103	5.65	5.33	4.96	3.25
104	5.51	5.37	4.81	2.91
105	4.09	5.4	3.83	2.76
106	4.1	3.27	3.96	3.25
107	4.27	3.68	3.41	2.83
108	4.1	4.3	3.25	2.27
109	5.26	4.86	4.47	3.24
110	4.4	3.92	4.13	3.33
111	6.15	5.34	5.04	3.6
112	5.04	4.21	4.16	2.68
113	4.63	3.49	3.94	2.59
114	5.18	4.83	4.23	2.6
115	5.03	5.53	2.18	2.98
116	6.19	5.51	5.3	3.15
117	4.84	4.36	3.41	3.02
118	3.55	3.51	2.55	2.7
119	5.77	5.67	5.3	3.14

120	5.1	4.51	3.76	2.65
121	5.05	4.13	3.86	2.78
122	4.02	3.33	3.09	1.77
123	3.69	5.25	3.46	1.94
124	3.83	5.18	2.84	2.43
125	4.22	3.73	3.4	2.27
126	3.78	5.35	3.52	2.66
127	5	4.96	3.96	2.2
128	4.78	4.8	4.36	2.97
129	5.45	5.25	4.61	2.99
130	4.34	3.79	4.19	2.45

Table F-8: Normalized Weighted Average Factor Scores for Four Extracted Factors

Section 2: Change Agile Practices for Security

Total Variance Explained							
Component	Initial Eigenvalues			Extraction Sums of Squared Loadings			Rotation Sums of Squared Loadings[a]
	Total	% of Variance	Cumulative %	Total	% of Variance	Cumulative %	Total
1	7.322	20.921	20.921	7.322	20.921	20.921	5.727
2	6.179	17.655	38.576	6.179	17.655	38.576	3.902
3	3.174	9.068	47.644	3.174	9.068	47.644	3.740
4	1.966	5.618	53.262	1.966	5.618	53.262	3.839
5	1.721	4.917	58.179	1.721	4.917	58.179	3.697
6	1.443	4.124	62.303	1.443	4.124	62.303	4.557
7	1.313	3.751	66.054	1.313	3.751	66.054	3.135
8	1.156	3.303	69.357	1.156	3.303	69.357	2.546
9	1.067	3.049	72.406	1.067	3.049	72.406	1.632
10	.941	2.688	75.093				
11	.876	2.504	77.597				
12	.727	2.078	79.675				
13	.675	1.927	81.603				
14	.616	1.761	83.363				
15	.592	1.691	85.054				
16	.519	1.482	86.536				
17	.463	1.324	87.860				
18	.447	1.278	89.137				
19	.418	1.194	90.331				
20	.403	1.152	91.483				
21	.371	1.059	92.541				
22	.321	.917	93.458				
23	.296	.847	94.305				
24	.266	.760	95.065				
25	.244	.698	95.763				
26	.230	.658	96.421				
27	.197	.564	96.985				
28	.194	.554	97.539				
29	.172	.492	98.031				
30	.160	.459	98.490				
31	.150	.428	98.917				

32	.124	.355	99.272				
33	.117	.336	99.608				
34	.101	.289	99.896				
35	.036	.104	100.000				
Extraction Method: Principal Component Analysis.							
a. When components are correlated, sums of squared loadings cannot be added to obtain a total variance.							

Table F-9: Total Variance Explained without specifying number of factors (Change Agile Practices for Security)

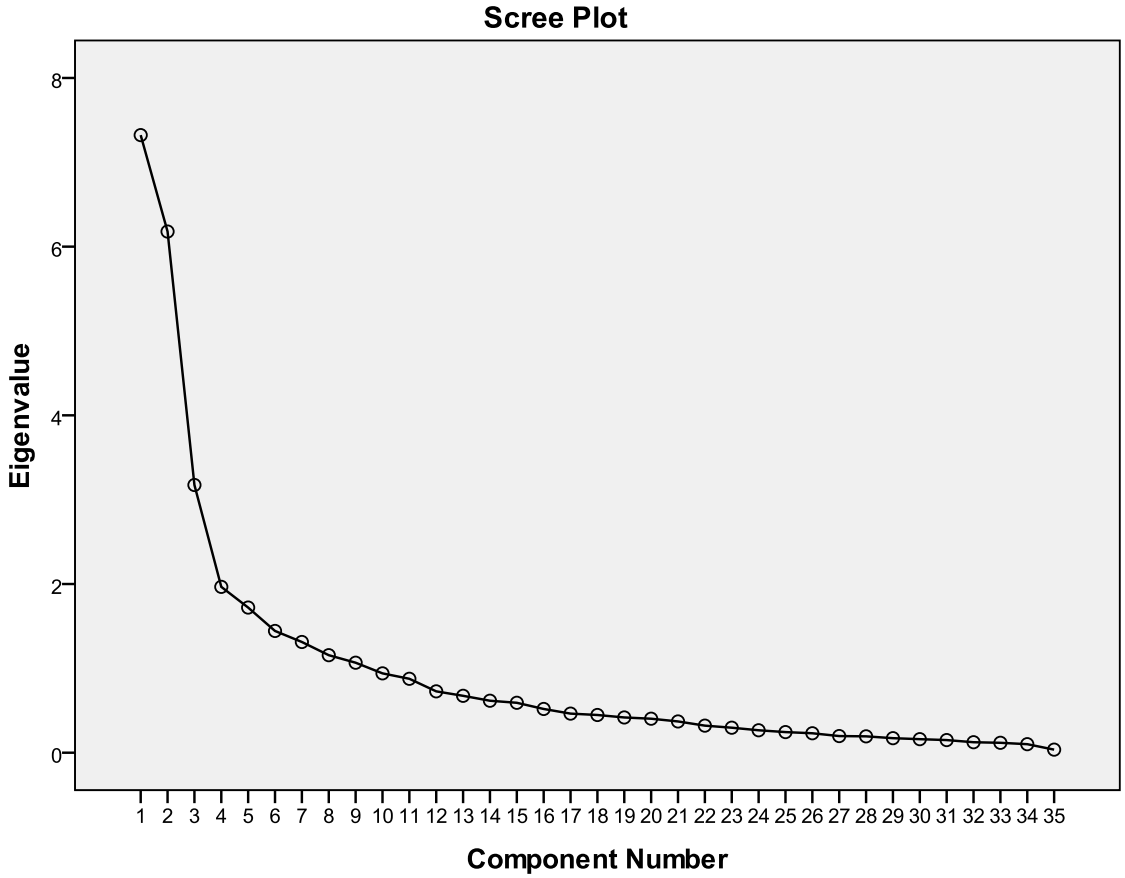


Figure F-3: Scree Plot (Change Agile Practices for Security)

Pattern Matrix[a]									
	Component								
	1	2	3	4	5	6	7	8	9
AccSch7	.769								
AccSch1	.767								
AccSch8	.744								
AccSch4	.738								
AccSch10	.669					-.351			
AccSch9	.642								
AwareOfSec2	.415	.349	-.358						
AwareOfSec6		.872							
AwareOfSec1		.767							
AwareOfSec8		.721							
AwareOfSec5		.600							
AwareOfSec11			-.871						
AwareOfSec13			-.865						
AwareOfSec14			-.747						
InternalAgile4			-.565	.319			.323		
AwareOfSec3			-.443	-.368				-.305	
WaterVs.Agile4				-.863					
WaterVs.Agile6				-.656					
WaterVs.Agile8				-.579					
WaterVs.Agile7				-.571					
InternalAgile1				-.451			.311		
AwareOfSec9					.832				
AwareOfSec10					.798				
AwareOfSec7					.647				
WaterVs.Agile2						.913			
WaterVs.Agile1						.885			
WaterVs.Agile3						.759			
WaterVs.Agile9				-.330		.365			
InternalAgile3							.831		
InternalAgile2							.806		
InternalAgile5							.614		
AccSch6								-.729	
AwareOfSec4								-.666	
WaterVs.Agile5									.896
AccSch5								-.408	.483

Extraction Method: Principal Component Analysis. Rotation Method: Oblimin with Kaiser Normalization.
a. Rotation converged in 26 iterations.

Table F-10: Pattern Matrix of Oblimin Rotation without specifying number of factors (Change Agile Practices for Security)

Structure Matrix									
	Component								
	1	2	3	4	5	6	7	8	9
AccSch7	.870				.387	-.499			
AccSch8	.816					-.511			
AccSch10	.815				.372	-.625			
AccSch9	.783				.452	-.530			
AccSch1	.765								
AccSch4	.733			.332	.306				
AwareOfSec2	.467	.384	-.333		.465				
AwareOfSec6		.847							
AwareOfSec1		.788						-.358	
AwareOfSec8		.752			.303				
AwareOfSec5		.697		-.403				-.337	
AwareOfSec13			-.878						
AwareOfSec11			-.849						
AwareOfSec14			-.786						
InternalAgile4			-.605		-.302		.430		
AwareOfSec3			-.501	-.454				-.368	
WaterVs.Agile4				-.862					
WaterVs.Agile6	-.319	.302		-.741					
WaterVs.Agile8	-.354	.377		-.706		.341			
WaterVs.Agile7	-.413	.391		-.699		.300			
WaterVs.Agile9	-.411	.360		-.477		.471		-.320	
InternalAgile1			-.373	-.416			.388		
AwareOfSec9	.345				.860				
AwareOfSec10					.828			-.421	
AwareOfSec7	.440				.742	-.354			
WaterVs.Agile2	-.397					.931			
WaterVs.Agile1	-.408					.903			
WaterVs.Agile3						.753			
InternalAgile3							.851		
InternalAgile2					.321		.808		
InternalAgile5			-.418				.715		

AwareOfSec4		.509						-.749	
AccSch6	.371							-.735	
WaterVs.Agile5									.870
AccSch5			-.307					-.521	.549
Extraction Method: Principal Component Analysis. Rotation Method: Oblimin with Kaiser Normalization.									

Table F-11: Structure Matrix of Oblimin Rotation without specifying number of factors (Change Agile Practices for Security)

Total Variance Explained							
Component	Initial Eigenvalues			Extraction Sums of Squared Loadings			Rotation Sums of Squared Loadings[a]
	Total	% of Variance	Cumulative %	Total	% of Variance	Cumulative %	Total
1	7.322	20.921	20.921	7.322	20.921	20.921	6.170
2	6.179	17.655	38.576	6.179	17.655	38.576	4.726
3	3.174	9.068	47.644	3.174	9.068	47.644	4.127
4	1.966	5.618	53.262	1.966	5.618	53.262	5.643
5	1.721	4.917	58.179				
6	1.443	4.124	62.303				
7	1.313	3.751	66.054				
8	1.156	3.303	69.357				
9	1.067	3.049	72.406				
10	.941	2.688	75.093				
11	.876	2.504	77.597				
12	.727	2.078	79.675				
13	.675	1.927	81.603				
14	.616	1.761	83.363				
15	.592	1.691	85.054				
16	.519	1.482	86.536				
17	.463	1.324	87.860				
18	.447	1.278	89.137				
19	.418	1.194	90.331				
20	.403	1.152	91.483				
21	.371	1.059	92.541				
22	.321	.917	93.458				
23	.296	.847	94.305				
24	.266	.760	95.065				

25	.244	.698	95.763				
26	.230	.658	96.421				
27	.197	.564	96.985				
28	.194	.554	97.539				
29	.172	.492	98.031				
30	.160	.459	98.490				
31	.150	.428	98.917				
32	.124	.355	99.272				
33	.117	.336	99.608				
34	.101	.289	99.896				
35	.036	.104	100.000				
Extraction Method: Principal Component Analysis.							
a. When components are correlated, sums of squared loadings cannot be added to obtain a total variance.							

Table F-12: Total Variance Explained with 4 Factors (Change Agile Practices for Security)

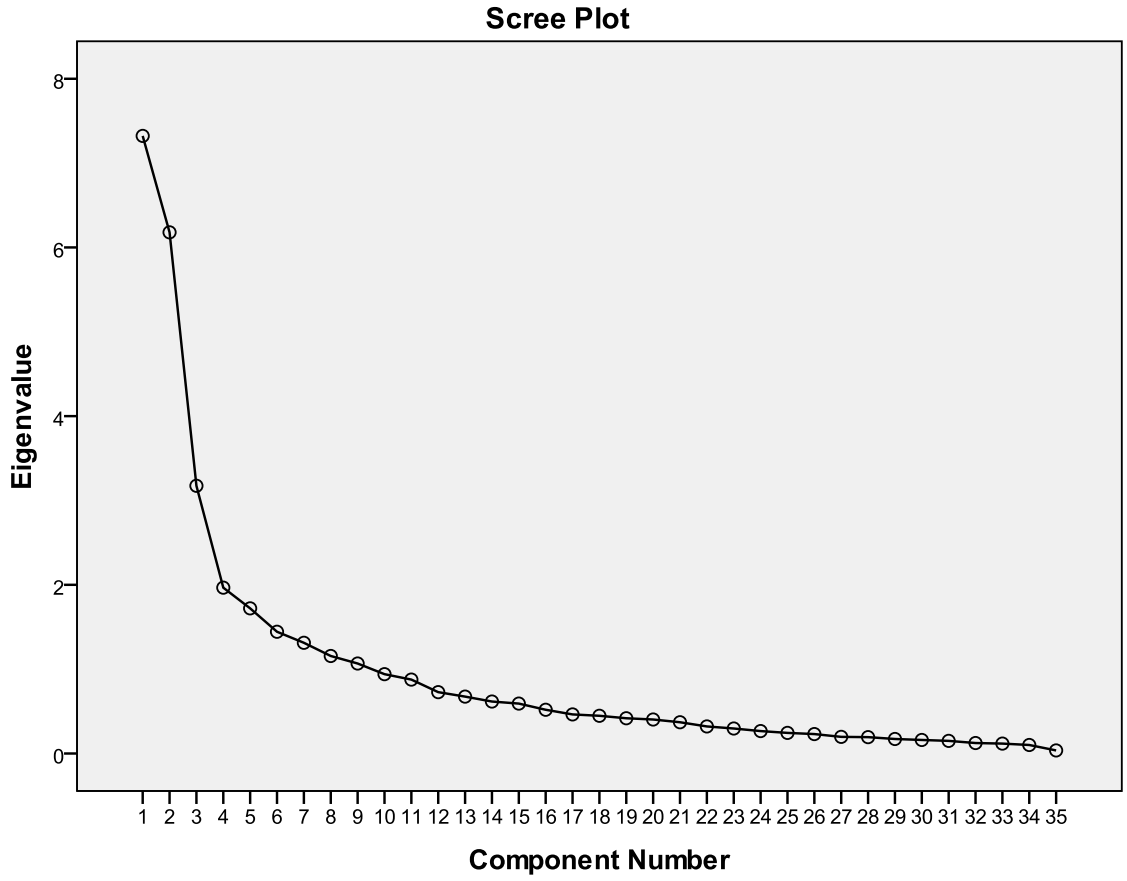


Figure F-4: Scree Plot (Change Agile Practices for Security)

Pattern Matrix[a]				
	Component			
	1	2	3	4
WaterVs.Agile2	-.775			
WaterVs.Agile3	-.757			
AccSch10	.739			
WaterVs.Agile1	-.702			
AccSch9	.698	.315		
AccSch7	.646	.321		.329
AccSch8	.633	.301		
InternalAgile2	.606			-.405
InternalAgile3	.538			-.302
InternalAgile1	.440	-.332	-.348	
AwareOfSec10		.673		
AwareOfSec4		.623		
AwareOfSec1		.568		-.374
AwareOfSec8		.567		
AwareOfSec6		.540		-.357
AwareOfSec9	.354	.536		
AwareOfSec7	.332	.526		
AwareOfSec2		.520		
AccSch5		.419	-.339	
AccSch6	.336	.407		
AwareOfSec13			-.862	
AwareOfSec11			-.823	
AwareOfSec14			-.792	
InternalAgile4			-.701	
InternalAgile5			-.510	
AwareOfSec3		.305	-.419	
WaterVs.Agile7				-.803
WaterVs.Agile6				-.758
WaterVs.Agile4				-.729
WaterVs.Agile8				-.681
WaterVs.Agile9				-.587
AccSch4				.576

AwareOfSec5		.493		-.505
AccSch1	.382			.446
WaterVs.Agile5				
Extraction Method: Principal Component Analysis. Rotation Method: Oblimin with Kaiser Normalization.				
a. Rotation converged in 85 iterations.				

Table F-13: Pattern Matrix of Oblimin Rotation with 4 Factors (Change Agile Practices for Security)

Structure Matrix				
	Component			
	1	2	3	4
AccSch10	.818	.372		
AccSch9	.782	.429		
WaterVs.Agile2	-.769			
AccSch7	.760	.404		.371
AccSch8	.729	.390		.303
WaterVs.Agile1	-.714			-.372
WaterVs.Agile3	-.686			
InternalAgile2	.563			-.352
InternalAgile3	.525		-.324	
AwareOfSec10	.315	.694		
AwareOfSec4		.622		-.375
AwareOfSec8		.596		-.333
AwareOfSec9	.455	.596		
AwareOfSec1		.587		-.425
AwareOfSec6		.565		-.423
AwareOfSec7	.464	.559		
AwareOfSec2	.341	.554		
AccSch6	.419	.474		
AccSch5		.438	-.344	
AwareOfSec13			-.855	
AwareOfSec11			-.804	
AwareOfSec14			-.794	
InternalAgile4			-.675	
InternalAgile5			-.576	-.335
AwareOfSec3		.323	-.461	-.313
InternalAgile1	.351		-.414	

WaterVs.Agile7				-.841
WaterVs.Agile6				-.772
WaterVs.Agile8			-.338	-.756
WaterVs.Agile4				-.693
WaterVs.Agile9	-.347			-.677
AccSch4	.332		.323	.617
AwareOfSec5		.527		-.550
AccSch1	.472		.313	.532
WaterVs.Agile5				
Extraction Method: Principal Component Analysis. Rotation Method: Oblimin with Kaiser Normalization.				

TableF-14: Structure Matrix of Oblimin Rotation with 4 Factors (Change Agile Practices for Security)

Appendix G: Questionnaire Field Testing

RQ1-Combining Security & Agility

RQ1A Dedicated Security Engineer

No	Statement		Comfortable	Somewhat Comfortable	Neutral	Somewhat Uncomfortable	Uncomfortable	Mean	Dropped
Having a Security Engineer within the agile team can									
1	Increase each team member's security knowledge	Freq.	2	3	0	0	0	4.40	No
		%	40.0	60.0	0	0	0		
2	Help in producing more secure software	Freq.	0	3	2	0	0	3.60	No
		%	0	60.0	40.0	0	0		
3	Help solve security issues faster	Freq.	0	4	0	1	0	3.60	No
		%	0	80.0	0	20.0	0		
4	Increase awareness of security issues	Freq.	1	4	0	0	0	4.20	No
		%	20.0	80.0	0	0	0		
5	Uncover security issues earlier in the project	Freq.	0	5	0	0	0	4.0	No
		%	0	100	0	0	0		
6	Help the customer make better security decisions	Freq.	0	1	3	1	0	3.0	No
		%	0	20.0	60.0	20.0	0		
7	Lengthen the iteration	Freq.	1	2	2	0	0	3.80	No
		%	20.0	40.0	40.0	0	0		
8	Help Establish/Increase the overall security assurance of Agile	Freq.	1	2	2	0	0	3.80	No
		%	20.0	40.0	40.0	0	0		
9	**Be better for less mature organizations' security needs**	**Freq.**	**0**	**1**	**2**	**2**	**0**	**2.8**	**Yes**
		%	**0**	**20.0**	**40.0**	**40.0**	**0**		
10	Help the project achieve compliance with security standards faster	Freq.	0	3	2	0	0	3.60	No
		%	0	60.0	40.0	0	0		
11	Lower the security risk of the produced software further than developers can	Freq.	0	4	1	0	0	3.80	No
		%	0	80.0	20.0	0	0		
12	Decrease the need for highly experienced developers/architects within the team	Freq.	0	1	3	1	0	3.0	No
		%	0	20.0	60.0	20.0	0		
13	Alleviate other team members from having to focus on specific security issues	Freq.	1	1	1	1	1	3.0	No
		%	20.0	20.0	20.0	20.0	20.0		
14	Maximize the precision of security requirement assessment	Freq.	0	3	2	0	0	3.60	No
		%	0	60.0	40.0	0	0		
15	Put more importance on security for the project and the team	Freq.	1	2	2	0	0	3.80	No
		%	20.0	40.0	40.0	0	0		
16	Mitigate the lack of mature security vulnerability assessment tools that are not yet available	Freq.	1	1	2	1	0	3.40	No
		%	20.0	20.0	40.0	20.0	0		
17	Allow the project to benefit from the security engineer as an independent on-site security auditor for the project if needed	Freq.	1	1	2	0	1	3.20	No
		%	20.0	20.0	40.0	0	20.0		

Table G-1: Field Testing Results for 'dedicated security engineer' items

RQ1B Software with Security in Mind

No	Statement		Comfortable	Somewhat Comfortable	Neutral	Somewhat Uncomfortable	Uncomfortable	Mean	Dropped
Building Software with Security in mind in agile projects can									
18	Establish/Increase the Security Assurance for the project	Freq.	1	2	1	1	0	3.60	No
		%	20.0	40.0	20.0	20.0	0		
19	Help in writing more secure code	Freq.	1	3	0	1	0	3.80	No
		%	20.0	60.0	0	20.0	0		
20	Help in achieving security focused testing	Freq.	1	2	2	0	0	3.80	No
		%	20.0	40.0	40.0	0	0		
21	Expand the awareness of security throughout the project	Freq.	1	3	1	0	0	4.0	No
		%	20.0	60.0	20.0	0	0		
22	Put more overall focus on security	Freq.	0	4	1	0	0	3.80	No
		%	0	80.0	20.0	0	0		
23	Promote the collective ownership of security by all stakeholders	Freq.	1	1	3	0	0	3.60	No
		%	20.0	20.0	60.0	0	0		
24	Help in decreasing vulnerabilities in the software	Freq.	0	5	0	0	0	4.0	No
		%	0	100	0	0	0		
25	Reduce security risk to the project and the resulting software	Freq.	1	4	0	0	0	4.20	No
		%	20.0	80.0	0	0	0		

Table G-2: Field Testing Results for ‘software with security in mind’ items

RQ1C Security Controls

No	Statement		Comfortable	Somewhat Comfortable	Neutral	Somewhat Uncomfortable	Uncomfortable	Mean	Dropped
Using Software Security Controls in the agile project can									
26	Alleviate developers from having to focus on security	Freq.	1	0	3	1	0	3.20	No
		%	20.0	0	60.0	20.0	0		
27	Be beneficial throughout the project	Freq.	1	1	3	0	0	3.60	No
		%	20.0	20.0	60.0	0	0		
28	Be reused on multiple projects	Freq.	1	1	3	0	0	3.60	No
		%	20.0	20.0	60.0	0	0		
29	Become a part of the framework/infrastructure	Freq.	1	1	1	2	0	3.20	No
		%	20.0	20.0	20.0	40.0	0		
30	Be available as part of the infrastructure before the next project begins	Freq.	0	2	2	1	0	3.20	No
		%	0	40.0	40.0	20.0	0		
31	Reduce costs if used on multiple projects	Freq.	1	2	1	1	0	3.60	No
		%	20.0	40.0	20.0	20.0	0		
32	Help the resulting software to have reduced weaknesses/vulnerabilities	Freq.	1	1	3	0	0	3.60	No
		%	20.0	20.0	60.0	0	0		
33	Reduce reliance on individuals within the team for security expertise	Freq.	1	0	3	1	0	3.20	No
		%	20.0	0	60.0	20.0	0		
34	**Be the best way to increase the security of the software**	**Freq.**	**1**	**0**	**1**	**2**	**1**	**2.60**	**Yes**
		%	**20.0**	**0**	**20.0**	**40.0**	**20.0**		

Table G-3: Field Testing Results for ‘security controls’ items

RQ1F Experience of Developers

No	Statement		Comfortable	Somewhat Comfortable	Neutral	Somewhat Uncomfortable	Uncomfortable	Mean	Dropped
Experienced Developers within the agile team can									
35	Create/write more secure user stories	Freq.	1	3	1	0	0	4.0	No
		%	20.0	60.0	20.0	0	0		
36	Discover more security issues/vulnerabilities/bugs than regular developers	Freq.	1	4	0	0	0	4.20	No
		%	20.0	80.0	0	0	0		
37	Help implement security requirements faster	Freq.	0	5	0	0	0	4.0	No
		%	0	100	0	0	0		
38	Add to the team's overall security awareness	Freq.	0	4	1	0	0	3.80	No
		%	0	80.0	20.0	0	0		
39	Mitigate more security issues than regular developers	Freq.	1	4	0	0	0	4.20	No
		%	20.0	80.0	0	0	0		
40	Be more effective than using tools on security	Freq.	0	2	2	1	0	3.20	No
		%	0	40.0	40.0	20.0	0		
41	Increase the overall security of the software	Freq.	1	3	1	0	0	4.0	No
		%	20.0	60.0	20.0	0	0		
42	Be more beneficial than QA when it comes to security focused testing	Freq.	1	2	2	0	0	3.80	No
		%	20.0	40.0	40.0	0	0		
43	Play a key role in avoiding to introduce security vulnerabilities into software	Freq.	1	2	2	0	0	3.80	No
		%	20.0	40.0	40.0	0	0		
44	Predict security vulnerabilities at earlier stages of the project	Freq.	1	2	2	0	0	3.80	No
		%	20.0	40.0	40.0	0	0		

Table G-4: Field Testing Results for 'experience of developers' items

RQ2-Change Agile Practices for Security

RQ2A Agile vs. Traditional Waterfall

No	Statement		Comfortable	Somewhat Comfortable	Neutral	Somewhat Uncomfortable	Uncomfortable	Mean	Dropped
Compared to Traditional Waterfall, Agile methodologies									
45	Impact security more negatively overall	Freq.	1	0	2	2	0	3.0	No
		%	20.0	0	40.0	40.0	0		
46	Do less on security	Freq.	1	2	0	2	0	3.40	No
		%	20.0	40.0	0	40.0	0		
47	Do not deal with security directly	Freq.	1	1	1	2	0	3.20	No
		%	20.0	20.0	20.0	40.0	0		
48	Produce more secure software overall	Freq.	1	0	3	1	0	3.20	No
		%	20.0	0	60.0	20.0	0		
49	Perform equally overall in terms of security	Freq.	3	1	0	1	0	4.20	No
		%	60.0	20.0	0	20.0	0		
50	Are more secure because of testing	Freq.	1	1	1	1	1	3.0	No
		%	20.0	20.0	20.0	20.0	20.0		
51	Provide team members more opportunities to mitigate security issues	Freq.	1	2	2	0	0	3.80	No
		%	20.0	40.0	40.0	0	0		
52	**Are less secure because of less upfront design**	**Freq.**	**1**	**1**	**0**	**1**	**2**	**2.60**	**Yes**
		%	**20.0**	**20.0**	**0**	**20.0**	**40.0**		
53	Are better on security because of their iterative nature	Freq.	0	3	0	2	0	3.20	No
		%	0	60.0	0	40.0	0		
54	Can discover security related issues faster	Freq.	1	4	0	0	0	4.20	No
		%	20.0	80.0	0	0	0		

Table G-5: Field Testing Results for ‘Agile vs. Waterfall’ items

RQ2B Awareness of Security to Agile

No	Statement		Comfortable	Somewhat Comfortable	Neutral	Somewhat Uncomfortable	Uncomfortable	Mean	Dropped
Awareness of security in Agile									
55	Allows for increased consideration of security throughout the project	Freq.	1	2	1	1	0	3.60	No
		%	20.0	40.0	20.0	20.0	0		
56	Requires elaboration phases in the project	Freq.	0	2	2	1	0	3.20	No
		%	0	40.0	40.0	20.0	0		
57	Contributes more to security for the project than training can	Freq.	1	0	4	0	0	3.40	No
		%	20.0	0	80.0	0	0		
58	Allows team members to voice their security concerns openly	Freq.	1	2	2	0	0	3.80	No
		%	20.0	40.0	40.0	0	0		
59	Reduces security risks overall	Freq.	1	2	2	0	0	3.80	No
		%	20.0	40.0	40.0	0	0		
60	Adds more focus/emphasis on security within the project	Freq.	1	2	2	0	0	3.80	No
		%	20.0	40.0	40.0	0	0		
61	Should include upfront planning for security specific issues	Freq.	2	1	2	0	0	4.0	No
		%	40.0	20.0	40.0	0	0		
62	Makes stakeholders more considerate of potential security issues	Freq.	1	1	3	0	0	3.60	No
		%	20.0	20.0	60.0	0	0		
63	Requires organizational maturity	Freq.	0	1	3	1	0	3.0	No
		%	0	20.0	60.0	20.0	0		
64	Requires organizational support	Freq.	1	0	4	0	0	3.40	No
		%	41.9	21.0	11.3	1.6	2.4		
65	Should be required in certain projects only	Freq.	2	0	2	0	1	3.40	No
		%	40.0	0	40.0	0	20.0		
66	Is important to all projects overall	Freq.	2	0	2	1	0	3.60	No
		%	40.0	0	40.0	20.0	0		
67	Is important to some projects only	Freq.	1	1	2	0	1	3.20	No
		%	20.0	20.0	40.0	0	20.0		
68	Is only important if security is required for the project	Freq.	2	0	3	0	0	3.80	No
		%	40.0	0	60.0	0	0		

Table G-6: Field Testing Results for ‘awareness of security to Agile’ items

RQ2C Impact of Accelerated Schedule

No	Statement		Comfortable	Somewhat Comfortable	Neutral	Somewhat Uncomfortable	Uncomfortable	Mean	Dropped
Accelerated Schedules of Agile Projects									
69	Does not hinder security	Freq.	1	2	2	0	0	3.8	No
		%	20.0	40.0	40.0	0	0		
70	Affect security less than additional requirements security do	Freq.	2	1	2	0	0	4.0	No
		%	40.0	20.0	40.0	0	0		
71	Affect security less than more functionality does	Freq.	2	0	2	1	0	3.60	No
		%	40.0	0	40.0	20.0	0		
72	Do not affect security substantially	Freq.	1	1	2	1	0	3.40	No
		%	20.0	20.0	40.0	20.0	0		
73	Are affected by security requirements	Freq.	1	0	3	1	0	3.20	No
		%	20.0	0	60.0	20.0	0		
74	Affect every aspect of the project including security	Freq.	2	2	1	0	0	4.20	No
		%	40.0	40.0	20.0	0	0		
75	Affect security practices negatively	Freq.	1	2	2	0	0	3.80	No
		%	20.0	40.0	40.0	0	0		
76	Increases the overall risk including security risk to the project	Freq.	1	2	1	1	0	3.60	No
		%	20.0	40.0	20.0	20.0	0		
77	Put major focus on achieving functionality and not much else	Freq.	1	2	2	0	0	3.80	No
		%	20.0	40.0	40.0	0	0		
78	Contributes to less secure software	Freq.	1	2	1	1	0	3.60	No
		%	20.0	40.0	20.0	20.0	0		

Table G-7: Field Testing Results for ‘impact of accelerated schedule’ items

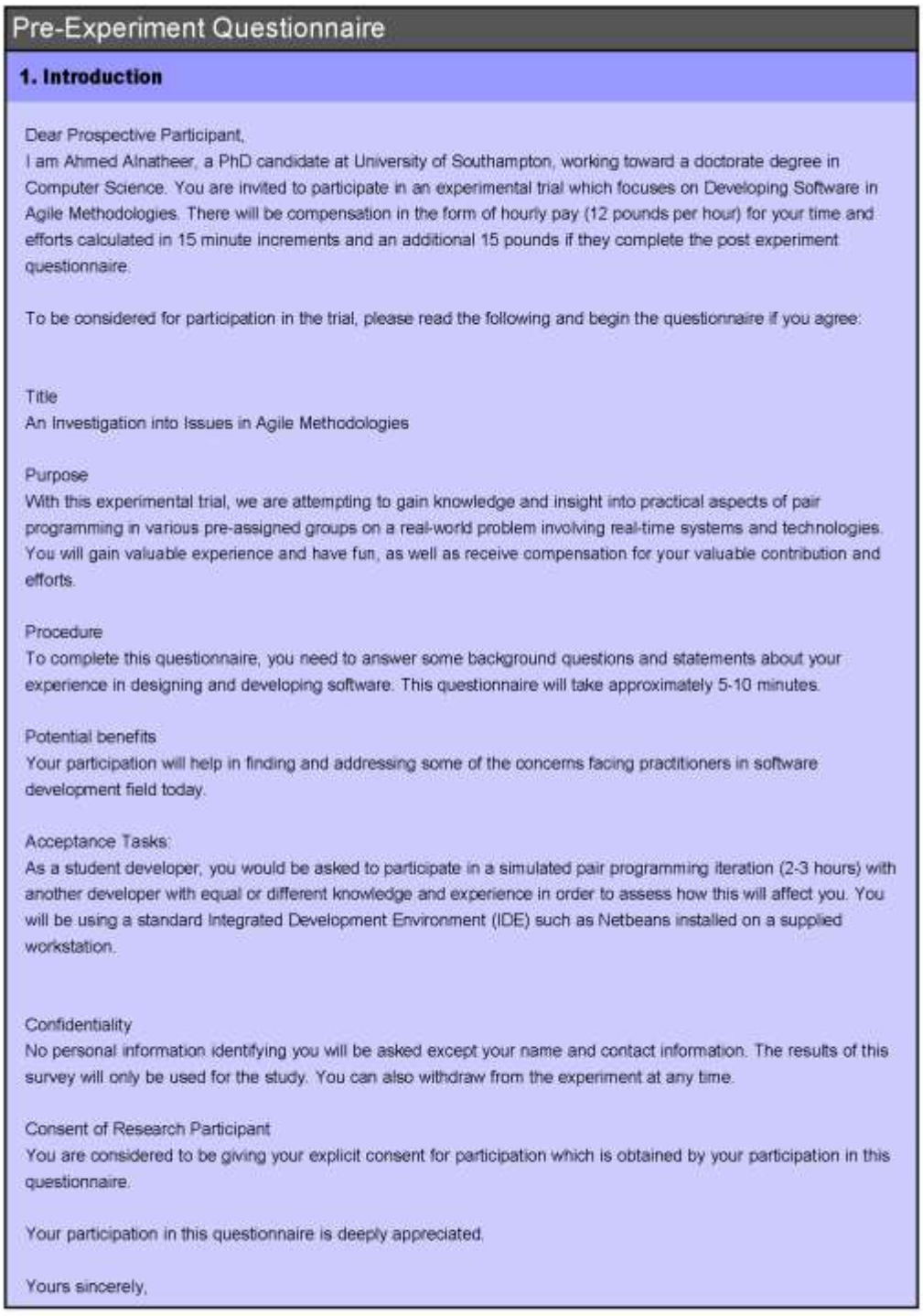
RQ2D Reduced Security for Internal Projects

No	Statement		Comfortable	Somewhat Comfortable	Neutral	Somewhat Uncomfortable	Uncomfortable	Mean	Dropped
Internal Agile Projects									
79	Can rely on the infrastructure for security	Freq.	0	3	1	1	0	3.40	No
		%	0	60.0	20.0	20.0	0		
80	Are less concerned with security	Freq.	2	1	1	1	0	3.80	No
		%	40.0	20.0	20.0	20.0	0		
81	Typically have less security requirements	Freq.	2	1	1	1	0	3.80	No
		%	40.0	20.0	20.0	20.0	0		
82	Do not need security in some cases	Freq.	2	1	1	0	1	3.60	No
		%	40.0	20.0	20.0	0	20.0		
83	Can have less security if they're not exposed publically	Freq.	1	2	1	0	1	3.40	No
		%	20.0	40.0	20.0	0	20.0		

Table G-8: Field Testing Results for 'reduced security for internal projects' items

Appendix H: Experimental Trial

Pre-Experiment Questionnaire



Pre-Experiment Questionnaire

1. Introduction

Dear Prospective Participant,
I am Ahmed Alnatheer, a PhD candidate at University of Southampton, working toward a doctorate degree in Computer Science. You are invited to participate in an experimental trial which focuses on Developing Software in Agile Methodologies. There will be compensation in the form of hourly pay (12 pounds per hour) for your time and efforts calculated in 15 minute increments and an additional 15 pounds if they complete the post experiment questionnaire.

To be considered for participation in the trial, please read the following and begin the questionnaire if you agree:

Title
An Investigation into Issues in Agile Methodologies

Purpose
With this experimental trial, we are attempting to gain knowledge and insight into practical aspects of pair programming in various pre-assigned groups on a real-world problem involving real-time systems and technologies. You will gain valuable experience and have fun, as well as receive compensation for your valuable contribution and efforts.

Procedure
To complete this questionnaire, you need to answer some background questions and statements about your experience in designing and developing software. This questionnaire will take approximately 5-10 minutes.

Potential benefits
Your participation will help in finding and addressing some of the concerns facing practitioners in software development field today.

Acceptance Tasks:
As a student developer, you would be asked to participate in a simulated pair programming iteration (2-3 hours) with another developer with equal or different knowledge and experience in order to assess how this will affect you. You will be using a standard Integrated Development Environment (IDE) such as Netbeans installed on a supplied workstation.

Confidentiality
No personal information identifying you will be asked except your name and contact information. The results of this survey will only be used for the study. You can also withdraw from the experiment at any time.

Consent of Research Participant
You are considered to be giving your explicit consent for participation which is obtained by your participation in this questionnaire.

Your participation in this questionnaire is deeply appreciated.

Yours sincerely,

Page 1

Pre-Experiment Questionnaire

Ahmed Alnatheer
PhD Candidate
School of Electronics and Computer Science
University of Southampton, UK
Phone: +44 (0)2380-581965
aaa1v09@ecs.soton.ac.uk

2. Background Questions

***1. Personal Information**

Name

University Email Address

Phone Number

***2. In which academic level?**

- ◯ Bachelor
- ◯ Master
- ◯ PhD

***3. In which academic year of your current degree?**

- ◯ 1st year
- ◯ 2nd year
- ◯ 3rd year
- ◯ 4th year

3. Knowledge

***1. How proficient are you in the following...**

Object Oriented Programming Languages:

	I am an expert	I am proficient (skilled)	I am comfortable with it	I know very little of it	I have no knowledge of it
Java	◯	◯	◯	◯	◯
C#	◯	◯	◯	◯	◯
Ruby	◯	◯	◯	◯	◯
PHP	◯	◯	◯	◯	◯

***2. Web Design:**

	I am an expert	I am proficient (skilled)	I am comfortable with it	I know very little of it	I have no knowledge of it
HTML	◯	◯	◯	◯	◯
JavaScript	◯	◯	◯	◯	◯

Pre-Experiment Questionnaire

*** 3. Web Development:**

	I am an expert	I am proficient (skilled)	I am comfortable with it	I know very little of it	I have no knowledge of it
Servlet/JSP	○	○	○	○	○
ASP/ASP.Net	○	○	○	○	○
PHP	○	○	○	○	○
RubyOnRails	○	○	○	○	○

*** 4. IDEs:**

	I am an expert	I am proficient (skilled)	I am comfortable with it	I know very little of it	I have no knowledge of it
Eclipse/Netbeans	○	○	○	○	○
Visual Studio	○	○	○	○	○

*** 5. Databases:**

	I am an expert	I am proficient (skilled)	I am comfortable with it	I know very little of it	I have no knowledge of it
MySQL	○	○	○	○	○
MS Access	○	○	○	○	○

*** 6. Software Development Process:**

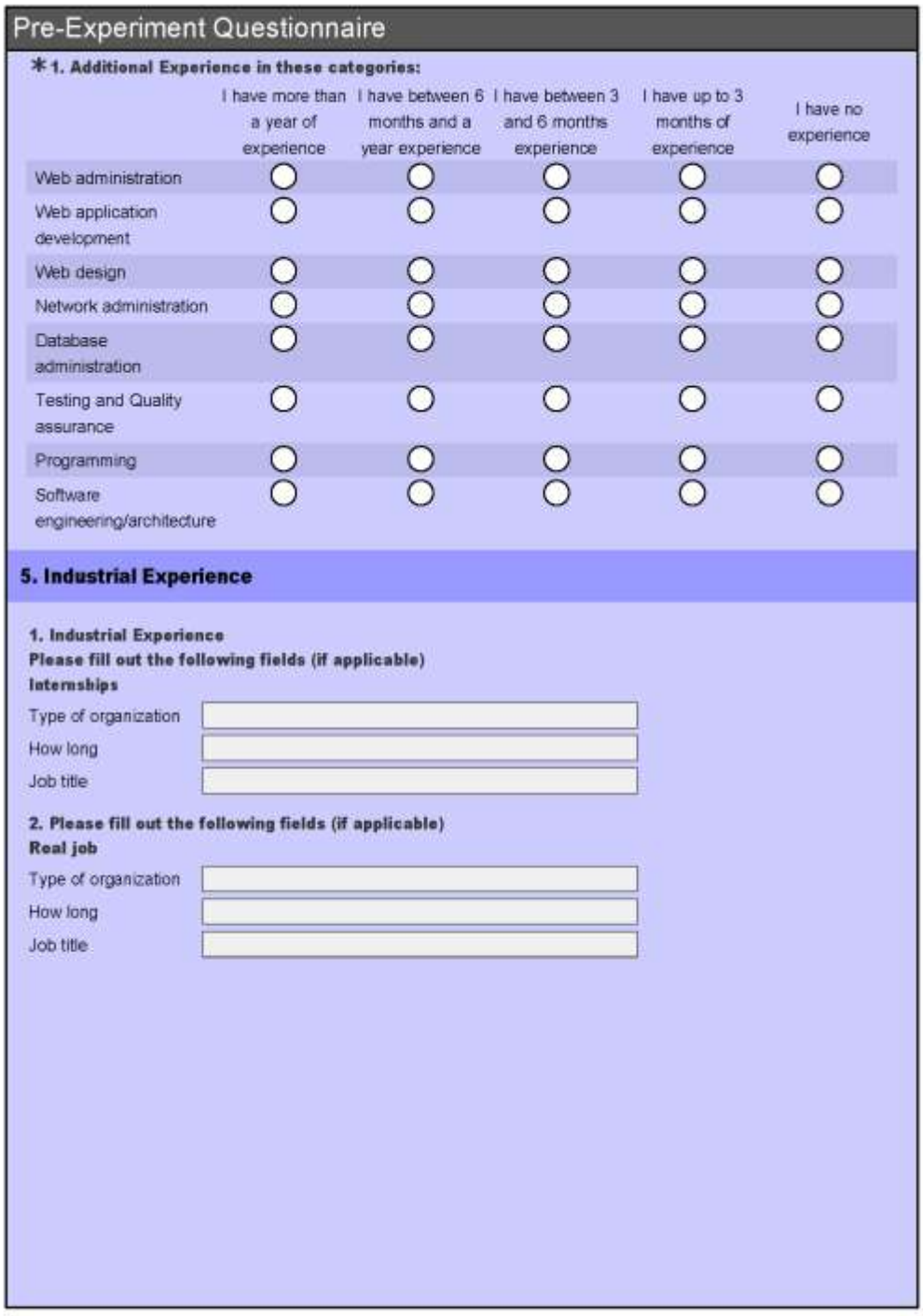
	I am an expert	I am proficient (skilled)	I am comfortable with it	I know very little of it	I have no knowledge of it
Waterfall	○	○	○	○	○
XP	○	○	○	○	○
Scrum	○	○	○	○	○
TDD	○	○	○	○	○

*** 7. Security related topics:**

	I am an expert	I am proficient (skilled)	I am comfortable with it	I know very little of it	I have no knowledge of it
Cryptography	○	○	○	○	○
Network Security	○	○	○	○	○

4. Experience

Page 3



Pre-Experiment Questionnaire

*1. Additional Experience in these categories:

	I have more than a year of experience	I have between 6 months and a year experience	I have between 3 and 6 months experience	I have up to 3 months of experience	I have no experience
Web administration	○	○	○	○	○
Web application development	○	○	○	○	○
Web design	○	○	○	○	○
Network administration	○	○	○	○	○
Database administration	○	○	○	○	○
Testing and Quality assurance	○	○	○	○	○
Programming	○	○	○	○	○
Software engineering/architecture	○	○	○	○	○

5. Industrial Experience

1. Industrial Experience
Please fill out the following fields (if applicable)
Internships

Type of organization
How long
Job title

2. Please fill out the following fields (if applicable)
Real job

Type of organization
How long
Job title

Page 4

Figure H-1: Pre-Experiment Questionnaire

Experimental Instructions

Experimental Instructions:
Client Side and Server-Side Form Processing

1. Objectives

Your group will be working on developing an online payment form that attempts to take credit card information from a fictitious customer through a payment form programmed in HTML and Javascript on the client side, as well as Java and JSP on the Server side. The tool that you will use to make this software is the NetBeans Integrated Development Environment which many Agile teams typically use to develop web-based applications.

2. Requirements

In this experiment, you are required to create a web based payment form, which will provide to a visitor the ability to submit their payment information. When a user inputs his/her information, your script will operate on this information and will then send it to the server which will subsequently process the data and respond appropriately informing the user of the result.

You are provided with a baseline project containing the structure of the form to be submitted and the necessary tools to be able to accomplish your task (see Figure 1). When "Submit" button is clicked, your web page (client side) will invoke a server-side script in Java that will retrieve the following information

- Credit Info
  - Name of card holder
  - Card Type (Visa, Master Card only)
  - Card number
  - Expiration month
  - Expiration Year

The server side must ensure that the information submitted is correct and will respond to each form submission appropriately taking the input formats and possible input variations into account.

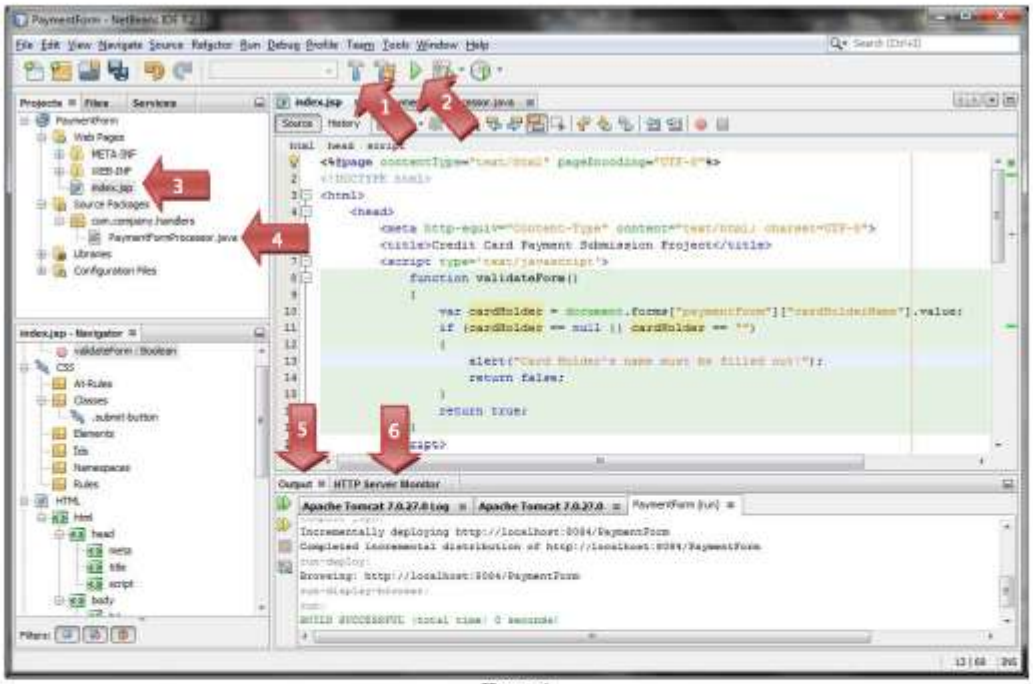


Figure 1

1. **Build Project Button:** Will build the project and generate a .war (web archive) file that can be deployed to the web server
2. **Run Project Button:** Will deploy the generated .war file into the web server and initiate the browser to point to the index.jsp page
3. **Index.jsp source file:** Contains the client side code consisting of the HTML form and the Javascript code
4. **PaymentFormProcessor source file:** Contains the server side Java code that is responsible for form handling
5. **The output window:** The widget that shows the output from the various pieces of the running application. Any standard output will be shown here!
6. **Http Server Monitor:** Details of any communication between the client browser and the server (apache tomcat) will be shown under this window.

Figure H-2: Experimental Instructions

Post-Experiment Questionnaire

Post-Experiment Questionnaire

1.

*1. Team Information

Group Number

Team Member Number

*2. Please answer with a number for the following questions. If you don't remember exactly just give an approximate number.

How many times did you as an individual ...

Switch your places as the observer and driver?

Discover bugs and/or vulnerabilities during development/testing?

Fix bugs and/or vulnerabilities during development/testing?

*3. Please answer with a number for the following questions. If you don't remember exactly just give an approximate number.

As a team, how many times you and your teammate...

Encounter issues and/or vulnerabilities that you did not get to fix?

Discuss issues regarding Security

Discuss issues regarding other aspects of Quality

Use outside sources to obtain security information in order to complete the task

Page 1

Post-Experiment Questionnaire

*** 4. Did you and your teammate discuss or consider...?**

	Yes	No	Don't Understand the question
Client side basic Validation (non-null value, non empty field, etc.)	○	○	○
Server side basic validation (non-null value, non empty field, etc.)	○	○	○
Client side Regular expression validation (input formatting, range validation, etc.)	○	○	○
Server side Regular expression validation (string formatting, range validation, etc.)	○	○	○
Combination of client side and server side validation	○	○	○
Encryption (HTTPS, etc.)	○	○	○
Invalid data submission (based on checking the content of the data itself)	○	○	○
Cross site scripting	○	○	○
Cross site request forgery	○	○	○

2.

*** 1. Would you say that your awareness of security increased as a result of this work in terms of...**

	Strongly Agree	Agree	Somewhat Agree	Neutral	Somewhat Disagree	Disagree	Strongly Disagree
Design	○	○	○	○	○	○	○
Development/Testing	○	○	○	○	○	○	○

*** 2. Would you say that your confidence in your ability to produce software increased as a result of this experiment in terms of...**

	Strongly Agree	Agree	Somewhat Agree	Neutral	Somewhat Disagree	Disagree	Strongly Disagree
Security	○	○	○	○	○	○	○
Other aspects of software Quality	○	○	○	○	○	○	○

*** 3. An important consideration for my team was...**

	Strongly Agree	Agree	Somewhat Agree	Neutral	Somewhat Disagree	Disagree	Strongly Disagree
Development/Testing the software?	○	○	○	○	○	○	○
Protecting the software against bugs/vulnerabilities?	○	○	○	○	○	○	○

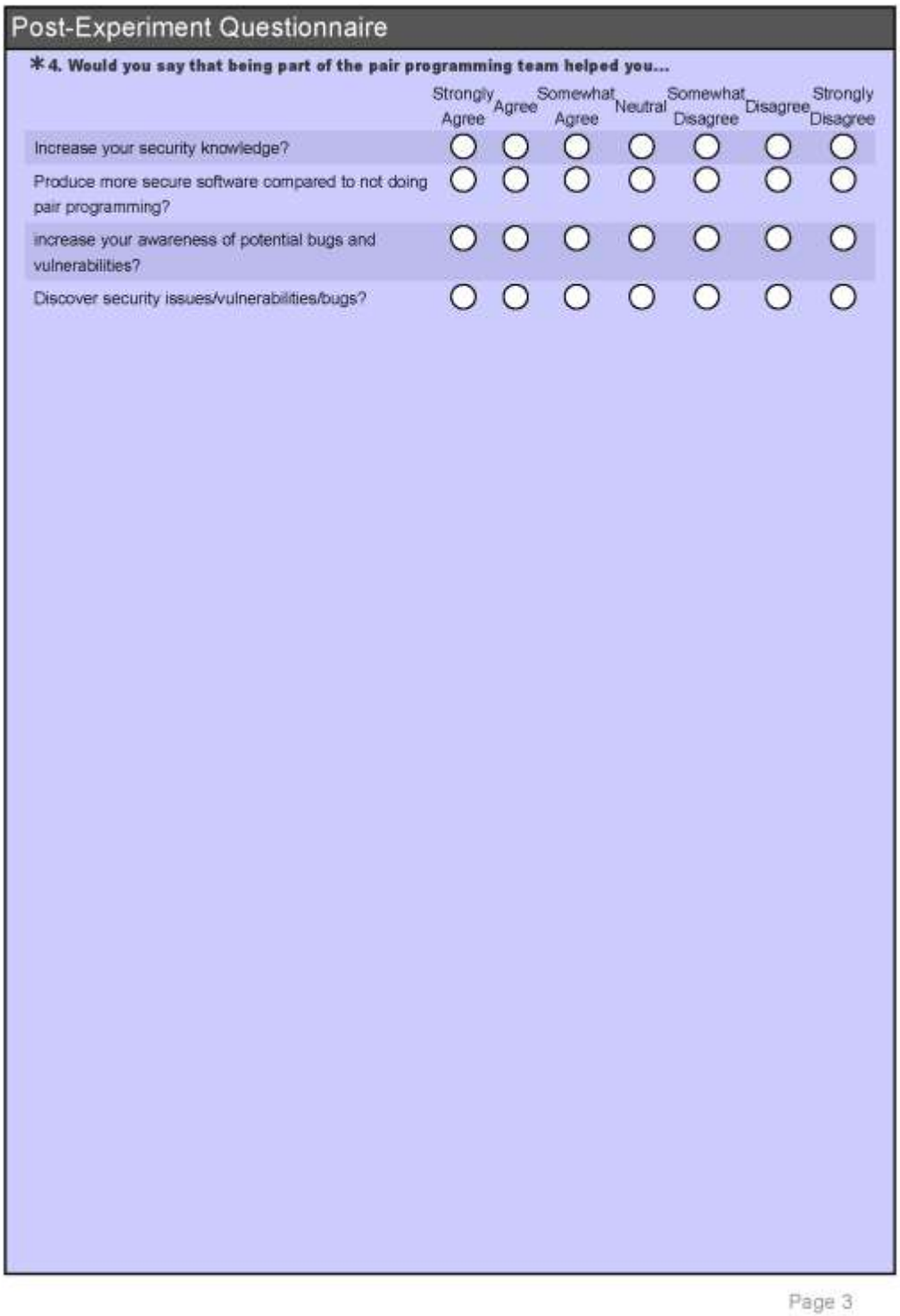


Figure H-3: Post-Experiment Questionnaire