WATERFORD INSTITUTE OF TECHNOLOGY

MASTERS THESIS

Small Scale Scrum: A Framework for Successful Implementation of the Scrum Methodology for Small Sized Teams

Author:
Agnieszka GANCARCZYK
(20060828)

Supervisor:
Mr. Eamonn DE LEASTAR
(with Dr. Leigh GRIFFIN)

A thesis submitted in fulfillment of the requirements for the degree of MSc in Computing (Communications Software)

in the

Department of Computing and Mathematics

September 3, 2018

Declaration of Authorship

I, Agnieszka GANCARCZYK (20060828), declare that this thesis titled, "Small Scale Scrum:
A Framework for Successful Implementation of the Scrum Methodology for Small Sized Teams" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at Waterford Institute of Technology.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at Waterford Institute of Technology 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.

Signed: _____

Date: _____

WATERFORD INSTITUTE OF TECHNOLOGY

Abstract

Department of Computing and Mathematics

MSc in Computing (Communications Software)

Small Scale Scrum:
A Framework for Successful Implementation of the Scrum Methodology for Small Sized Teams

by Agnieszka GANCARCZYK (20060828)

Growing global interest in Agile and in particular in Scrum and Hybrid Scrum forces teams to adopt new techniques for rapidly developing and delivering high quality software solutions. Red Hat follows suit and is very interested in the wide adoption of Agile. Red Hat Mobile Engineering team have already had a great success in the Agile transformation in their six to eight person teams, with team members occupying different, well-defined roles. Now, Red Hat Consulting and their customers are transitioning into Agile as part of a short internal project with bounded budget to address communication, quality and business challenges facing their small scale, distributed teams, with team members occupying multiple roles in the projects. Red Hat Consulting is one of thousand's of companies in the consultancy sphere with this challenge. In this research thesis, Small Scale Agile Manifesto and Small Scale Scrum principles, guidelines and recommendations, seen as a Hybrid variation of traditional Scrum are explored to overcome obstacles observed in consulting engagements.

Acknowledgements

I would first like to extend my thanks to my supervisors, Mr. Eamonn de Leastar and Dr. Leigh Griffin for their guidance and support throughout my MSc. In particular to Leigh, for his flexibility and the hours of guidance, motivation and conversations in this final part of my MSc.

I would also like to thank all the colleagues at Red Hat for their input along the way.

I would also like to sincerely thank the management and administration at TSSG for funding this course.

This thesis would not have been possible without substantial education received from Liceum Ogólnokształcące im. Stanisława Konarskiego in Oświęcim and fostered within Carlow Institute of Technology and Waterford Institute of Technology.

To my mother, Mieczysława, thank you for your support all my life, for going to unimaginable lengths to help me in difficult times, for guiding and motivating me to pursue my education to this level. To my sister, Diana and her family, thank you for your love, support and advise - I would not be able to finalize this thesis without your help. Finally, to my husband Michał, whom I have known for thirteen years and who has always been there for me. Words cannot express my feelings, nor my thanks. You have my deepest gratitude and love forever.

Contents

Declaration of Authorship i

Abstract ii

Acknowledgements iii

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Hypothesis . . . 2
- 1.3 Research Questions . . . 3
- 1.4 Contributions . . . 4
- 1.5 Thesis Outline . . . 5

2 State of the Art 7
- 2.1 Agile Manifesto, Large Scale Scrum and Large Teams . . . 7
- 2.2 Small Scale Scrum and Small Teams . . . 10
- 2.3 Barriers to Small Scale Scrum . . . 12
- 2.4 Research Methodologies . . . 12

3 Small Scale Scrum Principles 14

4 Small Scale Scrum Framework 17
- 4.1 Small Scale Agile Manifesto . . . 17
- 4.2 Small Scale Scrum . . . 18

5 Small Scale Scrum Survey & Projects Validation 25
- 5.1 Survey Rationale & Design . . . 25
- 5.2 Survey Results . . . 32
- 5.3 Post-Survey Interviews . . . 34
- 5.4 Projects Selection Process . . . 37
- 5.5 Projects Results . . . 37
  - 5.5.1 Small Scale Hybrid/Waterfall Project . . . 41
  - 5.5.2 Small Scale Scrum Experimental Project . . . 41
  - 5.5.3 Small Scale Scrum Validation Project . . . 42
  - 5.5.4 Projects Comparison - Statistics . . . 43

6 Conclusion and Future Work 46
- 6.1 Conclusion . . . 46
  - 6.1.1 Thesis Summary . . . 46
  - 6.1.2 Contributions . . . 47
  - 6.1.3 Conclusions . . . 48
- 6.2 Future Work . . . 51

7 References 53

A Questionnaire Survey **58**
A.1 General Questions . . . 58
A.2 Project Size . . . 58
A.3 Team Roles . . . 58
A.4 Agile & Scrum Principles & Ceremonies . . . 62
A.5 Software Requirements . . . 63
A.6 Software Development . . . 64
A.7 Software Testing . . . 66
A.8 Software Quality . . . 66
A.9 Project Management . . . 67
A.10 Communication . . . 68
A.11 Business . . . 69

B Questionnaire Results **71**

C Scrum Ceremonies, Principles and Roles **113**

D Barriers to Small Scale Scrum **115**

E Daily Scrum Questions **117**

List of Figures

3.1 Small Scale Scrum Principles 14

4.1 Small Scale Scrum Framework 20

B.1 Percentage of participants practicing Agility 71
B.2 Team size for Agile project 72
B.3 Teams distribution 72
B.4 Typical project duration/size 73
B.5 Involvement in more than 1 project/team 73
B.6 Percentage of projects involved in 74
B.7 Percentage of customer and sustaining/maintenance projects 74
B.8 Percentage of work estimated before development 75
B.9 Percentage of project extensions available 75
B.10 Percentage of tasks fully clarified before development 76
B.11 Percentage of participants asking for tasks clarification 76
B.12 Percentage of those who ask for tasks clarification 77
B.13 Percentage of times that customer clarifies requirements timely 77
B.14 Speed of customer's task clarifications 77
B.15 Percentage of participant's writing tests 78
B.16 Percentage of those who have code reviews 78
B.17 Percentage of those who adhere to best coding practices 79
B.18 Percentage of those who document their code 79
B.19 Percentage of those who add changes to code on initial code review and verification 80
B.20 Percentage of bugs introduction to codebase 80
B.21 Customer verification when task is in test 81
B.22 Percentage of those who ask for help/mentoring when task is complex 81
B.23 Scrum roles present in participant's project 82
B.24 Participant's primary role in the team 82
B.25 Percentage of participant's taking additional role(s) in project 83
B.26 Participant's additional role(s) in a typical project 83
B.27 Participant's skills to cover all additional roles in project 84
B.28 Possibility to up skill to perform additional roles 84
B.29 Difficulty levels for changing tasks on daily basis 85
B.30 Time loses when changing tasks daily 85
B.31 Percentage of participant's work completion on-time when fulfilling multiple roles in project 86
B.32 Length of participant's Agile practice 86
B.33 Agile methodologies followed by participants 87
B.34 Hybrid methodology followed 87
B.35 Scrum practices used by teams 88
B.36 Additional Scrum practices used by teams 88
B.37 Sprint duration across different size teams 89

B.38 Percentage of User Stories used by participants . . . 89
B.39 Percentage of estimating User Stories . . . 90
B.40 Units of measure for features used by participants . . . 90
B.41 Work estimation methods used . . . 91
B.42 Percentage of work items containing acceptance criteria . . . 91
B.43 Capturing acceptance criteria . . . 92
B.44 Involvement in software requirements creation, clarification etc. . . . 92
B.45 Scrum principles helping to reduce development complexity . . . 93
B.46 Results from follow up question . . . 93
B.47 Tools used for verifying Scrum reduction of development complexity . 94
B.48 Frequency of change requests in project . . . 94
B.49 Number of support cases to worked on . . . 95
B.50 Usage of 'Definition of Done' . . . 95
B.51 Level of adherence to 'Definition of Done' . . . 95
B.52 Reference of 'Definition of Done' to work items . . . 96
B.53 Use of 'Definition of Ready' . . . 96
B.54 Disagreement of elements of 'Definition of Done' for Feature . . . 97
B.55 Disagreement of elements of 'Definition of Done' for Feature . . . 97
B.56 Elements of 'Definition of Done' . . . 97
B.57 Disagreement of elements of 'Definition of Done' . . . 98
B.58 Disagreement of elements of 'Definition of Done' . . . 98
B.59 Use of 'Definition of Ready' . . . 98
B.60 Level of adherence to 'Definition of Ready' . . . 99
B.61 Disagreement of elements of 'Definition of Ready' . . . 99
B.62 Disagreement of elements of 'Definition of Ready' . . . 99
B.63 Use of testing . . . 100
B.64 Speed of work verification . . . 100
B.65 Speed of PR reviews . . . 100
B.66 Reasons for task completion . . . 101
B.67 Quality tools used . . . 101
B.68 Builds release within Sprint . . . 102
B.69 Assistance of Scrum in various aspects of quality . . . 102
B.70 Involvement of PM in project . . . 102
B.71 PM taking traditional Scrum roles . . . 103
B.72 Involvement of Manager/Senior Manager in project . . . 103
B.73 Manager/Senior Manager taking traditional Scrum roles . . . 104
B.74 Involvement of Architect in project . . . 104
B.75 Involvement of Sales team in project . . . 105
B.76 Involvement of DevOps in project . . . 105
B.77 Overall team collaboration . . . 106
B.78 Commonly used communication channels . . . 106
B.79 Trust percentage in the communication channels . . . 107
B.80 List of default channels . . . 107
B.81 Time spent in meetings weekly . . . 108
B.82 Time spent in face to face meetings weekly . . . 108
B.83 Time spent in web conference meetings weekly . . . 108
B.84 Time spent on email communication weekly . . . 109
B.85 Time spend with the customer weekly . . . 109
B.86 Customer's size . . . 109
B.87 Customer's knowledge of Scrum . . . 110
B.88 Product Owners in projects . . . 110

B.89 Customer's respect of 'no deadlines' 111
B.90 Customer's involvement in User Stories creation 111
B.91 Customer's participation in Scrum ceremonies 111
B.92 Customer's satisfaction with Agile 112
B.93 Participant's rating of projects success with Agile 112

List of Tables

2.1 Agile Manifesto Values. 7
2.2 Agile Manifesto Principles. 8

3.1 Collection of Small Scale Scrum Agile practices cont. 16

4.1 Small Scale Agile Manifesto Values. 17
4.2 Ceremonies in (Normal) Scrum vs Small Scale Scrum. 22
4.3 Roles in (Normal) Scrum vs Small Scale Scrum. 23
4.4 (Normal) Scrum vs Small Scale Scrum Roles Comparison. 23
4.5 Differences between (Normal) Scrum vs Small Scale Scrum. 24

5.1 Mapping of selected survey questions back to the RQs 26
5.2 Mapping of survey questions back to the research questions cont. . . . 27
5.3 Mapping of survey questions back to the research questions cont. . . . 28
5.4 Mapping of survey questions back to the research questions cont. . . . 29
5.5 Mapping of survey questions back to the research questions cont. . . . 30
5.6 Mapping of survey questions back to the research questions cont. . . . 31
5.7 Mapping of survey questions back to the research questions cont. . . . 32
5.8 Project Delivery Model. 38
5.9 Scrum ceremonies participation. 38
5.10 Comparison metrics for small scale projects delivered by small teams. 39
5.11 Comparison metrics cont. 40
5.12 Comparison metrics cont. 41
5.13 Small Scale Projects Comparison 44
5.14 Small Scale Projects Comparison cont. 45

6.1 Research Question Chapter Reference Table 47

List of Abbreviations

AFSP	**A**gile **F**ramework for **S**mall **P**rojects
GDPR	**G**eneral **D**ata **P**rotection **R**egulation
EMEA	**E**urope **M**iddle **E**ast and **A**ffrica
LeSS	**L**arge **S**cale **S**crum
PM	**P**roject **M**anager
PO	**P**roduct **O**wner
QA	**Q**uality **A**ssurance
QE	**Q**uality **E**ngineering
RQs	**R**esearch **Q**uestions
SSS	**S**mall **S**cale **S**crum

To my mother, Mieczysława Naglik-Gancarczyk, my sister, Diana Tsuchiya & my husband Michał Ogrodniczak

and

To the memory of my father, Andrzej Gancarczyk

Chapter 1

Introduction

1.1 Motivation

Historically, the beginning of Scrum for commercial product delivery goes back to 1986 with the publication of "The New Product Development Game" by Hirotaka Takeuchi and Ikujiro Nonaka. In 1995, Scrum entered its second phase of recognition with Jeff Sutherland and Ken Schwaber defining and presenting it to the wider audience. Despite the growth after 1995, however, it was not until 2001 that Agile and Scrum started gaining popularity. The publication of the Agile Manifesto and "Agile Software Development with Scrum" became a turning point.

Since 2001, Scrum has rapidly grown to become more than a revolutionary approach for software and non-software development enterprises with its recent update to make it generic and not software focused ***(Scrum.org (2018))***[50].

Approaches to Agile and Scrum have diversified dramatically and continue their remarkable evolution as they cater for new challenges.

Today, the various implementations of Agile all center around the concept of a team, with well defined cross functional roles present. Scrum, one of the most popular implementations of the Agile methodology as reported in "The 12th Annual State of Agile Survey" ***(VersionOne, Inc. (2018))***[32], has key roles in a team to ensure the smooth execution of the Agile methods and frameworks. Clearly aligned communication flows among Scrum teams are facilitated by Product Owners, ScrumMasters and external stakeholders to ensure that the team is set up for success. However, the adoption of Scrum by multiple teams in an organization is leading to issues around Scrum at scale, often termed Large Scale Scrum ***(LeSS (2018))***[1], managing multiple teams aligned towards a projects needs.

The size of global technology consulting market is currently estimated to be worth $50 million and is forecasted to grow by $2 million in 2019, and by $1 million in 2020 ***(Statista (2018)***[2]. In line with the growth of consulting market, the need for highly knowledgeable and skilled consultants grows along with consultants fees, customer's high product quality expectations and requirement for delivering best practices, tools and standards for advancing products in the future. The challenge for today's consultants is to deliver high quality and sustainable products adhering to best standards with less resourcing, optimal cost, expected time and duration.

As a Red Hat Consultant, I work on a number of 1-3 person projects with 20 Consultants on the site in Waterford and nearly 50+ in the Europe, Middle East and Africa

(EMEA) region. The entire consultancy industry including Red Hat Consultancy organization is very interested in the wide adoption of Agile because of the customer benefits it brings, such as rapid releases. Red Hat Mobile Engineering has had a huge success in the adoption of Agile in their 6 to 8 person teams with team members occupying different, well-defined roles. The Consulting teams can achieve the same success by following a set of principles, adopted for their small and distributed teams.

At present, the software industry is adopting a new team model. This can be seen in Red Hat Consulting, with projects ranging from a few weeks to 3 months, open source contributors working on their own and small start ups, with distributed teams of a maximum of 3 people. This is a significantly different team model compared to a traditional Scrum with 6-10 persons with skills ranging from Business Analyst, UI, UX, Frontend Engineer, Backend Engineer, Quality Engineer, and DevOps Engineer.

In small teams, role delimitation is impossible as members wear "multiple hats" in a given day, becoming the ScrumMaster, Software Engineer, Integrator and Quality Engineer, bringing a wider ranging viewpoint to bear, on a problem, from a single individual. The absence of separation of responsibilities within any team is associated with an overall reduction in the quality of the outputted product and a risk of "technical debt", which in software is related to the cost of any additional rework caused by implementing a simplified solution now, instead of taking a better but longer approach **(Hunt et al (1999)**) [46].

Small Scrum teams of a maximum of 3 people need to cover majority of skill set of a traditional Scrum team of 6-10 people. Therefore, in small teams, resourcing becomes an issue and by inference, quality can be questionable. Small consulting teams operate on a contract basis charging customers per person, per hour, per service to deliver contracted work, which once delivered, becomes the property and responsibility of the customer and consultants move to the next customer engagement. Typically this means lower numbers of resourced staff, shorter time to stay in budget and a lack of affinity with the products longer term life-cycle.

This thesis shows how small scale teams are capable of delivering the same quality as traditional teams despite of resourcing and contract boundaries. Expertise in rapid development and continuous delivery of small scale software solutions with continual customer involvement is a competitive advantage for small teams of consultants worldwide.

1.2 Hypothesis

The principal hypothesis of this thesis is that traditional Scrum principles and ceremonies can be modified to produce a Small Scale Scrum (SSS) framework, capable of supporting delivery of a highly resilient, sustainable, quality centric, full stack software solutions. This is possible in a fast paced, challenging environment dominated by small, geographically distributed teams. Delivered framework can significantly improve project life-cycle, that is, the management, development and delivery of the

project through vital quality assurance strategies.

The SSS model is expected to evolve over time as a result of Transparency, Inspection and Adaptation process (also known as the Three Pillars of Empiricism) ***(Scrum.org (2018))***[45] applied to the framework's implementations in software development projects.

Adoption of this model can effectively address a number of problems and concerns that exist in a traditional and Large Scale Scrum practices and implementations. These include project scoping, vague requirements, solution misunderstanding, communication problems, work organization, work prioritization, development specific blockers and lack of customer continuous involvement. Introducing SSS framework, specific principles, techniques and methods within any IT project, can ensure on-time delivery, business value, customer/user satisfaction, product quality, productivity, team satisfaction, project visibility, and process improvement.

1.3 Research Questions

Research questions (RQs) undertaken in this thesis are identified in the context of communication, quality, and business pillars which are considered as the key challenges faced by small scale, dedicated teams. This pragmatic approach is based on practical consideration of customer projects showcasing the value of the framework for software industry in the area of modern application development. Research questions are derived through real customer projects.

The following RQs are considered in this thesis:

RQ-1 Does the Agile Manifesto, with its Scrum methodology implementation stand for small scale distributed teams and small scale project implementations?

Understanding the main values and principles behind Agile Manifesto and its Scrum implementation can help to discover if the same iterative and people-centric approach with regards to communication, quality, and business in customer engagements can be adopted by small remote teams.

Within the Scrum framework, there exist well defined ceremonies and well defined roles implying management, collaboration, prioritization, development, and testing of software. This research question investigates if the Scrum methodology in its current form caters for small scale implementation or does it need to be modified before being applied. Potential problems are outlined and discussed as part of the investigation.

RQ-2 What Scrum principles and ceremonies can be applied to Small Scale Scrum, if any, with an emphasis on any implications for Small Scale Scrum teams?

Modifying principles and ceremonies of Agile development in a traditional Scrum leads to the formulation of SSS guidelines and recommendations. Therefore, understanding of structures and roles within small teams is critical for taking this new,

scaled down approach.

Existing concerns and implications for SSS teams are evaluated from communication, quality, and business perspective. Varied techniques and criteria utilized by Large Scale teams contribute to the discovery of implications targeted at Small Scale teams. This approach has potential to influence identification and formalization of preventative measures for successful full stack IT product development and delivery.

RQ-3 What are the barriers to adopting Small Scale Scrum? Do similar barriers also exist in regular Scrum teams that are trying to scale or is this a scale specific set of issues?

As a product matures it is normal for multiple teams of 7 +/- 2 ***(RGalen (2018))*** [4] (or 3..9 in the latest Scrum Guide ***(Schwaber et al (2013))***[5] to manage the life-cycle of it. In Scrum terms, that is known as LeSS. In SSS, small teams of a maximum of 3 individuals take charge of the project.

Scaling towards SSS team introduces a set of problems for how to scale not only the communication across multiple teams, but how to manage and control an aligned set of Scrum principles on several teams at once. The theory is to pre-align the teams in order for each team to successfully follow a Scrum methodology, at scale. It is the author's belief, that scaling multiple, Small Scrum teams will present similar challenges while also some novel challenges to ensure that the teams are pre-aligned in a similar manner as a LeSS framework team.

The end output of any Scrum project is a potentially shippable increment. The size of the team does not influence shippability of product in a given time period. However what is believed is that in smaller scale Scrum projects, quality centric principles are the first to suffer because of Scrum at small scale issues. While SSS has many obvious challenges, by putting quality at the forefront, it is believed that not only a quality centric increment can be delivered, but it can also address some of the misconceptions, such as role delineation, that currently hamper a real world implementation of Scrum, at small scale.

1.4 Contributions

At the heart of Small Scrum teams, is Small Scale Scrum - an innovative and adaptive Agile methodology based on the Agile Manifesto and Scrum principles. The SSS presented in this thesis is the main and novel concept, and a contribution to the field. For the purposes of the thesis, Agile principles, frameworks and ceremonies have been explored to understand how these principles and ceremonies differ in Large and Small Scale Scrum implementations.

In line with the research questions and hypothesis outlined above, the main contribution of the research thesis is a set of principles for SSS derived from implementation of the principles on real customer projects delivered by the Red Hat Consultancy team. This contribution will take form of a conference presentation, journal publications/ technical blog posts and a principal document outlining a set of guidelines

and framework for Consultancy in EMEA region.

To date contributions include "Quality, Performance and Success in 1-3 Person Teams - A Story of Small Scale Scrum" presentation on Agile Cambridge 2017 Conference which has been well received by its participants. "Agile Cambridge is England's premiere practical, hands-on Agile software development conference" ***(Agile Cambridge (2017))***[3] which influences Agile Software Development Practitioners. Parts of this research have also been published in journals/ blogs such as Medium ***(Gancarczyk (2018))***[55, 56, 57] and Opensource.com (proposal for an article has been submitted). Additionally, "Small Scale Scrum vs Large Scale Scrum" survey has been conducted on 54 participants spread across Red Hat including Red Hat Mobile Engineering, Red Hat Openshift Engineering, Red Hat JBoss and Red Hat Consulting. The outcome of the survey shows how small and large scale teams use Scrum in their daily work.

1.5 Thesis Outline

This research thesis outlines modifications made to the traditional Scrum approach to Agile Software Development, to ensure that SSS works and that size of a Scrum team should not be a barrier to delivering a quality product. However, not only does it function efficiently, it delivers a highly resilient and low volume technical debt project which may be potentially full stack. This can be achieved with a quality driven experience for a Scrum team that might lack a Quality Engineering (QE) or Quality Assurance (QA) presence. Real world examples from Red Hat Consultancy teams are presented to compare the performance from a quality perspective to that of a traditional Scrum team across several key quality metrics. The results are on parity with the output, from a quality perspective, of a cross functional Scrum team operating in a LeSS manner. A number of quality driven steps as a re-imagined, quality centric, SSS Manifesto are recommended.

The thesis proceeds as follows. Chapter 2 discusses the state of the art in a number of related areas of research including the Agile Manifesto, Agile principles and ceremonies, Large and Small Scale Scrum, Large and Small Scale teams and Research Methodologies. All these areas are related to the three RQs outlined in Chapter 1.

Chapter 3 primarily identifies a set of adjusted principles, techniques, implementations, and recommendations for SSS from the perspective of roles, responsibilities, execution of Scrum ceremonies by small Agile teams and on the basis of research findings from Red Hats' Scrum teams.

In Chapter 4, Small Scale Agile Manifesto and SSS framework are introduced. This is derived from observations in Chapter 3 and evaluation of real customer projects in Chapter 5.

Chapter 5 presents results from "Small Scale Scrum vs Large Scale Scrum" survey and Red Hat customer projects including Agile transformation, experimental and

validation projects. SSS principles outlined in Chapter 3 and SSS framework presented in Chapter 4 are recycled into projects delivered by the author's team to examine impact on the execution of SSS.

Finally, Chapter 6 presents the conclusions drawn based on the research carried out to date and answers provided to three RQs based on both academic and strong practical experience. This will also take the form of a critical Retrospective to further refine the SSS model and examine the outcomes of the overall thesis which will give rise to future work.

Chapter 2

State of the Art

In this chapter a review of the literature relevant to this thesis is presented. Topics covered include Agile Manifesto, Agile principles, Large Scale Scrum (LeSS) and Large Distributed teams, Small Scale Scrum (SSS) and Small Scale Distributed teams and barriers to Small Scale Scrum.

2.1 Agile Manifesto, Large Scale Scrum and Large Teams

This section reviews the Agile Manifesto, its principles and Scrum using relevant historical literature.

In 2001, seventeen individuals created a set of specific ideas for software development and called it "Agile Manifesto" ***(Kent Beck et al (2001))***[6]. Agile is an alternative approach to documentation based and heavy process driven software development.

Table 2.1 outlines the four values underlying the Agile position.

TABLE 2.1: Agile Manifesto Values.

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

Principles behind the Agile Manifesto are listed in Table 2.2 below:

TABLE 2.2: Agile Manifesto Principles.

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

The Agile Manifesto principles can be found in Scrum implementations which "allow teams to focus on delivering product and improved communication has made it one of the easiest and best software development techniques" ***(Khmelevsky et al (2017))***[7]. Challenges, benefits and lessons learned from an analysis of case studies and real life distributed Agile and Scrum projects conducted since 2011 are presented.

Similar position of Scrum is delivered in "The 11th Annual State of Agile Survey" ***(VersionOne, Inc. (2017))***[8], conducted between July and December 2016. According to the survey, Scrum "works well for any complex, innovative scope of work. The possibilities are endless. The Scrum framework is deceptively simple." In November 2017, Scrum was expanded to non IT usage as mentioned in the Scrum Guide 2017 Update ***(Scrum.org (2018))***[50] incorporated into the official Scrum Guide by Ken Schwaber and Jeff Sutherland ***(Scrum Guide (2018))***[51].

Kapitsaki et al (2014)[9] shows the success of Scrum adoption in software projects. Presented in the paper analysis of the success or failure factors of Scrum in comparison to heavyweight approaches such as the Waterfall model ***IEEE Future Directions (2018)***[49], Spiral model ***Boehm (1988)***[48] and the Rational Unified Process ***Krutchen (2003)***[47] show the success of Scrum with eight out of ten successful Scrum projects.

Each of the papers reviewed above confidently place Agile with its Scrum methodology at the forefront. Therefore this research takes into account Agile Manifesto values, its principles and that of Scrum to formulate SSS guidelines and framework. Due to the small numbers in a typical small team ranging from 1-3 people, it is not possible to align on the Scrum methodology in its entirety. All of the research is therefore concentrated on modifying Scrum for traditional sized teams and scaling that process out with a particular emphasis on the quality outputs of the team and customer satisfaction.

The literature offers some advice on practices used in large scale distributed Agile software development projects, but suggests that "It is very important to understand if, when, and how Agile methods can be suitable in the large." ***(Dingsøyr et al (2014))*** [10].

Increasingly and despite of many variations of Agile methodologies, Scrum and modified Scrum take a central part in software development process and with confirmed 78% mark "continue to be the most common Agile methodology used by organizations" ***(VersionOne, Inc. (2017))***[32].

Traditional Scrum focuses heavily on the use of ceremonies ***(Quickscrum (2018))***[12] and specific roles ***(Scrum Alliance (2018))***[13] to achieve Agility within the team. Most frequently mentioned Scrum principles and Scrum roles are listed and explained in Appendix C.

While ***Dingsøyr et al (2014)*** indicates a lack of agreement on the large-scale Agile development definition ***(Dingsøyr et al (2013))***[11] and accepts Webster's definition of "large-scale" as "very extensive; of great scope" ***(Webster (1989))***[14], ***(LeSS(2018))*** defines it as follows "Scaling Scrum starts with understanding standard one-team Scrum. From that point, your organization must be able to understand and adopt LeSS, which requires examining the purpose of one-team Scrum elements and figuring out how to reach the same purpose while staying within the constraints of the standard Scrum rules."

There are two different Agile large scale Scrum frameworks, namely (1) LeSS which is designed for up to 8 teams (of 8 people each) and (2) LeSS Huge which is for up to a few thousand people on a single product ***(LeSS(2018))***.

Methodologically, LeSS is based on traditional Scrum with its suggested team size of 7 +/- 2 or 3..9 people. ***Dingsøyr et al (2014)*** identifies three team sizes: 1 person small-scale, 2-9 persons large-scale and 10+ persons very large-scale. Coordination of small teams "can be done using Agile practices such as daily meetings, common planning, review and retrospective meetings", coordination of large teams "can be achieved in a new forum such as a Scrum of Scrums forum", whereas coordination of very large-teams can be done using "Several forums ... , such as multiple Scrum of Scrums" ***Dingsøyr et al (2014)***. It is worth noting that "small" is never explained in detail, it's only glanced over.

Scrum practices are investigated through a number of case studies by ***Paasivaara et al (2008)***[15] to show that they can be successfully applied in a 40- person development organization distributed between Norway and Malaysia. This paper discusses Agile practices adopted, agility supporting distributed practices, challenges

and benefits.

Agile practices adopted included the use of teleconference and web cameras for daily meetings, 4- week sprints, 2-week maintenance sprints, weekly Scrum-of-Scrums, distributed Sprint planning, distributed Sprint demos and retrospective meetings, nightly builds with automated testing, isolated backlogs for each team, transition backlog, and team separate office spaces. Agility supporting distributed practices included unofficial distributed meetings, centralized version control, visiting engineer during the first iteration, onsite system expert, frequent visits, and annual gatherings. Challenges with Scrum in large scale distributed team are also identified and listed: no possibility for video conference, misunderstanding requirements and silence caused by distance. Benefits included better overall software quality, better and more frequent communication between sites, improved motivation of team members ***(Paasivaara et al (2008))***.

Ramesh et al (2006)[16], ***Simons (2002)***[17]) support position of daily standups, Sprint planning, Sprint review meetings and Sprint demonstrations as the most frequently reported Scrum mechanisms.

In large distributed teams, meetings consume a substantial amount of time. Suggested approach towards shortening teams participation in meetings is described below:

- For daily standup meetings, ***Sutherland et al. (2007)***[18] recommends that team members answer three standard Scrum questions outlined in Appendix E, via email before the Skype or web conferencing meeting.
- For Sprint planning meetings, ***Layman et al (2006)***[19] and ***Simons (2002)***[17] provide conflicting approach. ***Layman et al (2006)*** suggests that only lead-developers should participate whereas ***Simons (2002)***[17] believes that the entire team should be present subject to pre-work before the actual meeting.
- For Sprint review meetings, ***Berczuk(2007)***[20] recommends to center meetings on the local team and Product Owner (PO).
- For Sprint demonstrations, ***Fowler (2007)***[21] advises that Development Team should do a demo to the customers at the end of each Sprint using videoconferencing with desktop sharing or VNC products.

Important elements of LeSS and Large teams are well known. All the resources are on large and normal teams. There is little to no small scale studies.

2.2 Small Scale Scrum and Small Teams

The literature reveals weakness on the adoption of SSS in software projects. Existing research does not pertain to open source teams or teams geographically distributed. Traditional Scrum approaches and research all focus heavily on the use of ceremonies and specific roles to achieve Agility within the team.

Work by ***Linda Rising and Norman S. Janoff (2000)*** [22] introduces a "flexible approach to development processes so that each team can apply what works best", but refers to small team as not larger than 10 people. By looking into their teams in small

development environments, they concluded the following: "(1) the product becomes a series of manageable chunks, (2) progress is made, even when requirements are not stable, (3) everything is visible to everyone, (4) team communication improves, (5) the team shares successes along the way and at the end, (6) customers see on-time delivery of increments, (7) customers obtain frequent feedback on how the product actually works, (8) a relationship with the customer develops, trust builds, and knowledge grows, and (9) a culture is created where everyone expects the project to succeed".

Dingsøyr et al (2014) is among a very few addressing clearly how small the team should be to be considered as small scale. According to the publication, small scale team is a 1 person team and states that coordination of small teams "can be done using Agile practices such as daily meetings, common planning, review and retrospective meetings". It's also suggested that "positive experience of Agile development methods in smaller projects has created interest in the applicability of such methods in larger scale projects". The paper, however, does not elaborate on communication, quality and business in small scale distributed teams.

Paasivaara et al (2008), advocates that "the successful use of Agile practices in small distributed projects already exist", but only refers to XP, and not Scrum based implementations.

Xuesong (Sonya) Zhang and Bradley Dorn (2011)[23] observes how "daily Scrums, backlogs, and sprints were successfully adopted to the development ... in a small-scale, time-intensive web development project at a college-level IT competition".

In a related work, ***Seiyoung Lee and Hwan-Seung Yong (2003)***[24] states that "Agile methods are highly attractive for small projects, but no Agile method works well as a standalone system. Therefore, some adaption or customization is always required." The paper proposes Agile Framework for Small Projects (AFSP) as an extension to Scrum process and Agile practices. This framework consists of collective Agile techniques and programming best practices. ***Seiyoung Lee and Hwan-Seung Yong (2003)*** suggest that focusing on people can lead to productivity and quality improvement, but their practices and ceremonies need to be updated and optimized to make them realistically applicable to today's rapid and distributed software development environment where resourcing and time is limited. Quality and team interactions are covered in the paper to some extent, but communication and business first practices for distributed small teams are missing.

Other related work identified in the SSS space include extended Scrum method for small projects with team size defined as ranging from 5 to 10 people ***(Rajkumar et al (2014))***[25], focused on developing " the project before the customer itself so that the communication develops and time consumption is reduced". For ***Johannes Brodwall (2013)***[26] feedback in Sprint review and demonstration is the most critical aspect of Agile Manifesto as "delaying feedback is delaying critical learning" and by "showing what you have done to someone who cares frequently, you can improve".

Alex Andrews (2017)[27] interestingly outlines structured approach on "how to bring Scrum into your one-person operation" advising on introduction of 5 minutes daily standups, 30-45 minutes story time, sprint release, last day of the Sprint, 2 hours long retrospective, 2 hours Sprint planning, keeping organized with the task board,

rest and explore on last day of the Sprint.

Literature is, however, missing advice for small scale distributed teams with regards to communication, quality and business. This paper presents a progress study for a distributed software development projects with the example of three projects which used hybrid/waterfall methodology, followed by a project which applied Agile practices from Scrum and finally a project that adopted SSS principles.

2.3 Barriers to Small Scale Scrum

The literature on barriers to SSS practically does not exist. Furthermore, there is not any reference to SSS specific context of communication, quality and business.

Some indirectly related papers include ***Poole (2004)***[28] describing five challenges to small projects that make traditional approaches ineffective including lack of planning, low priority, inexperienced project teams, project manager responsible for multiple functions, the use of standard project management tools and processes for small projects. The main challenge for the organization remains on application of the most suitable Agile practices.

Nils Brede Moe et al (2009)[29] paper deals with overcoming barriers to self-management in software teams with team size of 6-8 individuals. Paper recognizes three barriers on team level including individual commitment, individual leadership and failure to learn, but does not narrow it down to Scrum nor SSS.

Paasivaara et al (2008) outlines challenges caused by distribution of team members including communication problems, lack of close physical proximity, lack of team cohesion, lack of shared context and knowledge and unavailability of team members.

Additional challenges in so called "virtual teams" defined as "geographically dispersed individuals, collaborating through electronic communications" would be lack of trust, coordination issues, cultural conflict, technology-task-fit and team performance issues ***Michael She (2010)***[30].

It is understood that most of the above barriers apply to Scrum and LeSS and could be considered as candidate barriers to SSS by inference.

2.4 Research Methodologies

The use of survey and focus groups was dictated mainly by the suitability of addressing RQs in this thesis and by the overall industry utilization of surveys.

Examples of surveys' utilization are Developer's Surveys carried out by Stackoverflow ***(Stackoverflow (2018)***[41], Hackerrank ***Hackerrank (2018)***[42] and GitLab ***GitLab (2018)***[43]), just to name a few.

Similarly, focus groups including Conferences are widely used by the industry professionals to share and validate knowledge obtained. Example of this is attended Agile Cambridge 2017 Conference as well as pre- and post-survey interviews with selected Agile Practitioners in Red Hat.

Chapter 3

Small Scale Scrum Principles

This chapter proposes Small Scale Scrum (SSS) principles and guidelines. Communication, quality and business aspects considered for SSS framework come from a number of small scale projects within Red Hat Consulting and are regarded as main pillars within the framework. These three pillars are rightly considered as theory results in this research thesis and are discussed in Chapter 4. Proposed in this chapter principles and guidelines help address (1) what approach towards communication should be taken in SSS, (2) what processes should be introduced to ensure the highest quality of delivery, and (3) what are the benefits behind implementing SSS for the business.

The Small Scale Scrum principles are non-negotiable and must be applied as presented in Figure 3.1.

FIGURE 3.1: Small Scale Scrum Principles

- **Value-Based Communication** - This principle emphasizes the inner and outer communication within Development Team and the customer and is focused on delivering a value. Understanding of the solution, its purpose and its desired

functionality is based on effective communication. Initiating and maintaining valuable communication between parties requires openness and dedication in looking for the best solution to a given functionality request. Openly discussing and sharing steps to be taken or taken to fix any issue or to implement a feature constitute effective communication and it may lead to some further discussions around progress made in software delivered which can in turn uncover any potential omissions in the requirements. With finite time, all communication should be valuable.

- **Quality-First Development** - This principle focuses on taking a quality approach to software development each Sprint. This means ensuring that features are delivered as per acceptance criteria, solution is bug free (or at least free from any evident bugs), any inconsistencies in the software are removed, solution is tested, any edge case bugs along with any omitted features are reported, logged and considered by the customer.

- **Delivery Ownership** - This principle is about taking initiative in driving software delivery by Development Team on a Sprint-to-Sprint basis. Enabling the team to consult the customer directly is seen as having a positive impact on the overall performance of the team and customer satisfaction. Removal of micromanagement barrier in a project, if such exists, is critical to allow the team to take ownership in delivering software solutions.

- **Iterative Sign Off** - This principle focuses on reducing technical debt and identifying gaps in requirements through iterative sign off approach. As the solution evolves and matures, business requirements change. With iterative development, functionality delivered within a Sprint should be signed off on Sprint completion. Similarly, requirements for the upcoming Sprint should be signed off prior to a new Sprint commencement.

A subset of accompanying guidelines listed in Table 3.1 can be used to build confidence in using SSS framework and help achieve the objectives of any small scale project. These guidelines are addressing communication, quality and business challenges faced by small teams.

TABLE 3.1: Collection of Small Scale Scrum Agile practices cont.

Category	**Guideline**
Software Requirements	Requirements Creation
	Requirements Review & Refinement
Software Development	Cyclomatic Complexity Reduction
	Change Request Frequency
	Bugs Turnover Increase
	Small & Regular Test & Production Builds
	Simple Design
	Mentoring/ Pair Programming/ Code Review
	Shared Code Ownership
	CI/CD
Software Testing/ Quality	Bugs Identification Decrease
	Bugs Verification Increase
	Feedback to Developers through good communication
	Test Coverage Increase
	Test Approach Definition & Enforcement
	Final Regression Test Bugs Decrease
	UAT Test Bugs Decrease
	Software Quality Increase
	Quality Tools Use Increase
Software Project Management	Initiation and Scope Definition
	Software Project Planning
	Project's Progress Review and Evaluation
	Project Release
Additional Proposed Guidelines	Sprint (Sprint Zero, Regular Sprint, Inactive Sprint and Hardening Sprint)
	Sprint Planning
	Sprint Estimation
	Sprint Underestimated Work Decrease
	Sprint Retrospective
	Sprint Demonstration
	Sprint Code Freeze
	Sprint Test
	Daily Standups
	Developer's Minimum Testing
	Backlog Review & Refinement
	Sprint Emergency Strategy
	Team Collaboration Tools Use

Chapter 4

Small Scale Scrum Framework

This chapter proposes Small Scale Agile Manifesto along with its Small Scale Scrum (SSS) implementation.

4.1 Small Scale Agile Manifesto

The Agile Manifesto is an umbrella term that describes and governs several light-weight and fuller Agile methodologies to handle IT teams and projects. Scrum, Kanban, Lean Development, Crystal and Extreme Programming (XP) are among the most popular and light-weight Agile approaches.

While SSS outlined in this chapter fits into the Agile Manifesto, it's believed that some of the additional values would complement the Agile Manifesto and enhance it for smaller teams. These additional values are outlined in Table 4.1.

By understanding small teams, new ways of planning, developing and delivering quality software need to be considered:

TABLE 4.1: Small Scale Agile Manifesto Values.

Wider communication over narrow communication
Team feature delivery over individual responsibility
Quality delivery over speed of development
Multiple project responsibilities over fixed assignment
Accelerating innovation over marginal request driven thinking
Customer growth over customer engagement

Wider communication over narrow communication - Having narrow communication with the manager in any project is important and needs to be standalone, but it's wider communication that offers more value. Wider communication would involve all stakeholders as team member is Product Owner, ScrumMaster and the team in one. Excellence through application of best principles in every team-team or team-customer communication is very important. Therefore, time for preparation before meetings to best accommodate changes and ensure productive outcome of each interaction should be encouraged. Communication can be enabled via preferred communication channels to quickly and effectively create a welcoming environment.

Team feature delivery over individual responsibility - Individual responsibility of team members for delivering single features is important and is considered as a standard and standalone practice across software projects. Members of small teams, however, are expected to have much broader involvement in the project life-cycle and therefore their attention need to be focused on taking mutual responsibility for delivering work items. In this approach, teamwork plays an important role with team members working together as a single unit to ensure success of the project from the bottom up. This can be achieved with some common techniques including support for remote team members, fair workload, pair programming, code review and shadowing.

Quality delivery over speed of development - Rapid development and high quality delivery are well known expectations in every customer engagement. While speed of development is important, quality delivery has much greater impact on success of the project. For continuous success, team members need to focus on delivering quality. Investing time in quality development and testing with the use of quality assurance techniques, tools, mentoring and training can help the team to continuously excel.

Multiple project responsibilities over fixed assignment - Fixed project responsibility is important and is generally practiced but for small scale teams the real value is in taking additional roles so that software developer becomes frontend developer, backend developer, quality engineer, UI designer. The idea behind this approach is to ensure that the small team is self-sufficient as much as possible. For small teams to adopt multiple responsibilities, workload needs to be fair and project's processes need to be streamlined. This can be achieved by continuously reviewing, improving and simplifying processes to help the team to focus solely on delivering. Additionally, coaching can be introduced to provide team members with guidelines to help them improve their skills.

Accelerating innovation over marginal request driven thinking - Request driven thinking in enterprise engagements is important but it's innovation that customers value the most. Planting innovation is critical to get the team and customers to think outside of the project's sole requirements so that they can envision the final solution with the most optimal architectures, requirements and designs.

Customer growth over customer engagement - A successful customer engagement is very important for success of every project, but it's not sufficient for building and maintaining a strong and successful relationship with the customer. For small teams, it's important to approach business engagement from the customer's business success perspective. Growth or creation of the customer's business through the use of software solutions is the customers highest priority, and should also be the highest priority of the team.

4.2 Small Scale Scrum

Due to its strong statistical position with implementation popularity of 56% globally ***(VersionOne, Inc. (2018))***[32], developers liking, high success rates in both Scrum

adoption and project deliveries, strong principles and values behind Scrum including focus, courage, openness, commitment and respect, Scrum is considered as a leading candidate for the implementation of Small Scale Agile ***(Scrumaliance(2018))***[33].

The 12th Annual State of Agile Survey 2018 reports that Scrum (56%), ScrumBan and Scrum/XP Hybrid (14%) collectively rated at 70%, continue to be the most common Agile methodologies used by organizations ***(VersionOne, Inc. (2018))***.

The State of Scrum 2017 - 2018 report ***(Scrumaliance(2018))***[40] shows Hybrid Scrum as the most popular methodology with a sharp drop in the traditional Scrum (year on year substantial drops). Companies are increasingly moving from Waterfall methodology and are modifying the Scrum. The small teams are using Small Scale Agile and are not embracing it all. In this thesis, SSS is technically seen as a Hybrid variation and hence a good choice to experiment with Small Scale Agile.

For comparison, ***VersionOne, Inc. (2018)*** rates Kanban, Lean Development and Extreme Programming (XP) at 7% popularity globally with Kanban being the most popular along the three Agile methodologies with a score of 5%.

Kanban, Lean Development and Extreme Programming (XP) don't provide all in one solution for teams in the same way Scrum does.

The SSS has been delivered through modification process based on scaling traditional Scrum ceremonies down to fit a small team of a maximum of 3 members and center heavily on communication, quality and business to deliver success in a short time period. Review of literature and real enterprise projects have indicated a strong Agile Scrum movement in the space of modern application development.

The SSS is an implementation of Small Scale Agile and can be best described as "a people first framework defined by and for small (a maximum of 3 people) teams and supporting planning, developing and delivering production quality software solutions". Proposed framework centers around the concept of team members occupying additional roles in any given project.

The framework is important due to its strong support for small scale, distributed teams seen in organizations all over the world and in particular in Red Hat Consulting. With the growth of customer expectations of rapid delivery and high quality solutions, small teams need to find new ways on how to meet continuously growing expectations. The SSS is a set of guidelines and principles to help address the issue.

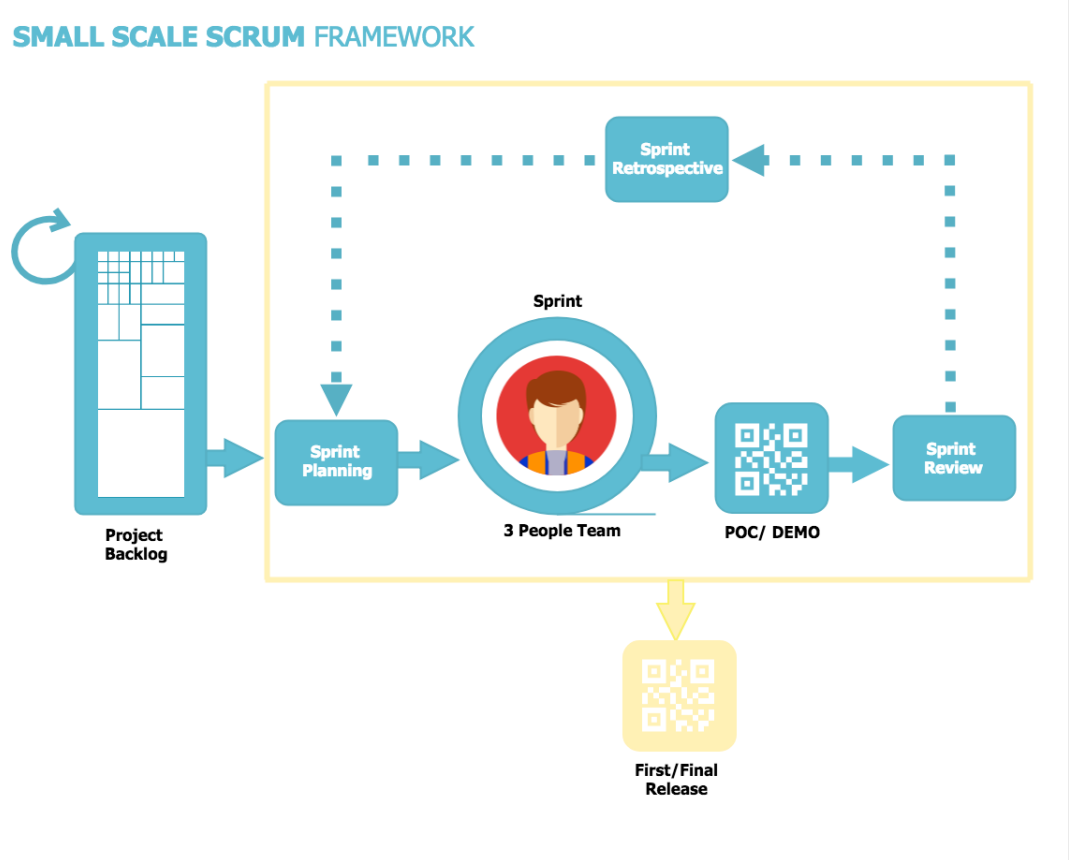

FIGURE 4.1: Small Scale Scrum Framework

Project Backlog is a list of everything that is known to be required in the project. The Project Backlog is created before commencement of any development work and maintained by Development Team due to the absence of the Product Owner (PO). The Project Backlog contains development tasks and requirements for software. This replaces the traditional Product and Sprint Backlogs.

Sprint Planning meeting is time-boxed and focused on planning work for the upcoming Sprint. The Development Team runs and is involved in Sprint Planning. It is recommended that the team does some level of preparation work before Sprint Planning in order to keep the planning concise. The Sprint Planning is value based. At this point requirements planned to be worked on in the upcoming Sprint should be clear and contain acceptance criteria. Capacity, typically measured as velocity, is not used, instead the team has to take an educated guess. Velocity only emerges after 3+ Sprints and often the SSS projects are over at that stage.

3 People Team is Development Team and Project Manager (PM) (most of the time). The Development Team is expected to be involved in gathering and clarifying business requirements, development work, quality testing, meetings organization and running and releasing software to the customer.

POC/Demo is a realization of small work completed in the Sprint with the aim of showcasing progress made in the solution development. Any small deviation in work not fully completed are discussed during demonstration.

Sprint Review meeting is organized and run by Development Team and PM (if involved in the project) and attended by the customer and/or the customer team. The Sprint Review is time-boxed and contains demonstration of Sprint work with a short outline of work completed/not completed, bugs raised (if any). At the end of demonstration, the customer is asked for a feedback. All feedback notes are taken and incorporated into future Sprints. There is very little preparation required before the meeting, if any, to keep it short and structured.

Sprint Retrospective meeting is organized and run by Development Team and PM (if involved in the project) and attended by the customer and/or the customer team. The Sprint Retrospective is time-boxed and does not require prior preparation. Due to the small size of the Development Team and small Sprint builds (with smaller number of features delivered), the meeting is relatively short. The Sprint Retrospective takes place after the Sprint Review and prior to the new Sprint Planning.

Sprint Review, Sprint Retrospective and Sprint Planning can be combined and happen in the one meeting. On average, all three meetings combined are 1.5 hr long. They are concise, structured and avoid any unnecessary/unrelated discussions.

First/Final Release is the end result of the project. The Development Team runs the tests, verifies completed work, confirms fixes and signs off the release build.

The Scrum guide timebox guidelines are followed for SSS ceremonies.

Sprint State is the duration 1 week to 4 weeks if the project is less than 30 day duration and for a minimum of 2 Sprints.

TABLE 4.2: Ceremonies in (Normal) Scrum vs Small Scale Scrum.

(Normal) Scrum Framework	**Small Scale Scrum Framework**	**Comments**
Product Backlog	Project Backlog	In Small Scale Scrum, Project Backlog is a list of all things that need to be done to deliver project. Project is considered to be a few weeks effort delivering POC/Demo at the end.
Sprint Planning	Sprint Planning	In Small Scale Scrum, Sprint Planning is the same, but more structured and quicker in execution.
Sprint Backlog	Project Backlog	In Small Scale Scrum, Sprint Backlog is not required. Team relies on Project Backlog.
Scrum Team (6-8 cross functional members to deliver in the vertical)	Scrum Team (1-3 to deliver in the vertical)	In Small Scale Scrum, team size is small, but expected to deliver the same quality as Normal Scrum Team. Team members have multiple roles in the project.
Daily Scrum	For team greater than 1 person. Otherwise not recommended	In Small Scale Scrum, Daily Scrum is not mandatory. It would not be justified to run Daily Scrum for 1 person team.
Sprint Review	Sprint Review	In Small Scale Scrum, Sprint Review is the same but typically structured and quicker in execution.
Product Increment	POC/Demo	In Small Scale Scrum, team does not deliver Product Increment and therefore it does not follow any specific process for delivering Product Increment including integration with CI/CD, PR Reviews etc. In Small Scale Scrum, POC or Demo is delivered and process overhead is avoided to allow the team to deliver faster.
Sprint Retrospective	Sprint Retrospective	In Small Scale Scrum, Sprint Retrospective is the same but more structured and quicker in execution. From Sprint 2 on-wards, Sprint Review, Sprint Retrospective and Sprint Planning can happen in the single meeting.

The SSS does not have sufficient resources to separate roles and therefore is faced with a challenge of realizing multiple roles. The difficulty having multiple roles in a single project is relatively high as each role is posing different level of complexity, require different skills set and pose a risk of under performing due to the context switching. The context switching has a negative impact in a technical role like programming for instance and should be avoided, if possible. In SSS, there is Development Team and PM who are also Product Owner (PO) and ScrumMaster. They could be spread out across multiple projects, in particular PM is not a dedicated role. Oftentimes, Development Team who builds the software, Product Owner who holds the vision for the product (in Small Scale Scrum, Product Owner holds the vision for the project) and ScrumMaster who helps the team with the best use of Scrum is a single person. Due to the high risk of not being able to perform all the Scrum roles, a simplified approach is taken as per Table 4.3 below.

TABLE 4.3: Roles in (Normal) Scrum vs Small Scale Scrum.

Small Scale Scrum Roles	**(Normal) Scrum Roles**
Development Team, Project Manager	Development Team Product Owner ScrumMaster

Roles in (Normal) Scrum and Small Scale Scrum are compared in Table 4.4.

TABLE 4.4: (Normal) Scrum vs Small Scale Scrum Roles Comparison.

(Normal) Scrum Role	**(Normal) Scrum Role Type**	**Small Scale Scrum Role**	**Small Scale Scrum Role Type**
Development Team	Full Time, Dedicated	Development Team	Full Time, Dedicated
Product Owner	Full Time, Dedicated	No dedicated Team Member. Can be fulfilled by Development Team Member.	Part Time, Shared
ScrumMaster	Full Time, Dedicated	No dedicated Team Member. Can be fulfilled by Development Team Member.	Part Time, Shared

In terms of SSS values, it's worthwhile to mention that all existing Scrum values including focus, courage, openness, commitment and respect remain valid with an addition of teamwork, quality, business focus.

Table 4.5 shows differences between (Normal) Scrum and Small Scale Scrum to point out any deviations between the two Agile methodologies.

TABLE 4.5: Differences between (Normal) Scrum vs Small Scale Scrum.

Ceremony/ Practice	**(Norma) Scrum**	**Small Scale Scrum**
Dedicated SM/PO	Yes	No
Acceptance of Sprints (no PO)	No	Yes
Daily Standups	Yes	No
Product Backlog	Yes	No
Product Increment	Yes	No
Sprint Planning	Yes	Yes
Sprint Review	Yes	Yes
Sprint Retrospective	Yes	Yes
Delivery of quality work	Yes	No
Agreed deviation from original functionality	No	Yes
Work/Code ownership	Yes	No
Knowledge sharing	Yes	No
Work items estimation e.g. Story Points	Yes	No

Chapter 5

Small Scale Scrum Survey & Projects Validation

This chapter is justification for utilization of survey and focus groups as main research methodologies for the purposes of addressing RQs undertaken in this thesis. The process of designing the survey questions and drawing on findings from careful analysis of responses are described in respective sections of this chapter. The survey followed by interviews with key individuals identified during its creation process is outlined. Review of real, small scale customer engagements from Red Hat Consulting is presented in the context of research validation.

5.1 Survey Rationale & Design

The rationale behind using the survey is multifold. Distribution of teams globally (or else lack of co-located teams), too small data sample available locally, access to the customers and industry guided utilization of surveys ***(Stackoverflow (2018)***[41], ***Hackerrank (2018)***[42], ***GitLab (2018)***[43]) are considered as the main reasons behind using the survey methodology.

The use of focused groups included presenting topic titled "Quality, Performance and Success in 1-3 Person Teams: A Story of Small Scale Scrum" and discussing the thesis hypothesis during the Agile Cambridge 2017 Conference ***(Agile Cambridge (2017))***[3] as well as pre- and post-survey interviews with a couple of selected Agile Practitioners. The pre-survey interviews contributed to the organization and creation of survey questions. The post-survey interviews focused on (1) the process of managing, developing, testing and delivering software solutions in time pressure, (2) software quality and (3) software requirements (user stories and acceptance criteria).

The survey design was a three step process and involved survey requirements gathering, questions creation and validation.

- **Survey Requirements** - The first step involved a round of interviews with selected members of Red Hat Engineering and Red Hat Consulting. Interviews had an informal format with a focus on the use or lack of Agile in the interviewees' projects. This step helped identify main sections of the survey including project(s) size(s), role(s), Agile and Scrum principles and ceremonies utilized, approach to software requirements, development, testing and project management, quality, communication and customer related aspects.

- **Survey Creation** - The outcome of the interviews, fed directly into the process of creating survey questions which exhaustively covered sections identified in the survey requirements step above.

  The number of questions in the survey was rather large for a typical survey, but it was important to create an exhaustive data sample to cover all possible scenarios related to the use of Agile in order to address RQs adequately. Mapping of the most relevant survey questions to RQs is presented in Tables 5.1, 5.2, 5.3, 5.4, 5.5, 5.6 and 5.7.

TABLE 5.1: Mapping of selected survey questions back to the RQs

Survey Question	**Research Question**
1.1 Does your team practice a form of Agility? **Purpose: Is this survey suitable for a targeted audience?**	RQ-1
1.2 What is your current team size for an Agile project? **Purpose: What differentiates small and large Agile teams? This is to understand if Agile with its Scrum implementation stands for small scale teams.**	RQ-1
1.3 Is your current team distributed (if at least 1 team member is working remotely?) **Purpose: Does Agile (with its Scrum implementation) support distributed teams?**	RQ-1
2.1 Which one best describes your typical project duration? **Purpose: Does Agile (with its Scrum implementation) support small scale projects? This is to understand if Agile stands for small scale project implementations.**	RQ-1
3.1 Are you frequently involved in more than 1 project? i.e. assigned to more than 1 team? 3.1.1 If 'Yes', please indicate the number of projects. 3.1.2 If 'Yes', are these customer projects or sustaining/maintenance based projects? e.g. technical debt reduction, automation. 3.1.3 Is work estimated before start of development? 3.1.3.1 If 'Never/Sometimes, Most of the Time', are extensions available?	RQ-3

TABLE 5.2: Mapping of survey questions back to the research questions cont.

Survey Question	**Research Question**
3.3 Does the customer clarify requirements timely? 3.4 Do you write tests for your code? 3.5 Is your code reviewed before merging? 3.6 Do you adhere to best coding practices? 3.7 Is your code documented? 3.11 Do you ask for help/mentoring if task is complex? 3.12 What Scrum roles are present in your current project? 3.13 What is your current primary role in the team? 3.14 Are you taking additional roles in your current project? 3.15.1 What other role(s) do you fulfill in a typical project? 3.15.3 How difficult is changing tasks on a daily basis? 3.15.4 How much time do you feel you lose (if any)? 3.15.5 Do you get your work completed on-time? **Purpose: What are the implications for small scale teams i.e (1) delivery implications: time-based caused by involvement in multiple projects playing the same or different roles; skills-based caused by lack of required skills, (2) requirements clarifications, (3) quality implications (tests, code reviews, bugs, best practices)? Can these implications be considered as barriers to adopting Small Scale Scrum? If so, do they also exist in regular Scrum teams?**	RQ-3

TABLE 5.3: Mapping of survey questions back to the research questions cont.

Survey Question	**Research Question**
4.2 Which agile software development methodology is followed by your team? 4.3 Which of the following Scrum ceremonies and principles is your team practicing (if any)? i.e. Daily Standup, Sprint, Sprint Planning and Estimating, Sprint Code Freeze, Minimum testing before QE, Sprint QE Testing, Backlog Grooming, Sprint Retrospective. 4.4 Does your team use any of the following? i.e. Sprint 0, Additional Sprint (extra Sprint created during active Sprint for additional user stories completed in the active Sprint), Hardening Sprint (extra Sprint at the end of the project), Pre-work before Sprint planning, Pre-work before Sprint retrospective, Pre-work before Sprint review and demonstration. 4.6 Does you team use User Stories?	RQ-2 RQ-3
4.6.1 Are user stories estimated? 4.6.4 Do they contain acceptance criteria? **Purpose: What Scrum/Large Scale Scrum principles and ceremonies are currently used by small and large Agile teams? Are any of Scrum ceremonies and principles modified? This is to identify Scrum principles and ceremonies that could be applied to Small Scale Scrum.**	RQ-2
5.1 Are you involved in any of the following? Business Requirements Creation, Business Requirements Review and Refinement, Customer negotiation, Customer interviews, Customer handover or training. **Purpose: This is to identify possible barriers for small scale teams with regards to business requirements and customer related activities as part of business pillar.**	RQ-3

TABLE 5.4: Mapping of survey questions back to the research questions cont.

Survey Question	**Research Question**
6.1 Do you think that your Scrum principles help you to reduce development complexity? 6.1.1 If 'Yes', how do you know? i.e. I use a tool, its my inner instinct etc. 6.1.1.1 If you use a tool, please specify what tool do you use? 6.2 How frequently does change request happen in your project (if any)? 6.3 How many support cases do you do every week (if any)? 6.4 Does your team have 'Definition of Done'? 6.4.1 Do you adhere to 'Definition of Done' strictly? 6.5 Does your team use 'Definition of Ready'? 6.5.1 If 'Definition of Ready' is used, do you adhere to it strictly? **Purpose: What quality barriers could be identified from software development activities? This is to identify possible barriers for small scale teams with regards to software development as part of quality pillar.**	RQ-3

TABLE 5.5: Mapping of survey questions back to the research questions cont.

Survey Question	**Research Question**
7.1 Is your team using the following (if any)? Unit Test, Integration Test, Functional Test, Automated Test, Performance/Load Test, Acceptance Test. **Purpose: What quality barriers could be identified from testing centered activities? This is to identify possible barriers for small scale teams with regards to software testing as part of quality pillar.**	RQ-3
8.1 How quickly are committed changes in you project verified? 8.2 How quickly are PRs reviewed? 8.3 Which of the following are the most common causes for delaying completion of your task (if any)? Lack of proper communication between team members, Unplanned absence of team members, Different understanding of solution required, Lack of proper task description, Task blocked by another task in progress, Unnecessary meetings. 8.4 What quality tools/techniques are used by your team (if any)? SonarQube, Code review, Coding standards, Test-driven development, Behaviour-driven development, Unit testing, Automated acceptance testing, Pair Programming, CI/CD, Collective Code Ownership. 8.6 Does Scrum help your team with any of the following? Number of bugs verified within Sprint is greater, Communication between team member(s) is improved, Team collaboration is greater, Successful adaptation of Sprint test approach is possible, Number of bugs from final Regression Test is smaller, Test coverage is greater, Number of bugs from User Acceptance Test is smaller, None. **Purpose: What quality and communication barriers could be identified from quality centered activities? This is to identify possible barriers for small scale teams with regards to software quality as part of quality pillar and team communication and collaboration as part of communication pillar.**	RQ-3
9.1 Do you run a Project Manager over the project?	RQ-1

TABLE 5.6: Mapping of survey questions back to the research questions cont.

Survey Question	**Research Question**
9.5 Do you have Architect involved in the project? 9.7 Do you have Dev Ops team involved in the project? **Purpose: What barriers could be identified for small scale teams? Do similar barriers also exist in regular scrum teams?**	RQ-3
10.1 How would you mark your team's overall collaboration? 10.2 Which of the following communication channels are you currently using in project (if any)? Face to face, Jira, Whiteboard, Email, BlueJeans, Skype, Slack, IRC, Google Drive docs, Confluence, Trello, Collaborative Design/Mockups (e.g Azure, Lucidcharts, Draw.io etc), GitHub/GitLab (PRs discussions). 10.3 Do you have a singular source of truth from the above channels? 10.4 If you trust any of the channel(s) (not necessarily listed above) please specify which channel(s) do you default to? 10.5 How much time do you spend weekly in meetings (if any)? 10.6 How much time do you spend weekly on face-to-face meetings (if any)? 10.7 How much time do you spend weekly on web conferences (if any)?	QR-3

TABLE 5.7: Mapping of survey questions back to the research questions cont.

Survey Question	**Research Question**
10.8 How much time do you spend weekly on email communication (if any)? 10.9 How much time do you spend weekly with the customer (if any)? **Purpose: What communication barriers could be identified from communication centered activities? This is to identify possible barriers for small scale teams with regards to team communication in a project as part of communication pillar.**	RQ-3
11.4 Is your customer respectable of 'no deadlines'? 11.5 Does your customer help create user stories? 11.6 Does your customer participate in any of the following? **Purpose: What business barriers could be identified from business centered activities? This is to identify possible barriers for small scale teams with regards to business as part of business/customer pillar.**	RQ-3

- **Survey Validation** - Survey questions were validated by representatives from both, Red Hat Engineering and Red Hat Consulting. Received feedback was applied before survey was opened to a wider audience.

5.2 Survey Results

A group of 54 participants from Red Hat Mobile, Red Hat Consulting, Red Hat Openshift and Red Hat JBoss participated in the "Small Scale Scrum vs Large Scale Scrum" survey.

Participants were asked to answer 94 short questions relating to their experience with Agile.

Main results of the survey are summarized as follows:

- 96% of survey participants practice a form of Agility, work in distributed teams, think that Scrum principles help them reduce development complexity and that Agile contributes to success of the projects.
- Only 8% of survey participants belong to small 1-3 person teams with 10 out of 51 describing their typical project as short lived (0-3 months).

- Majority of survey participants are Software Engineers, but also Quality Engineers (QE), Project Managers (PM), Product Owners (PO) and ScrumMasters.
- ScrumMaster, PO and Team Members are typical roles present in the projects.
- Nearly half of survey respondents work on/ are assigned to more than 1 project at the same time.
- Almost half of the projects are customer/ value generating vs. improvement/ not directly value generating and unclassified.
- Almost half of survey participants answered that their work is sometimes or most of the time clarified and estimated before development with extensions sometimes or most of the time available.
- Asking for clarification of work items is the team's responsibility.
- Almost half of survey respondents write tests for their code most of the time, adhere to best coding practices, document their code, and always get their code reviewed before merging.
- Almost all of survey participants introduce bugs to the codebase which are prioritized by them, the team, PM, PO, Team Lead or even ScrumMaster.
- Participants ask for help, mentoring when task is complex and take additional roles in their projects when needed including mainly role of Business Analyst, PM, QE and Architect and they find changing roles sometimes difficult.
- When changing roles on daily basis, individuals feel that they lose on average of about 1-2 hours but they still manage to get their work completed on time most of the time.
- Most of survey participants use Scrum (65%), followed by Hybrid (18%) and Kanban (12%). This is in line with the State of Agile Report ***(VersionOne, Inc. (2018))***[32].
- Daily Standup, Sprint, Sprint Planning and Estimating, Backlog Grooming and Sprint Retrospective are among the top Scrum ceremonies and principles followed with team members doing preparation work before meetings.
- Majority of Sprints (62%) are 3 week long, followed by 2 week long Sprints (26%), 1 week long Sprints (6%) and 4 week long Sprints (4%). 2% of participants point out that they are not using Sprints due to strict release and update timings with all activities being organized and planned around those dates.
- Teams use and estimate (story point) User Stories though Poker Planning technique. User Stories contain acceptance criteria.
- Teams have and use 'Definition of Done' mainly in respect of Features and User Stories. 'Definition of Done' is created and thus it belongs to the team and is used to determine completion of a User Story ***(Scrum Inc. (2018))***[53].
- Majority of teams don't have nor use 'Definition of Ready'. The purpose of 'Definition of Ready' is to ensure that the User Stories are actionable, testable and clear ***(Scrum Inc. (2018))***[54].

- Unit, integration, functional, automated, performance/load and acceptance tests are commonly used.
- Overall collaboration between team members is considered as high with team members using various communication channels.
- Majority of survey participants spend more than 4 hours in meetings weekly including face to face, web conferences and email communication.
- Majority of customers are considered as large with half of them understanding and following Scrum principles.
- Customers are respectable of 'no deadlines' most of the time, they sometimes help create User Stories and participate in Sprint Planning, Sprint Review and Demonstration, Sprint Retrospective, and Backlog Review and Refinement.
- Only 27% of survey participants know their customer's level of satisfaction with the adoption of Agile as high. Majority (58%) don't know this information at all.

5.3 Post-Survey Interviews

During the survey process, a few individuals were identified and invited for follow up interviews. The purpose of these interviews was to understand the take on Agile and Scrum in particular from Agile Practitioners perspective in both large and small size teams. It was an opportunity to understand the Scrum strategy within their teams including any modifications made to existing Scrum ceremonies and any barriers to adopting these.

Quality, requirements and pressure with delivering on time were identified as barriers to adopting Small Scale Scrum (SSS) and it was evident from interviews that the same barriers existed in traditional Scrum adoption.

The first interviewee was a ScrumMaster for one team of ten members including Team Lead in large scale project. The team adopted "Kanplan" strategy tailored to their needs. "Kanplan" was a mixed methodology for practicing Agile software development. It's a mix of Kanban and Scrum following all of the traditional Scrum ceremonies with exception for running Sprint Goal and Sprint Review, Backlog Prioritization instead of Backlog Refinement and User Stories not being broken down. Traditional Scrum was unsuitable due to its structured format.

Quality, requirements and pressure with delivering on time were addressed in the first interview as follows:

- Quality - The number of work items delivered and QE reviews performed was limited to a specific number. All work items were verified by two members of the team.
- Requirements (User Stories with Acceptance Criteria) - All functional requirements came from Support and Product team. Any improvement work like automation tasks were created by the Development Team.

- Pressure with delivering on time - The absence of Sprints, Sprint Goals and Clients, removed the time pressure from the team with the exception for occasional support issues.

The second interviewee was a ScrumMaster for one large team initially, split into two teams of eight members each and recently merged again into one team of twelve members in large scale project. The last team size proved difficult to manage. Remoteness of team members added extra layer of complexity in terms of (1) getting the team into meetings without making prior arrangements for example, (2) limiting communication to video tools. On the other hand, having one team allowed to have all required skill set to get the work completed. Traditional Scrum principles and ceremonies were followed with some simple modifications including Sprint Planning duration being changed from recommended 4 hours to 1 hour, Sprint Retrospective and Review being merged, Sprint Review being attended by Development Team and Product Owner, Sprint Retrospective meeting being attended by broader group of participants including other teams, similarly Sprint Demo being turned into a cross team demo.

- Quality - Not considered as implication. Dedicated QE team follows their quality testing process. All members of Development Team were involved in work verification and automation improvements, UAT was performed by QE.
- Requirements (User Stories with Acceptance Criteria) - There was no "Definition of Done". Team members were new with Acceptance Criteria. Team Lead used to write tasks, but then Tello board was used making the process of creating new tasks more manageable. Main user stories were used as a placeholder for several related tasks.
- Pressure with delivering on time - Estimating work at a time level proved very time consuming and was dropped.

Additional implications included:

- Team size - Team size was too large, despite of all needed skill set being in one team. It was much easier to work with smaller teams.
- Remoteness - Remoteness proved difficult in the context of team building and team communication. New tools to improve communication had to be introduced.

The third interviewee was a ScrumMaster for one team of twelve members in medium scale project (7-8 months). Traditional Daily Scrum and Scrum Planning ceremonies were used without any modifications. The team worked in Sprints of two weeks with Wednesday being the start of the Sprint. Sprint Demo, Sprint Retrospective and Sprint Review were merged into one meeting as it was easier for remote Development Team and remote customer to attend one session instead of three. Stakeholders provided feedback during Sprint Demo, Sprint Retrospective and Sprint Review meeting.

- Quality - Platform issue of separate environments not being set up to cater for three teams working on three different projects for the same customer. This resulted in the resources issues within the Platform. Quality team was not able to properly test work completed within early Sprints (6 Sprints) and was

forced to take shortcuts in testing. Situation improved over time, once separate environments were created and Quality team had their own test environment. Setting up CI/CD helped with deploying builds regularly for testing. The initial testing arrangement was having testers in both Red Hat and customer site. Customer involved their own testers three months before the end of the 7-8 month long contract. This inevitably had a quality impact on delivered solution. On the other hand, having Business Analyst and UX/UI Designer working a Sprint ahead helped the team to consider quality before start of development.

- Requirements (User Stories with Acceptance Criteria) - Not all stakeholders were involved in requirements creation in the first six Sprints of the project. The change of PO after six Sprints, introduction of stakeholders and end users had a positive impact on the project. There were lots of gaps in requirements which affected the customer's perspective of quality. The initial six Sprints of work had to be re-written with lots of change requests. The approach was taken to get the original/baseline requirements signed off and start working on multiple change requests. Changing PO helped to get the project moving forward. The new PO could talk to the stakeholders which resulted in the entire solution being changed completely.

- Pressure with delivering on time - Due to difficult start of the project, demo and delivery times changed multiple times. Change requests had to be negotiated against contract duration.

Additional implications included:

- Backlog Prioritization - For the first four Sprints, the Development Team was prioritizing work. The PO took charge of the Backlog prioritization from Sprint five on-wards.

- Team size - Team of twelve members was considered to be too large. It was suggested that in the similar future engagements, the team should be split into smaller teams of (1) DevOps and Infrastructure and (2) Development Team.

The fourth interviewee was a ScrumMaster for one large team of 6 members initially, split into two small teams of 3 members each and recently merged again into one team with the addition of 2 new members. The current size of the team proved difficult to manage. The overall team dynamics changed due to more viewpoints, not sharing information, less communication, working in silos, decreased code quality (regressions caused by inadequate testing before releasing to QE), just to name a few. Remoteness of team members didn't have any impact on the team's performance, but the expansion of the team did have a negative impact. Traditional Scrum principles and ceremonies were followed with some modifications including Sprint Planning duration being changed from recommended 4 hours to 1 hour, Sprint Demonstration, Retrospective, Review and Planning being merged into a single 1.5-2hr long session, Sprint Demonstration, Retrospective, Review and Planning being attended by Development Team, customer representatives (PO and QA/QE).

- Quality - Quality was considered as implication. Dedicated QE team followed their quality testing process shared with all team members. After expansion of the team, it was impossible to get all team members to adhere to the previously

established process. All team members were able to contribute to the automation and improvement tasks, but there was lack of communication between them. The ratio of QEs vs Developers in the team was also inappropriate, 1 (QE) vs. 4 (Developers). Most of the completed User Stories had to be rejected by Quality team on initial testing. Quality team had to engage with the PO during testing to clarify some of the User Stories in test. This improved over time.

- Requirements (User Stories with Acceptance Criteria) - There was no "Definition of Done" and there was no "Definition of Ready". User Stories improved over time by officially giving some of the main Business Analyst responsibilities to a single member of the team.
- Pressure with delivering on time - Estimating work at a complicity level proved difficult at the start. The team was not sure as to whether designs, development and testing should all be included in a single User Story estimate. There was a pressure with delivering on time but it improved over time with modifications to development, quality and delivery processes and changes to Small Scale Scrum ceremonies.

5.4 Projects Selection Process

Five projects with similar duration were selected in order to present project related results accurately.

5.5 Projects Results

This section is about a phased Agile transformation in small scale dedicated team over 1.5 year, starting with Hybrid/Waterfall Agile and successively moving into Scrum methodology. Both, perfect and imperfect transformation moments helped the team reflect and learn from past mistakes and bring more knowledge and experience to the future engagements. The overall transformation process was 1.5 year commitment and was an experimental journey all the way. Challenges faced during Agile transformation included the following:

- Changing the team's and customer's approach towards ***"doing Agile vs being Agile"*** and ***"practices vs mindset"***[34]. Largely Waterfall customer had to be on-boarded with the idea of Agile and challenges behind Agile transformation. Over time all efforts in adoption of Agile paid off and resulted in the customer seeing value in Agile and deciding to go through similar transformation with their business customer. Similarly, getting the team members on board with the idea of Agile transformation was challenging. Lack of experience with Agile and Scrum in particular helped the team to focus on what was important for the team first and foremost. From that moment, the team could modify most suitable Scrum principles and ceremonies.
- Addressing ***"requirements vs deadlines"*** based on existing delivery model:

TABLE 5.8: Project Delivery Model.

Specified Features	Yes	No	Yes
Lack of Requirements	Yes	No	Yes
Set Deadline	Yes	Yes	No
Delivery Outline	Delivering initially agreed features as per estimates provided	Delivering unknown features of final solution	Delivering initially agreed but subject to change and initially omitted features
Delivery Date	Delivering on specific date	Delivering on specific date	Delivering on unknown date

- Establishing the level of customer involvement in Scrum ceremonies as in ***"internal Scrum ceremonies vs customer Scrum ceremonies"***. Due to distribution of the team and the customer, a preparation work was required before Scrum meetings like Sprint Planning, Sprint Retrospective and Demonstration.

TABLE 5.9: Scrum ceremonies participation.

Dedicated Team (Red Hat Consulting)	**Common**	**Customer**
Daily Standups	Project Kick Off	Initial User Stories Creation
Backlog Review	Backlog Review	Backlog Review
	Daily Calls	
	Sprint Planning	
	Sprint Retrospective	
	Sprint Demonstration	
	Lessons Learned	

- ***Targeting the right team members and communication issues*** in Sprint 0 and Sprint Planning in the first Small Scale Scrum team was challenging. The initial idea was to split the team of 6 into 2 teams with 2 Developers and 1 QE in each team. Each team would be responsible for delivering their own solution. Plans changed and the first team of 3 was expanded temporarily by additional 3 members from the second team. Involvement of additional 3 team members who were keen on offering additional viewpoints on every functionality, requirement, wireframe, creative design, was challenging and had a negative impact on the core team. The core Development Team was overshadowed in terms of providing estimates and leading discussions on solution implementations. Overall, this affected the team's performance, communication and interpersonal relationships.
- ***Scaling up the team from small to medium/large*** was a challenge. Modified SSS principles did not withstand team expansion from 3 to 6 team members. This was due to the overall changes in the team dynamics caused by more viewpoints, not sharing information, working in silos, less communication,

decreased code quality (regressions caused by inadequate testing before releasing to QE).

- ***Introducing user story estimation*** in the first SSS team was challenging. Team members were unsure what should be included in the overall user story estimation e.g design, development, testing or just development and testing, while taking into account creating reusable components and streamlining quality testing with introduction of best practices and test automation. In the past, work estimation was not required.

Maturity of skills among team members both social and technical including communication, openness, proactivity, confidence, teamwork as well as growing customer satisfaction and confidence in the team delivering quality work were considered as core factors contributing to the team's transformation success. Importantly, the customer who was going through Agile transformation at the same time, had successfully changed their business and made a significant progress in removing waterfall dependencies.

Common metrics applicable to all Red Hat Consulting projects identified for comparison purposes include:

TABLE 5.10: Comparison metrics for small scale projects delivered by small teams.

Metric Category	**Metric No.**	**Metric Description**	**Indicator**
Business Requirements	BR1	What was the total number of user stories initially in the backlog before start of development?	Indicating the extent of welcoming additional features and scoping issues.
	BR2	What was the number of user stories not fully clarified before development?	Indicating the extent of welcoming change requests.
Project Management	PM1	How many team members were involved in project life-cycle?	Indicating the size of the team.
	PM2	How many Sprints (how much time) did it take to complete the project?	Indicating the size of the project.

TABLE 5.11: Comparison metrics cont.

Metric Category	**Metric No.**	**Metric Description**	**Indicator**
Project Development	PD1	What was the number of issues completed vs committed work (aka team velocity) within Sprint?	Indicating if the team is over committing or if there is excessive scope creep.
	PD2	How many user stories were removed from the Sprint?	Indicating if the team is over committing or if there is excessive scope creep.
	PD3	What was the average ticket development time?	Indicating team's performance.
	PD4	What was the number of issues completed outside of the Sprint?	Indicating issues around estimation of tasks. More tickets could be developed in the Sprint than planned.
	PD5	How many defects were marked as "Known Issue" during project development?	Indicating backlog review & refinement during development.
	PD6	How many user stories were descoped prior to start of project development?	Indicating backlog review & refinement prior to the start of development.
	PD7	How many user stories were descoped during development?	Indicating potential planning issues.

TABLE 5.12: Comparison metrics cont.

Metric Category	**Metric No.**	**Metric Description**	**Indicator**
Project Quality	PQ1	What is the percentage of defects which passed initial test?	Indicating misunderstanding user story's"'Definition of Done".
	PQ2	How many defects were found during development in the Sprint?	Indicating quality level of the source code delivered each Sprint.
Project Release	PR1	How many defects were found after demo/ final release by customer's team?	Indicating quality level of the source code delivered each Sprint.

Projects reviewed and compared in Tables 5.13 and 5.14 included 3 Small Scale Hybrid/Waterfall projects, 1 SSS experimental project which helped to create SSS principles and guidelines, and 1 SSS validation project.

5.5.1 Small Scale Hybrid/Waterfall Project

Three projects have been investigated to help with comparison of projects using SSS Hybrid/Waterfall and SSS principles. Projects A-C were developed prior to Agile transition and their similarities are outlined as follows:

- No Sprints
- Set Deadline
- Set Features
- Majority of User Stories Created
- Big, complex, not always testable User Stories
- No Estimation of User Stories
- Dedicated Business Analyst in addition to Project Manager, Software Developer and Quality Engineer

5.5.2 Small Scale Scrum Experimental Project

The SSS framework originated from Project D and was experimentally adopted therein. This was an Agile Transformation project. Its techniques and methods had to be reshaped multiple times in order to achieve optimal and digestible format for small size teams. The format was driven by the team's experience and through retrospectives, all in line with the top reasons for adopting Agile.

Period of 6 months allowed this small team to study, modify and adopt modified traditional Scrum principles and ceremonies in a real customer engagement including the following:

- No Sprint 0 designed for a project setup, investigation and admin activities
- Introduction of Sprints (2 weeks)
- No Deadline
- Set Features
- Majority of User Stories Created
- Smaller, less complex, testable User Stories
- No Estimation of User Stories
- Introduction of inactive Sprint for issues completed outside of the active Sprint
- Daily Standups
- Daily Calls with the Customer
- Sprint Planning
- Sprint Retrospective
- Sprint Demonstration
- Backlog Refinement
- Smaller team - no dedicated Business Analyst

Project D was considered as a great success for both, the team and for the customer who was previously Hybrid/Waterfall. On-time delivery, business value, customer/user satisfaction, product quality, productivity, project visibility, and process improvement contributed to project success. The SSS techniques and methods had a positive impact on the team's morale, motivation and overall level of satisfaction, which are critical in any successful team.

5.5.3 Small Scale Scrum Validation Project

Project E was the first project using SSS framework. The following practices were adopted:

- Sprint 0 for setting up the project, creating project's skeleton, creating test approach, wireframes and creative designs for Sprint 1, reviewing User Stories in the Backlog. At the start of the project but before Sprint 0, 45 new User Stories were created. During Sprint 0, 15 new User Stories were created.
- Introduction of Sprints (2 weeks)

- During Sprint 1, there was only 1 new User Story created suggesting that introducing Sprint 0 allowed for brainstorming the project and creating most of the User Stories before start of development. Thus, allowed customer feedback.
- Despite of honouring 'no hard deadlines', the customer still asked for a guess project completion estimate so that they can communicate it to their business customer.
- Despite of reviewing and querying requirements on Sprint basis, Sprint 0 and 1 allowed us to review requirements for the entire project. The extent of review was of concern as it allowed for anti-Agile approach towards requirements as opposed to reviewing requirements on Sprint basis with product maturing and welcoming changes. An attempt to fully clarify project requirements can be seen as restricting, and not allowing entire team to participate, but on the other hand it can be used as a pre-work for structured discussions in the future Sprint Planning.
- Giving team members an access to test cases on first day of new Sprint is a welcome change so that they are aware of test scenarios. This can eliminate the risk of tickets failing on initial test. Tests reflect 'Definition of Done' for each User Story in the active Sprint.

5.5.4 Projects Comparison - Statistics

Anonymity of the customers and products delivered by Red Hat Consulting team is preserved in accordance with the European Directive on consumer rights (Directive 2011/83/EU) (known as the Consumer Rights Directive) and General Data Protection Regulation (GDPR).

Projects A-E outlined above are compared in this section based on comparison metrics introduced in Tables 5.10, 5.11 and 5.12. Results presented in tables below show a successful Agile Transformation journey of small, distributed team from project A to project E with project Project E using Small Scale Scrum Framework.

Based on statistics in Tables 5.13 and 5.14 below, project E proved that the team welcomed changes throughout the project. Number of User Stories increased from the initial 61 to 120. Most of the User Stories (60%) were in "Ready" state before start of development. Initially, 3 out of 6 team members were involved in the project.It took 10 Sprints (2 weeks each) to complete the project. Overall velocity of the team was 95%. Overall, 5% of User Stories were not delivered in the project. The average turnover of the tickets in the project was 0.5 ticket /1 day. Only 10% of all issues were completed outside of the Sprints. Only 3 out of all issues (120 User Stories and 50 Defects) were marked as "Known Issues". There were 16 out of 120 User Stories descoped during the entire development. Initially, 0% of Defects passed initial testing. Closer to the end 75% was recorded. On average, 5 Defects were found during development across all Sprints. Overall, 2 Defects were found by the customer's team.

TABLE 5.13: Small Scale Projects Comparison

Met. No.	**Hybrid/ Waterfall**	**Hybrid/ Waterfall**	**Hybrid/ Waterfall**	**Small Scale Scrum Experiment**	**Small Scale Scrum Validation**
	Project A	Project B	Project C	Project D	Project E
BR1	8/12	30/32	52/66	23/60	61/120
BR2	54/8	227/30	311/52	305/23	90/120
PM1	4 (BA, PM, Developer, Tester)	4 (BA, PM, Developer, Tester)	4 (BA, PM, Developer, Tester)	3 (PM, Developer, Tester)	6 (PM, 4 x Developer, Tester)
PM2	No Sprints (18/05/16 - 27/07/17)	No Sprints (04/02/16 - 15/09/16)	9 Sprints (17/04/17 - 16/02/18)	12 Sprints (28/08/17 - 21/03/18)	10 Sprints overall
PD1	No Sprints	No Sprints	Sprint 1(3/3) Sprint 2(1/9) Sprint 3(1/11) Sprint 4(26/43) Sprint 5(13/28) Sprint 6(23/37) Sprint 7(12/27) Sprint 8(76/88) Sprint 9(14/14)	Sprint 1(1/1) Sprint 2(10/10) Sprint 3(11/11) Sprint 4(6/10) Sprint 5(10/10) Sprint 6(13/13) Sprint 7(15/15) Sprint 8(18/21) Sprint 9(19/24) Sprint 10(19/20) Sprint 11(28/28) Sprint 12(11/11)	95% overall
PD2	No Sprints	No Sprints	Sprint 1(10) Sprint 2(7) Sprint 3(11) Sprint 4(0) Sprint 5(0) Sprint 6(0) Sprint 7(1) Sprint 8(5) Sprint 9(0)	Sprint 1(1) Sprint 2(3) Sprint 3(1) Sprint 4(0) Sprint 5(5) Sprint 6(2) Sprint 7(0) Sprint 8(1) Sprint 9(0) Sprint 10(0) Sprint 11(7) Sprint 12(1)	5% overall

TABLE 5.14: Small Scale Projects Comparison cont.

Met. No.	**Hybrid/ Waterfall**	**Hybrid/ Waterfall**	**Hybrid/ Waterfall**	**Small Scale Scrum Experiment**	**Small Scale Scrum Validation**
	Project A	Project B	Project C	Project D	Project E
PD3	2-3 days	2-3 days	2-3 days	3-5 days	0.5 day
PD4	No Sprints	No Sprints	None	Sprint 6(12) Sprint 8(2) Sprint 10(3) Sprint 12(1)	10/120
PD5	6/73	1/260	0/272	10/208	3/170
PD6	0/12	0/32	0/66	1/60	0/120
PD7	1/12	0/32	1/66	0/60	16/120
PQ1	60%	60%	60%	70%	80%
PQ2	No Sprints	No Sprints	20-25	15-20	5
PR1	- After final release - 5-6 - 1-3 in the last	- After final release - 5-6 - 1-2 in the last	- After final release - 1-2 - None in the last	- After each demo build - 1-2 - None in the last	2 overall

Chapter 6

Conclusion and Future Work

In this chapter the thesis is concluded with some potential areas for future work. All RQs answered in this chapter are based on both academic and strong practical experience.

6.1 Conclusion

6.1.1 Thesis Summary

Chapter 2 presented an overview of the literature involved in formulation of Small Scale Scrum (SSS) framework including Agile Manifesto, Large Scale Scrum (LeSS), large teams, SSS, small teams, and barriers to SSS. The area of SSS was first reviewed in the context of Agile Manifesto, its Scrum ceremonies and principles in regular and large teams and was seen as being helpful with the team's focus on delivering product and communication improvement. A comprehensive review of SSS and Small teams was discussed, showing absence of a definition for small scale teams and small scale projects and weakness on the adoption of SSS in open source software projects delivered by small distributed team. Literature agreed with the need of modifying Agile methods for small projects. The final section discussed considering most of the barriers to Scrum and LeSS as candidate barriers to SSS.

In Chapter 3, the four core SSS principles were created alongside with a comprehensive collection of guidelines addressing communication, quality and business aspects.

In Chapter 4, Small Scale Agile Manifesto and SSS framework were created and proposed for small scale, open source, distributed teams. Created methodology was based on Agile and Scrum in particular and was considered as hybrid variation of Scrum. Ceremonies and roles in traditional and SSS were compared.

Chapter 4 looked at the SSS survey and validation of SSS in real customer projects. Survey design process, rationale and results were discussed. Selected survey questions were mapped to research questions and survey results were analyzed and conclusions drawn. Key projects were analyzed showing Agile transformation of open source, distributed and small scale team for the purposes of validating SSS framework.

6.1.2 Contributions

The usage of SSS will continue to expand across consultancy services globally. A significant force for this expansion will be continued distribution of business and resources, competitive delivery times and projects duration, all tightly related to companies budgets. In this regard, small scale teams and SSS will be at the center of attention. This thesis proposes SSS as a novel model for delivering software solutions by small and geographically distributed teams in a fast paced and challenging environment. This approach contains a number of contributions, arrived at from answering relevant RQs posed:

- *Formulation of Small Scale Agile Manifesto:* This represents a contribution to the understanding of Agile and its Scrum methodology and structuring the core values of Agile software development required to successfully facilitate small scale distributed teams in small scale project implementation. Research Question RQ-1 was partially addressed by this contribution.
- *Formulation of Small Scale Scrum Framework:* The SSS ceremonies and principles that represent modifications made to the traditional Scrum ceremonies and are capable of improving software delivery process by way of identifying and understanding implications existing in traditional Scrum for small teams in communication, quality and business areas. Research Questions RQ-1 and RQ-2 were addressed by this contribution.
- *Identification of barriers to adopting Small Scale Scrum:* This represents a contribution to identification of barriers including improper communication, lack of knowledge sharing and multinational roles in order to evolve SSS framework. Research Question RQ-3 was addressed by this contribution.

Table 6.1 provides a reference to the chapters where Research Questions were investigated.

TABLE 6.1: Research Question Chapter Reference Table

Research Question	Thesis Chapter
RQ-1 Does Agile Manifesto with its Scrum methodology stand for small scale distributed teams and small scale project implementations?	Chapters 1 (1.3), 2 (2.1), 4, 5
RQ-2 What Scrum principles and ceremonies can be applied to Small Scale Scum, if any, with an emphasis on any implications for Small Scale Scrum Teams?	Chapters 1 (1.3), 2 (2.1), 3, 5
RQ-3 What are barriers to adopting Small Scale Scrum? Do similar barriers also exist in regular Scrum teams that are trying to scale or is this a scale specific set of issues?	Chapters 1 (1.3), 2 (2.3), 5

6.1.3 Conclusions

RQ-1 Does Agile Manifesto with its Scrum methodology stand for small scale distributed teams and small scale project implementations?

Yes, the Agile Manifesto and Scrum methodology do stand for small scale distributed teams and small scale project implementations. This is based on the review of six Agile methodologies including Waterfall, Hybrid, Kanban, Scrum (6 weeks Sprints), Scrum (2 week Sprints) and Hyper Agile revealing Scrum (2 weeks Sprints) as the most suitable Agile methodology for small teams.

The following is a short reflection on factors and Agile methodologies used for consideration when deciding on the right methodology for the team (here small scale team of a maximum of 3 people) ***dreamcatcher (2017)***[35]:

- ***Size and Growth of the team*** is considered as an important factor for deciding on the most suitable Agile methodology for the team. Small team size is considered as a maximum of 3 people team, but it's also important to consider the prospect of the team changing its size (scaling up or down) during the project life-cycle before project commencement.
- ***Communication between team members*** is equally important factor mainly due to team's remoteness element. The right methodology should enable continuous and as much real time communication between team members as possible.
- ***Work Quality*** needs to be considered in context of team's ability to deliver quality and is related to team's size and skills. Desired methodology should enable incorporating quality testing throughout the project development. Scrum with 2 weeks Sprints is seen as sufficient to produce well tested builds and achieve visible progress in implementing features.
- ***Core Competency of team members*** which is writing software, positions it well for Agile methods.
- ***Scrum*** is a popular methodology with work items completed and delivered within Sprints. Sprint duration can range between 2-6 weeks.

RQ-2 What Scrum principles and ceremonies can be applied to Small Scale Scrum, if any, with an emphasis on any implications for Small Scale Scrum Teams?

Modified Daily Scum (but only if team size is greater than 1 person), Sprint Planning, Sprint Review and Sprint Retrospective meetings can be applied to SSS subject to quality, requirements and time pressure implications commonly applicable to large and small teams. A complete comparison of ceremonies in both, traditional Scum and SSS is presented in Chapter 4 Table 4.2.

The approach taken in relation to Scrum modifications is to allow for communication, quality and business pillars to be incorporated into Scrum practices and ceremonies. Importantly, trust and communication (and leadership, all known as "The Three Laws of Influence") go hand in hand and in SSS, building trust with team

members and customers through communication (and leadership) is critical. Excellence in listening and effective communication is key to success. Similarly, high quality assurance throughout project development and project management life-cycle is required to improve customer satisfaction. And last but not least, understanding customers, their business and needs is the final success factor.

Ability to listen and communicate in SSS means:

- Structured communication to allow for limited viewpoints taken into consideration including communication with the customer and internal communication between team members.
- Early communication when things go and don't go according to the plan including seeking help/assistance at any stage of the project, expressing project related updates, concerns, blockers and risk factors.
- Asking questions and not assuming project related aspects including requirements.
- Reviewing, providing, asking for and accepting feedback during requirements clarification sessions. finalizing design sessions, test approach sessions and demos.
- Communication on achievements to allow team to celebrate successes and impediments to outline nature of the problem(s)/issue(s) early.

Quality in SSS means:

- Quality oriented team including conversations/ calls/ meetings with preparation work beforehand.
- Quality projects including quality requirements and acceptance criteria (independent, negotiable, valuable, estimable, small and testable) and quality documentation (Readme.md, documentation generated from comments in code etc).
- Quality development techniques and use of tools including the use of a version control system for tracking changes (feature branches, regular commits, tags etc), CI/CD, quality Sprint builds (passing smoke test at very minimum) and final solution deliveries. Quality code (readable, easily extendable, refactored, 'Big Ball of Mud' avoided at all cost [37])
- Quality testing (unit test, integration test, functional test, automated test, performance/load test, acceptance tests), quality issues/bugs descriptions (clearly defined, with detailed acceptance criteria/ DOD).
- Quality work organization (Sprints structure clearly outlining development, testing, demo (progress update and feedback oriented) and delivery phases)

Business in SSS means:

- Relationship with customer and customer team
- Business generating software solution(s)
- Analysis of any gaps in software requirements

- End users testing (UAT)
- Acceptance and review of feedback

The SSS requires projects with a small scope. Small scale teams tend to be distributed in terms of location and projects (team members can be allocated to more than one assignment). Small teams are excellent and coaching driven in terms of quality and best practices. Under the hood small teams work as large scale teams but with less resourcing, quality and cost constraints.

This Research Question is validated through a success of a small, dedicated team in leading Agile transformation and coaching of long-term remote customer on Agile practices and processes to maximize performance, quality and overall satisfaction. It took three years before the team reached the state of mixed Hybrid and Waterfall methodology and then another year to fully transition into Agile and now to be in a position to run the latest project with SSS. Scrum ceremonies and practices have been considerably reduced to the bare minimum. In customer projects, time is of an essence and the team had to stick rigorously to the agreed format of ceremonies. Customer could apply practices to their team's transformation happening in parallel.

Approach to simplified Scrum based on traditional Scrum explained in detail in section 4.2 and is as follows:

- Daily Scrum should be practiced provided that the team size is at least 2-3 members. Daily Scrum is not recommended for 1 person team.
- Sprint Planning should be included but with preparation work completed beforehand.
- Sprint Retrospective should be practiced.
- Sprint Review should be practiced.

Importantly, application of techniques and tools exploiting communication, high quality and business aspects in SSS is required.

RQ-3 What are the barriers to adopting Small Scale Scrum? Do similar barriers also exist in regular Scrum teams that are trying to scale or is this a scale specific set of issues?

Identified barriers to SSS are listed and explained in Appendix D. The top three barriers are as follows:

1. **Improper communication** - Clear communication provides clear direction. can be supported by the use of suitable communication channels. Sufficient task description, the same understanding of solution required, willingness to initiate or participate in requirements clarifying meetings, discipline for the avoidance of assumptions and efficient communication of task related progress. They all contribute to successful communication within and outside of the team. The benefit of effective communication is genuine motivation, team unity and common vision resulting in high productivity, happy team members and the willingness to go the extra mile.

2. **Lack of knowledge sharing** - Knowledge sharing makes problem-solving experiences reusable helping individuals to save time and prevent them from reinventing known solutions, enables better and faster decision making, stimulates innovation and growth, improves delivery to customers and reduces the loss of know-how.

3. **Multifunctional roles** - Changing tasks on daily basis proves to be difficult and is associated with a loss of time required on getting up the speed with the project and codebase prior to any implementation of change requests. Multifunctional roles require ability to act as Business Analyst, Developer, Tester and Delivery Manager as one. Structured roles can fundamentally improve productivity and quality of delivered solution.

Furthermore, post-survey interviews discussed in Chapter 5 Section 5.3 reveal that quality, requirements and time pressure barriers to adopting SSS also exist in traditional Scrum.

6.2 Future Work

Future work will center around further dissemination and application of SSS framework falling within the Three Pillars of Empiricism (Scrum) ***(Scrum.org (2018))***[45]. The Three Pillars of Empiricism is a progress focused on investigations of real world for the purposes of reaching business and organizational Agility. Transparency, Inspection and Adaptation are the three pillars.

Activities will include:

- Dissemination through presentation at Agile Cambridge 2017 Conference ***(Agile Cambridge (2017))***.

  Transparency - The hypothesis of this thesis was presented and validated with Agile Practitioners at Agile Cambridge 2017 Conference.

  Inspection - The hypothesis was openly discussed with Conference participants in order to gather valuable feedback. Received feedback was very positive and didn't require any change of hypothesis.

  Adaptation - Feedback was reviewed for continuous improvement purposes and to show the ability to adapt based on the results of the inspection.

- Dissemination through presentation at Agile-Lean Ireland 2019 Conference ***(Agile Conference (2018))***[44].

  Transparency - The SSS framework, its principles, techniques and methods will be presented at the Agile-Lean Ireland 2019 Conference.

  Inspection - The SSS framework, specific principles, techniques and methods will be discussed with Conference participants in order to gather feedback.

  Adaptation - Received feedback will be reviewed and adopted for continuous improvement purposes.

- Publication strategy through journal and technical blogs in an open source manner to provide updates to Literature Review.

  Transparency - The SSS framework will be further disseminated through well approached open source publication strategy.

Inspection - The SSS framework will be published in appropriate journals and technical blogs. Feedback will be asked for in blogs in order to gather feedback.

Adaptation - Received feedback will be reviewed and adapted.

- Revisiting and broadening survey and interviews utilization.

  Transparency - The "Small Scale Scrum vs. Large Scale Scrum" survey will be further disseminated through improvement and distribution outside of Red Hat organization.

  Inspection - The survey will be revisited, broadened and distributed to other organizations in order to gather feedback from survey participants.

  Adaptation - Received results will be reviewed and adapted.

- Potential application for PhD.

  Transparency - Elements of the SSS framework will potentially be considered for PhD application.

  Inspection - Elements of the SSS framework will be proposed as a potential PhD topic in order to gather feedback from supervisors/examiners.

  Adaptation - Received feedback will be reviewed and adapted for a potential PhD application.

- Application of SSS framework on real world customers on-site.

  Transparency - The framework will be presented and applied to several real world customers on-site in the future.

  Inspection - Practical application of framework in several projects in 2019 will contribute to its constant update.

  Adaptation - Findings will be reviewed and adapted.

Chapter 7

References

[1] Overview - Large Scale Scrum (LeSS). 2018. Overview - Large Scale Scrum (LeSS) . [ONLINE] Available at: https://less.works/. [Accessed 26 March 2018].

[2] Statista. 2018. Technology consulting market size 2016 | Statistic. [ONLINE] Available at: https://www.statista.com/statistics/624501/global-technology-consulting-market-size/. [Accessed 12 April 2018].

[3] Agile Cambridge. 2017. Agile Cambridge. [ONLINE] Available at: http://agilecambridge.net/2017/. [Accessed 4 March 2018].

[4] RGalen Consulting. 2018. The 3 Bears of Agile Team Size — RGalen Consulting. [ONLINE] Available at: http://rgalen.com/agile-training-news/2015/8/22/the-3-bears-of-agile-team-size. [Accessed 01 April 2018].

[5] Schwaber, Sutherland, KS, JS, 2013. The Scrum Guide, The Definitive Guide to Scrum: The Rules of the Game. [Online]. Available at: http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-US.pdf [Accessed 1 May 2018].

[6] Manifesto for Agile Software Development . 2018. Manifesto for Agile Software Development . [ONLINE] Available at: http://agilemanifesto.org/. [Accessed 02 April 2018].

[7] Khmelevsky, Li, Madnick, YK, XL, SM, 2017. Software Development Using Agile and Scrum in Distributed Teams. 2017 Annual IEEE International Systems Conference (SysCon), [Online]. Unknown, 1-4. Available at: https://ieeexplore.ieee.org/document/7934766/ [Accessed 7 February 2018].

[8] VersionOne, Inc. 2018. CollabNet VersionOne. [ONLINE] Available at: https://explore.versionone.com/state-of-agile. [Accessed 04 April 2018].

[9] Kapitsaki, Christou, GMK, MC, 2015. Where Is Scrum in the Current Agile World?. 2014 9th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE), [Online]. Unknown, 1-8. Available at: https://ieeexplore.ieee.org/document/7077124/ [Accessed 6 February 2018].

[10] Dingsøyr, Fægri, Itkonen, TD, TEF, JI, 2018. What is Large in Large-Scale? A Taxonomy of Scale for Agile Software Development. Product-Focused Software Process Improvement, [Online]. 8892, 273-276. Available at:
https://pdfs.semanticscholar.org/4930/03cd2be6e79d4d214846c785bea9122c62a3.pdf
[Accessed 16 February 2018].

[11] Dingsøyr, T. and Moe, N. B., "Research Challenges in Large-Scale Agile Software Development," ACM Software Engineering Notes, vol. 38, pp. 38-39, 2013.

[12] QuickScrum. 2018. Scrum Ceremonies | Scrum Guide | QuickScrum. [ONLINE] Available at: https://www.quickscrum.com/Help/184/sg-Scrum-Ceremonies. [Accessed 05 April 2018].

[13] Scrum Alliance - Scrum Roles Demystified . 2018. Scrum Alliance - Scrum Roles Demystified . [ONLINE] Available at: https://www.scrumalliance.org/agile-resources/scrum-roles-demystified. [Accessed 05 April 2018].

[14] Webster's, Encyclopedic Unabridged Dictionary of the English Language. New York: Gramercy Books, 1989.

[15] Paasivaara, Durasiewicz, Lassenius, MP, SD, CL, 2008. Distributed Agile Development: Using Scrum in a Large Project. 2008 IEEE International Conference on Global Software Engineering, [Online]. Unknown, 87-95. Available at:
https://ieeexplore.ieee.org/document/4638656/ [Accessed 15 February 2018].

[16] Ramesh, B. Cao, L., Mohan, K. and Xu, P. Can distributed software development be agile? Communications of the ACM, vol. 49(10), pp. 41-46, 2006.

[17] Simons, M. Internationally Agile. InformIT, March 15th, 2002.

[18] Sutherland, J., Viktorov, A., Blount, J. and Puntikov, N. Distributed scrum: Agile project management with outsourced development teams. Proceedings of the 40th Annual Hawaii International Conference on System Sciences, HICSS 2007, pp. 274a-274a.

[19] Layman, L., Williams, L., Damian, D. and Bures, H. Essential communication practices for extreme programming in a global software development team. Information and Software Technology, vol. 48, no. 9, 2006, pp. 781-794.

[20] Berczuk, S., Back to basics: The role of agile principles in success with an distributed scrum team. Proceedings of AGILE, 2007, pp. 382-388.

[21] Fowler, M. Using an agile software process with offshore development. 2006 (Available: http://martinfowler.com/articles/agileOffshore.html)
Referenced: 19.12.2007.

[22] Rising, Janoff, LR, NSJ, 2000. The Scrum software development process for small teams. IEEE Software, [Online]. 17, 26-32. Available at:
https://ieeexplore.ieee.org/document/854065/ [Accessed 30 January 2018].

[23] Zhang, Dorn, XZ, BD, 2011. Agile Practices in a Small-Scale, Time-Intensive Web Development Project. IEEE Computer Society, [Online]. 187, 1060-106. Available at: https://ieeexplore.ieee.org/document/5945388/ [Accessed 4 March 2018].

[24] Lee, Yong, SL, HY, 2013. Agile Software Development Framework in a Small Project Environment.[Online].69-88. Available at: https://www.semanticscholar.org/paper/Agile-Software-Development-Framework-in-a-Small-Lee-Yong/43328ce79b239b0373b38df8f6cad06c7d076c34 [Accessed 11 March 2018].

[25] Rajkumar, Selvakumar, Ranjith, RN, SJ, RB, 2014. Extended Scrum method of Agile Practice for Small Scale Project Development. International Journal of Innovative Research in Science, Engineering and Technology, [Online]. 3, 374-377. Available at: http://www.rroij.com/open-access/extended-scrum-method-of-agile-practice-forsmall-scale-project-development.pdf [Accessed 17 February 2018].

[26] Micro-Scrum: A stamp-sized version of Scrum » Thinking Inside a Bigger Box. 2018. Micro-Scrum: A stamp-sized version of Scrum » Thinking Inside a Bigger Box. [ONLINE] Available at: http://johannesbrodwall.com/2013/10/20/micro-scrum-a-stamp-sized-version-of-scrum/. [Accessed 09 April 2018].

[27] Ray Wenderlich. 2018. Scrum Of One: How to Bring Scrum into your One-Person Operation. [ONLINE] Available at: https://www.raywenderlich.com/162654/scrum-one-bring-scrum-one-person-operation. [Accessed 09 April 2018].

[28] Poole, C. J. Distributed product development using extreme programming. Proceedings of the 5th International Conference, XP 2004, Garmisch-Partenkirchen, Germany, June 6-10, 2004. Springer Berlin/Heidelberg, pp. 60-67.

[29] Moe, Dingsøyr, Dybå, NBM, TD, TD, 2009. Overcoming Barriers to Self-Management in Software Teams. IEEE Software, [Online]. Volume: 26, Issue: 6, 20-26. Available at: https://ieeexplore.ieee.org/abstract/document/5287005/ [Accessed 10 April 2018].

[30] She, MS, 2018. Virtual Team Dynamics: Issues and Resolutions. [Online]. 1-23. Available at: https://pdfs.semanticscholar.org/c6a5/fd161b9ab32d89c9bc9433eef9ab5fb3939d.pdf [Accessed 9 April 2018].

[31] Scrum Alliance - The Scrum Guide . 2018. Scrum Alliance - The Scrum Guide . [ONLINE] Available at: https://www.scrumalliance.org/learn-about-scrum/the-scrum-guide. [Accessed 02 April 2018].

[32] VersionOne, Inc. 2018. CollabNet VersionOne. [ONLINE] Available at: https://explore.versionone.com/state-of-agile/versionone-12th-annual-state-of-agile-report. [Accessed 27 April 2018].

[33] Scrum Alliance - Scrum Values . 2018. Scrum Alliance - Scrum Values . [ONLINE] Available at:
https://www.scrumalliance.org/learn-about-scrum/scrum-values. [Accessed 27 April 2018].

[34] Michael Sahota - The Sahota System for High Performance Leadership. 2018. Doing Agile vs. Being Agile - Michael Sahota - The Sahota System for High Performance Leadership. [ONLINE] Available at: http://agilitrix.com/2016/04/doing-agile-vs-being-agile/. [Accessed 13 April 2018].

[35] Where does your team fit on the Agile continuum? –. 2018. Where does your team fit on the Agile continuum? –. [ONLINE] Available at:
http://dreamcatchersoftware.com/agile-continuum/. [Accessed 23 April 2018].

[36] Fernandes, Almeida, JMF, MA,, 2010. Classification and Comparison of Agile Methods. 2010 Seventh International Conference on the Quality of Information and Communications Technology, [Online]. Unknown, 391-396. Available at:
https://ieeexplore.ieee.org/document/5655659/ [Accessed 23 January 2018].

[37] Big Ball of Mud. 2018. Big Ball of Mud. [ONLINE] Available at:
http://www.laputan.org/mud/. [Accessed 16 April 2018].

[38] 6 Main Principles of Scrum Methodology | SCRUMstudy Blog. 2018. 6 Main Principles of Scrum Methodology | SCRUMstudy Blog. [ONLINE] Available at:
http://blog.scrumstudy.com/6-main-principles-of-scrum-methodology/. [Accessed 26 April 2018].

[39] Niinimaki, TN,, 2011. Face-to-face, Email and Instant Messaging in Distributed Agile Software Development Project. 2011 Sixth IEEE International Conference on Global Software Engineering Workshops, [Online]. Unknown, 78-84. Available at:
https://ieeexplore.ieee.org/document/6065582/ [Accessed 25 January 2018].

[40] Scrum Alliance - 2018 State of Scrum . 2018. Scrum Alliance - 2018 State of Scrum . [ONLINE] Available at: https://www.scrumalliance.org/learn-about-scrum/state-of-scrum/2018-state-of-scrum. [Accessed 17 April 2018].

[41] Stack Overflow. 2018. Stack Overflow Developer Survey 2018. [ONLINE] Available at: https://insights.stackoverflow.com/survey/2018/. [Accessed 14 August 2018].

[42] 2018 Developer Skills Report by HackerRank. 2018. 2018 Developer Skills Report by HackerRank. [ONLINE] Available at: https://research.hackerrank.com/developer-skills/2018/. [Accessed 14 August 2018].

[43] GitLab. 2018. GitLab 2018 Global Developer Report | GitLab . [ONLINE] Available at: https://about.gitlab.com/developer-survey/2018/. [Accessed 14 August 2018].

[44] Agile-Lean Ireland 2019 Conference | Dublin, April 24th-26th 2019. 2018. Agile-Lean Ireland 2019 Conference | Dublin, April 24th-26th 2019. [ONLINE] Available

at: https://www.agileleanireland.org/. [Accessed 17 August 2018].

[45] Scrum.org. 2018. The Three Pillars of Empiricism (Scrum) | Scrum.org. [ONLINE] Available at: https://www.scrum.org/resources/blog/three-pillars-empiricism-scrum. [Accessed 17 August 2018].

[46] Hunt, A, Thomas, D, 1999. The Pragmatic Programmer. 2nd ed. United States of America: Addison Wesley Longman, Inc.

[47] Krutchen, P, 2003. The Rational Unified Process – An Introduction (3rd edition). Reading, MA: Addison-Wesley.

[48] Boehm, B, 1988. A Spiral Model of Software Development and Enhancement. IEEE Computer, 21 (5), 62-72.

[49] IEEE Future Directions. 2018. Do you still remember the Waterfall Model? – IEEE Future Directions. [ONLINE] Available at: http://sites.ieee.org/futuredirections/2018/01/26/do-you-still-remember-the-waterfall-model/. [Accessed 18 August 2018].

[50] Scrum.org. 2018. Scrum Guide Update November 2017 | Scrum.org. [ONLINE] Available at: https://www.scrum.org/resources/blog/scrum-guide-update-november-2017. [Accessed 18 August 2018].

[51] Scrum Guide | Scrum Guides. 2018. Scrum Guide | Scrum Guides. [ONLINE] Available at: https://www.scrumguides.org/scrum-guide.html. [Accessed 19 August 2018].

[52] Agile Alliance. 2018. What are the Three Questions in Agile? | Agile Alliance. [ONLINE] Available at: https://www.agilealliance.org/glossary/three-qs/. [Accessed 25 August 2018].

[53] Scrum Inc. 2018. Definition of Done - Scrum Inc. [ONLINE] Available at: https://www.scruminc.com/definition-of-done/. [Accessed 26 August 2018].

[54] Scrum Inc. 2018. Definition of Ready | Scrum Inc.. [ONLINE] Available at: https://www.scruminc.com/definition-of-ready/. [Accessed 26 August 2018].

[55] Medium. 2018. Small Scale Agile Manifesto – Agnieszka Gancarczyk – Medium. [ONLINE] Available at: https://medium.com/@agagancarczyk/small-scale-agile-manifesto-a628ae1c0dfe. [Accessed 31 August 2018].

[56] Medium. 2018. Small Scale Scrum Framework – Agnieszka Gancarczyk – Medium. [ONLINE] Available at: https://medium.com/@agagancarczyk/small-scale-scrum-framework-5165c9a3d6b7. [Accessed 31 August 2018].

[57] Medium. 2018. Small Scale Scrum Principles – Agnieszka Gancarczyk – Medium. [ONLINE] Available at: https://medium.com/@agagancarczyk/small-scale-scrum-principles-fdf8567eb424. [Accessed 31 August 2018].

Appendix A

Questionnaire Survey

This questionnaire is part of the MSc in Computing (Communications Software), Department of Computing, Mathematics and Physics Waterford Institute of Technology.

A.1 General Questions

1.1 Does your team practice a form of Agility?
a) Yes
b) No
c) Maybe

1.2 What is your current team size for an Agile Project?
a) 1-3 people
b) 3-9 people
c) 10+ people

1.3 Is your current team distributed (if at least 1 team member is working remotely?)
a) Yes
b) No

A.2 Project Size

2.1 Which one best describes your typical project duration?
a) Small (0-3 months)
b) Medium (3-9 months)
c) Large (9+ months)

A.3 Team Roles

3.1 Are you frequently involved in more than 1 project? I.e assigned to more than 1 team?
a) Yes
b) No

3.1.1 If 'Yes', please indicate the number of projects
_____ (Short answer text)

3.1.2 If 'Yes', are these customer projects or sustaining/maintenance based projects? E.g. technical debt reduction, automation?
a) Customer/ value generating
b) Improvement/ not directly value generating
c) N/A

3.1.3 Is work estimated before start of development?
a) Never
b) Sometimes
c) Most of the time
d) Always

3.1.3.1 If 'Never/Sometimes/Most of the time', are extensions available?
a) Never
b) Sometimes
c) Most of the time
d) Always

3.2 Are tasks fully clarified before start of development?
a) Never
b) Sometimes
c) Most of the time
d) Always

3.2.1 If 'Never', do you ask customer/Product Owner for clarification before start of development?
a) Yes
b) No

3.2.1.1 If 'No', who does ask for clarification?
a) Project Manager
b) Other _____ (text)

3.3 Does the customer clarify requirements timely?
a) Never
b) Sometimes
c) Most of the time
d) Always

3.3.1 If not 'Always', how long does it take to receive clarifications?
a) Hour
b) Day
c) Hours
d) Days
e) Other _____ (text)

3.4 Do you write tests for your code?
a) Never

b) Sometimes
c) Most of the time
d) Always

3.5 Is your code reviewed before merging?
a) Never
b) Sometimes
c) Most of the time
d) Always

3.6 Do you adhere to best coding practices?
a) Never
b) Sometimes
c) Most of the time
d) Always

3.7 Is your code documented?
a) Never
b) Sometimes
c) Most of the time
d) Always

3.8 How often do you have to add changes to your code on initial code review and verification?
a) Never
b) Sometimes
c) Most of the time
d) Always

3.9 Does your code introduce bugs to the codebase?
a) Never
b) Sometimes
c) Most of the time
d) Always

3.10 Does your code require customer clarification when in test?
a) Never
b) Sometimes
c) Most of the time
d) Always

3.11 Do you ask for help/mentoring if task is complex?
a) Never
b) Sometimes
c) Most of the time
d) Always
e) I don't have access to a mentor

3.12 What Scrum roles are present in your current project?
a) ScrumMaster
b) Product Owner

c) Team Member(s)
d) Other _____ (text)

3.13 What is your current primary role in the team?
a) Software Engineer
b) Quality Engineer
c) Project Manager
d) Product Owner
e) ScrumMaster
f) Other _____ (text)

3.14 Are you taking additional roles in your current project?
a) Yes
b) No

3.15.1 What other role(s) do you fulfill in a typical project?
a) Quality Engineer
b) Business Analyst
c) Project Manager
d) Architect
e) Other _____ (text)

3.15.2 Do you feel you have the right skills to cover all of these roles?
a) Yes
b) No

3.15.2.1 If you don't have the right skills, do you have an opportunity to up skill?
a) Never
b) Sometimes
c) Most of the time
d) Always

3.15.3 How difficult is changing tasks on a daily basis?
a) 0 (Easy)
b) 1
c) 2
d) 3
e) 4
f) 5
g) 6
h) 7
i) 8
j) 9
k) 10 (Very Difficult)

3.15.4 How much time do you feel you lose (if any)?
a) 0-1 hours
b) 1-2 hours
c) 2+ hours

3.15.5 Do you get your work completed on-time?

a) Never
b) Sometimes
c) Most of the time
d) Always

A.4 Agile & Scrum Principles & Ceremonies

4.1 How long are you practicing Agile?

a) I am not actively practicing it
b) 0-1 years
c) 1-2 years
d) 2-5 years
e) 5+ years

4.2 Which agile software development methodology is followed by your team?

a) Scrum
b) Kanban
c) Lean
d) Hybrid
e) Other _____ (text)

4.2.1 If you follow Hybrid, please describe

_____ (Long answer text)

4.3 Which of the following Scrum ceremonies & principles is your team practicing (if any)?

a) Daily Standup
b) Sprint
c) Sprint Planning & Estimating
d) Sprint Code Freeze
e) Minimum testing before QE
f) Sprint QE Testing
g) Backlog Grooming
h) Sprint Retrospective
i) Other _____ (text)

4.4 Does your team use any of the following?

a) Sprint 0
b) Additional Sprint (extra Sprint created during active Sprint for additional user stories completed in the active Sprint)
c) Hardening Sprint (extra Sprint at the end of the project)
d) Pre-work before Sprint planning
e) Pre-work before Sprint retrospective
f) Pre-work before Sprint review & demonstration

4.5 How long are your Sprints?

a) 1 week
b) 2 weeks

c) 3 weeks
d) 4 weeks
e) >4 weeks
f) Other _____ (text)

4.6 Does your team use User Stories?
a) Yes
b) No

4.6.1 Are user stories estimated?
a) Never
b) Sometimes
c) Most of the time
d) Always

4.6.2 What unit of measure do you use?
a) Story Points
b) Hours
c) T-shirt sizes

4.6.3 What estimation and planning technique is your team using?
a) Poker Planning
b) The Bucket System
c) Big/Uncertain/Small
d) Dot Voting
e) T-Shirt Sizes
e) Other _____ (text)

4.6.4 Do they contain acceptance criteria?
a) Never
b) Sometimes
c) Most of the time
d) Always

4.6.5 Who captures acceptance criteria?
a) ScrumMaster
b) Product Owner
c) Team Member
d) Team Lead
e) Other _____ (text)

A.5 Software Requirements

5.1 Are you involved in any of the following?
a) Business Requirements Creation
b) Business Requirements Review & Refinement
c) Customer negotiation
d) Customer interviews
e) Customer handover/ training

f) Other _____ (text)

A.6 Software Development

6.1 Do you think that your Scrum principles help you to reduce development complexity?
a) Yes
b) No
c) I don't know

6.1.1 If 'Yes', how do you know?
a) I use a tool
b) It's my inner instinct

6.1.1.1 If you use a tool, please specify what tool do you use?
_____ (Long answer text)

6.2 How frequently does change request happen in your project (if any)?
a) 0
b) 1 per day
c) 1 per week
d) 1 per month
e) Other _____ (text)

6.3 How many support cases do you do every week (if any)?
a) 0
b) 1-3
c) 3+

6.4 Does your team have 'Definition of Done'?
a) Yes
b) No
c) I don't know

6.4.1 Do you adhere to 'Definition of Done' strictly?
a) Never
b) Sometimes
c) Most of the time
d) Always

6.4.2 Does 'Definition of Done' refer to any of the following?
a) Feature
b) Sprint
c) Story
d) Release
e) Other _____ (text)

6.4.2.1 Does 'Definition of Done' for Feature consist of any of the following (if any)?

a) Code produced
b) Code documented
c) Code checked into source control
d) Builds without errors
e) Static code analysis done and issues fixed
f) Unit tests written and passing
g) Peer reviewed
h) Deployed to test/staging environment and passed system test
i) Any build/deployment/configuration changes implemented/documented and communicated
j) Relevant documentation produced and/or updated (User Documentation)
k) Other _____ (text)

6.4.2.2 Are there any elements of 'Definition of Done' for Feature you disagree with?
a) Yes
b) No
c) Other _____ (text)

6.4.2.3 If you disagree with 'Definition of Done' for Feature, what elements do you disagree with and why?
_____ (Long answer text)

6.4.3 Does 'Definition of Done' consist any of the following?
a) The feature/bug fix implemented - code is complete, documented and checked in
b) Unit tests written and running cleanly in the CI environment
c) Integration and/or E2E tests written and running cleanly in the CI environment
d) Non-functional tests (performance, security etc) written and running cleanly in the CI environment
e) Code changes reviewed by at least one peer
f) Acceptance criteria are verified and fulfilled
g) Code merged in the main branch and running in Prod-preview
h) User docs completed, reviewed and published
i) Code deployed in Production
j) Other _____ (text)

6.4.4 Are there any elements of 'Definition of Done' you disagree with?
_____ (Long answer text)

6.4.5 If you disagree with 'Definition of Done', what elements of 'Definition of Done' do you disagree with and why?
_____ (Long answer text)

6.5 Does your team use 'Definition of Ready'?
a) Yes
b) No
c) I don't know

6.5.1 If 'Definition of Ready' is used, do you adhere to it strictly?

a) Never
b) Sometimes
c) Most of the time
d) Always

6.5.2 Are there any elements of 'Definition of Ready' you disagree with?

a) Yes
b) No
c) Other _____ (text)

6.5.2.1 If you disagree, what elements of 'Definition of Ready' do you disagree with and why?

_____ (Long answer text)

A.7 Software Testing

7.1 Is your team using the following (if any)?

a) Unit Test
b) Integration Test
c) Functional Test
d) Automated Test
e) Performance/Load Test
f) Acceptance Test
g) Other _____ (text)

A.8 Software Quality

8.1 How quickly are committed changes in your project verified?

a) Immediately
b) 1-2 hours
c) 2+ hours
d) <1 day
e) 1-3 days
f) >3 days

8.2 How quickly are PRs reviewed?

a) Immediately
b) 1-2 hours
c) 2+ hours
d) <1 day
e) 1-3 days
f) >3 days

8.3 Which of the following are the most common causes for delaying completion of your task (if any)?

a) Lack of proper communication between team members
b) Unplanned absence of team members

c) Different understanding of solution required
d) Lack of proper task description
e) Task blocked by another task in progress
f) Unnecessary meetings
g) Other _____ (text)

8.4 What quality tools/ techniques are used by your team (if any)?
a) SonarQube
b) Code review
c) Coding standards
d) Test-driven development
e) Behaviour-driven development
f) Unit testing
g) Automated acceptance testing
h) Pair programming
i) CI/CD
j) Collective Code Ownership
k) Other _____ (text)

8.5 How often are builds provided within a Sprint?
a) Everyday
b) Based on feature completion
c) Sprint end
d) Other _____ (text)

8.6 Does Scrum help your team with any of the following?
a) Number of bugs verified within Sprint is greater
b) Communication between team member(s) is improved
c) Team collaboration is greater
d) Successful adaptation of Sprint test approach is possible
e) Number of bugs from final Regression Test is smaller
f) Test coverage is greater
g) Number of bugs from User Acceptance Test is smaller
h) None
i) Other _____ (text)

A.9 Project Management

9.1 Do you run a Project Manager over the project?
a) Yes
b) No
c) Other _____ (text)

9.1.1 If you run Project Manager over the project, does that person take any of the traditional Scrum roles?
a) Yes
b) No
c) Other _____ (text)

9.2 Do you run a Manager/Senior Manager over the project?
a) Yes
b) No

9.2.1 If you run Manager/Senior Manager over the project, does that person take any of the traditional Scrum roles?
a) Yes
b) No

9.3 Do you have Architect involved in the project?
a) Yes
b) No

9.4 Do you have Sales team involved in the project?
a) Yes
b) No

9.5 do you have DevOps team involved in the project?
a) Yes
b) No

A.10 Communication

10.1 How would you mark your team's overall collaboration?
a) High
b) Medium
c) Low

10.2 Which of the following communication channels are you currently using in project (if any)?
a) Face to face
b) Jira
c) Whiteboard
d) Email
e) BlueJeans
f) Skype
g) Slack
h) IRC
i) Google Drive docs
j) Confluence
k) Trello
l) Collaborative Design/Mockups (e.g Azure, Lucidcharts, Draw.io etc)
m) GitHub/GitLab (PR discussions)
n) Other _____ (text)

10.3 Do you have a singular source of truth from the above channels?
a) Yes
b) No

10.4 If you trust any of the channel(s) (not necessarily listed above) please specify which channel(s) do you default to?
_____ (Long answer text)

10.5 How much time do you spend weekly in meetings (if any)?
a) 0-1 hours
b) 1-2 hours
c) 2-4 hours
d) 4+ hours

10.6 How much time do you spend weekly on face-to-face meetings (if any)?
a) 0-1 hours
b) 1-2 hours
c) 2-4 hours
d) 4+ hours

10.7 How much time do you spend weekly on web conferences (if any)?
a) 0-1 hours
b) 1-2 hours
c) 2-4 hours
d) 4+ hours

10.8 How much time do you spend weekly on email communication (if any)?
a) 0-1 hours
b) 1-2 hours
c) 2-4 hours
d) 4+ hours

10.9 How much time do you spend weekly with the customer (if any)?
a) 0-1 hours
b) 1-2 hours
c) 2-4 hours
d) 4+ hours

A.11 Business

11.1 What size is your customer?
a) Small
b) Medium
c) Large

11.2 Does your customer understand and follow Scrum principles?
a) Yes
b) No

11.3 Who is Product Owner in your team?
a) Customer
b) Other _____ (text)

11.4 Is your customer respectable of 'no deadlines?

a) Never
b) Sometimes
c) Most of the time
d) Always

11.5 Does your customer help create user stories?

a) Never
b) Sometimes
c) Most of the time
d) Always

11.6 Does your customer participate in any of the following?

a) Sprint planning
b) Sprint review & demonstration
c) Sprint retrospective
d) Backlog review & refinement
e) Instant feedback
f) Final project handover
g) Other _____ (text)

11.7 What is your customer's satisfaction level with adoption of Agile?

a) High
b) Low
c) Medium
d) I don't know

11.8 Do you think that Agile contributes to success of the project?

a) Yes
b) No
c) I don't know

Appendix B

Questionnaire Results

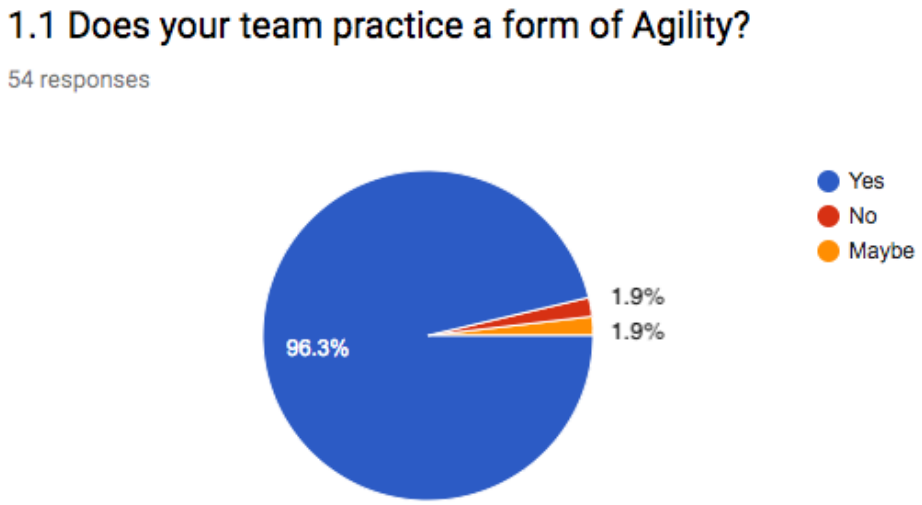

FIGURE B.1: Percentage of participants practicing Agility

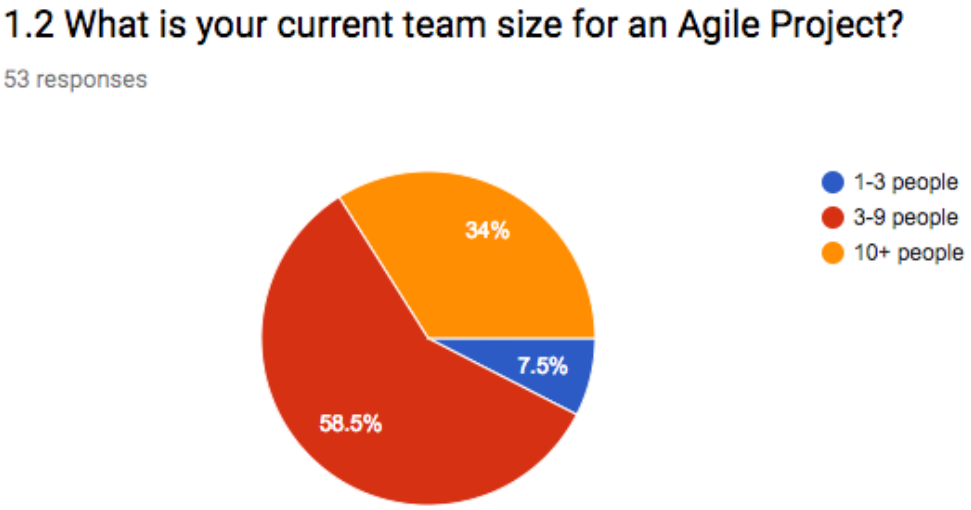

FIGURE B.2: Team size for Agile project

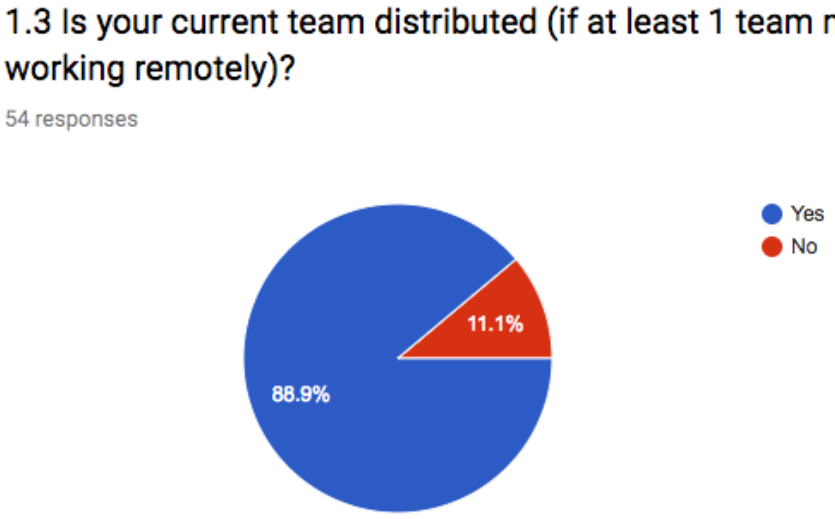

FIGURE B.3: Teams distribution

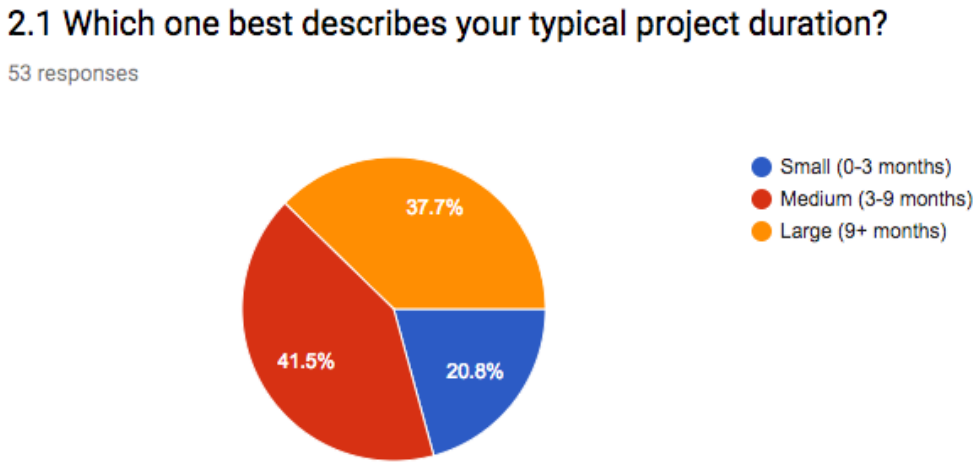

FIGURE B.4: Typical project duration/size

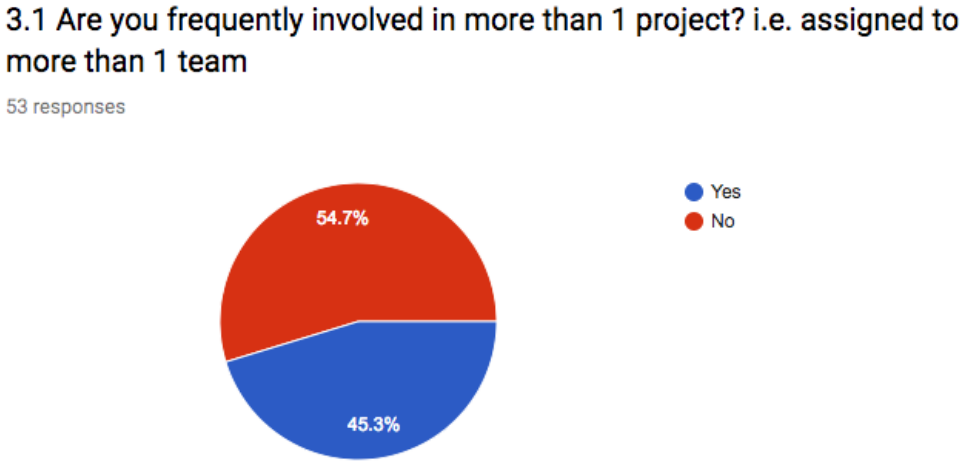

FIGURE B.5: Involvement in more than 1 project/team

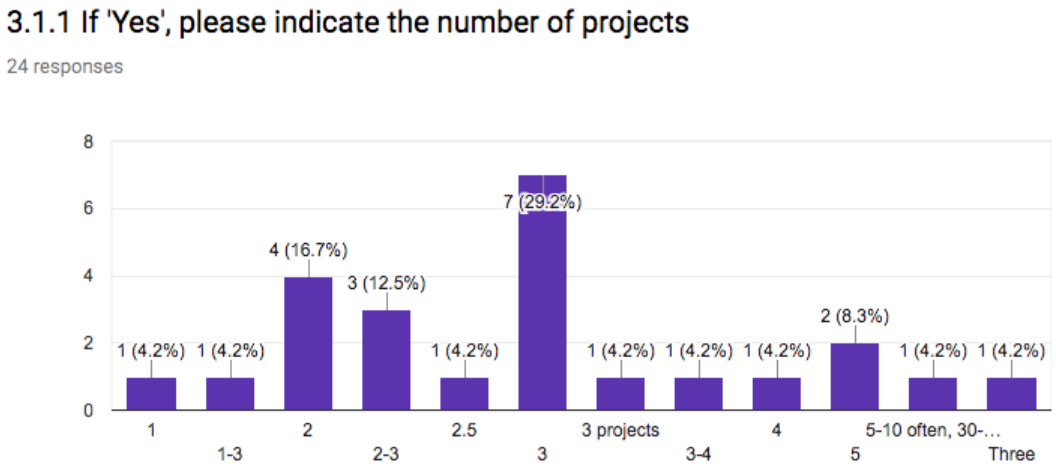

FIGURE B.6: Percentage of projects involved in

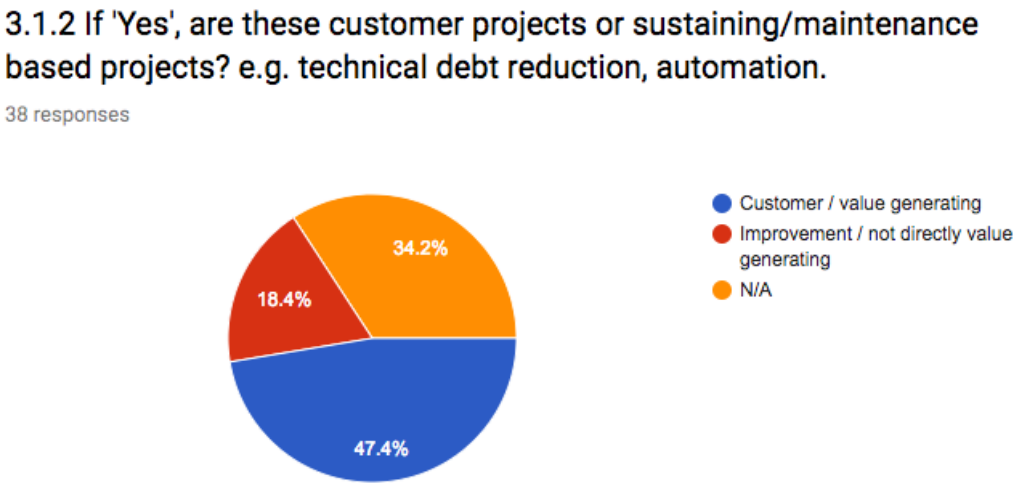

FIGURE B.7: Percentage of customer and sustaining/maintenance projects

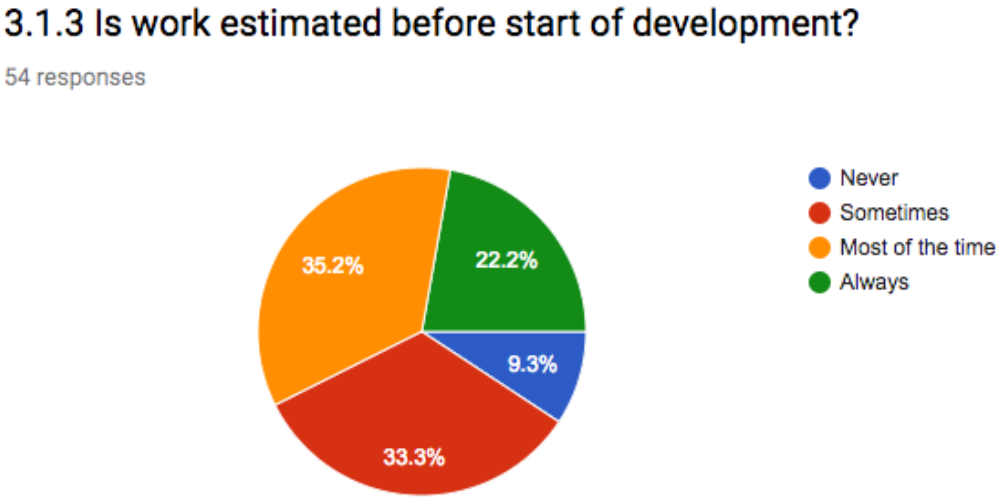

FIGURE B.8: Percentage of work estimated before development

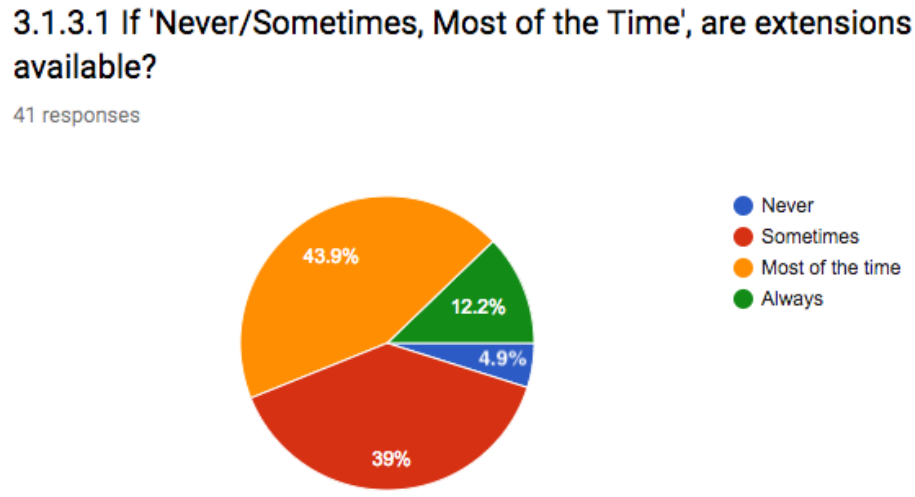

FIGURE B.9: Percentage of project extensions available

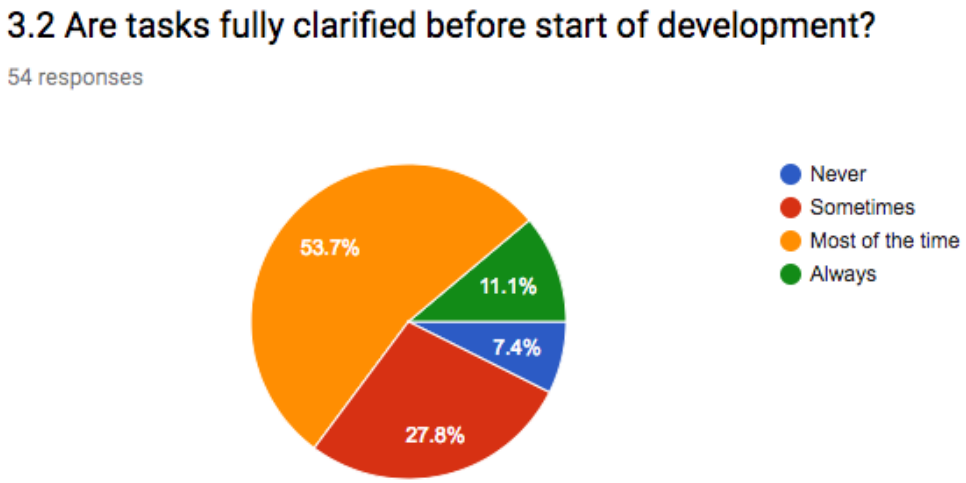

FIGURE B.10: Percentage of tasks fully clarified before development

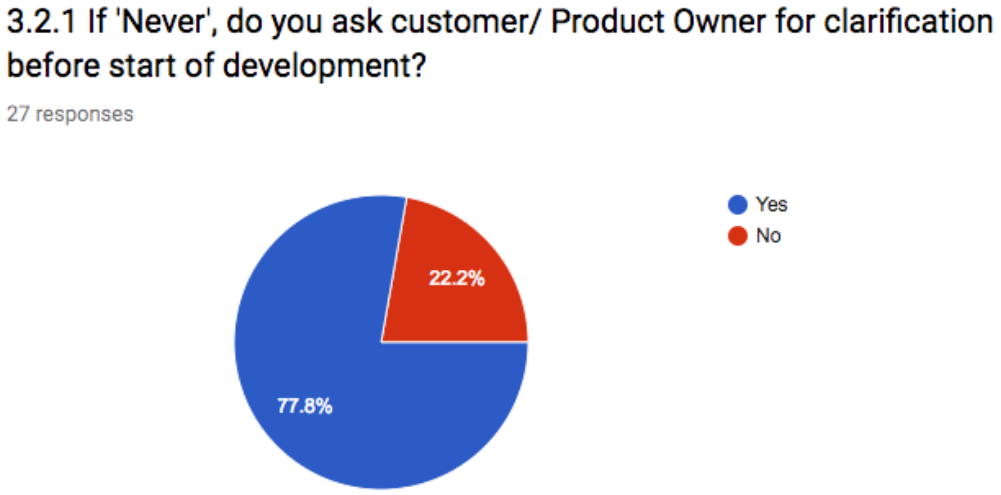

FIGURE B.11: Percentage of participants asking for tasks clarification

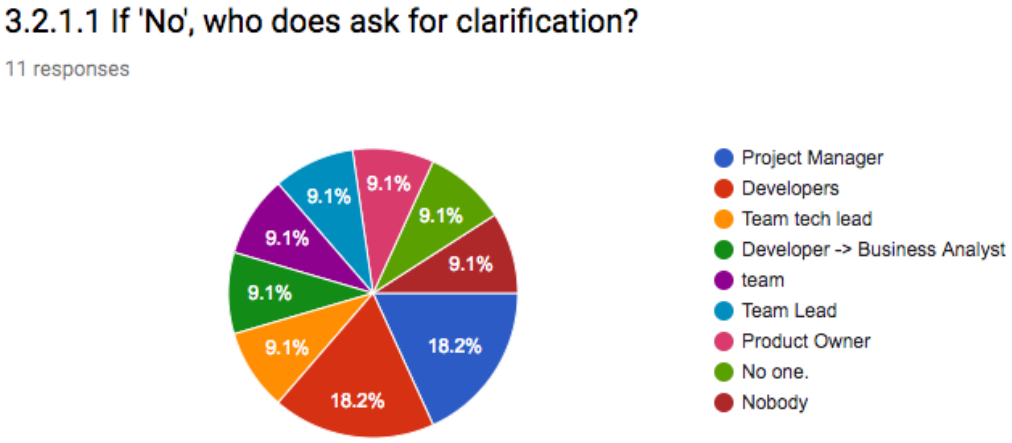

FIGURE B.12: Percentage of those who ask for tasks clarification

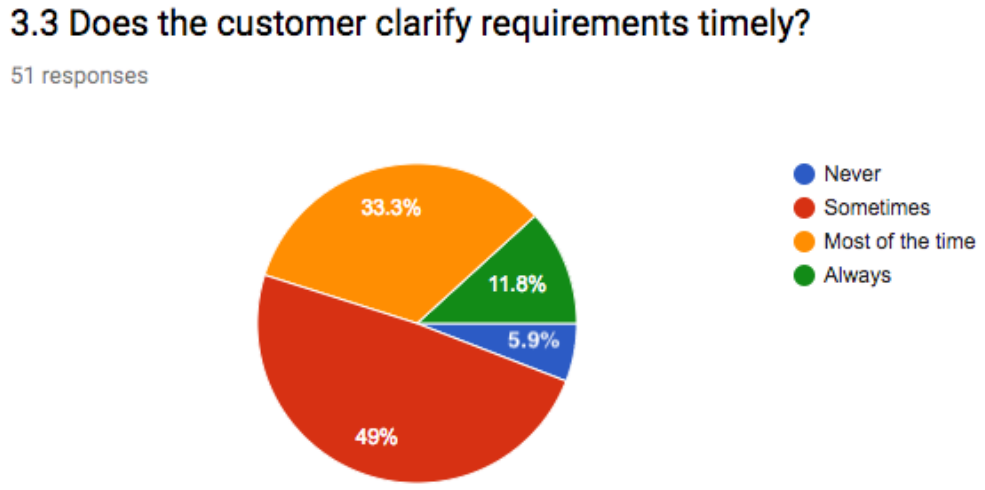

FIGURE B.13: Percentage of times that customer clarifies requirements timely

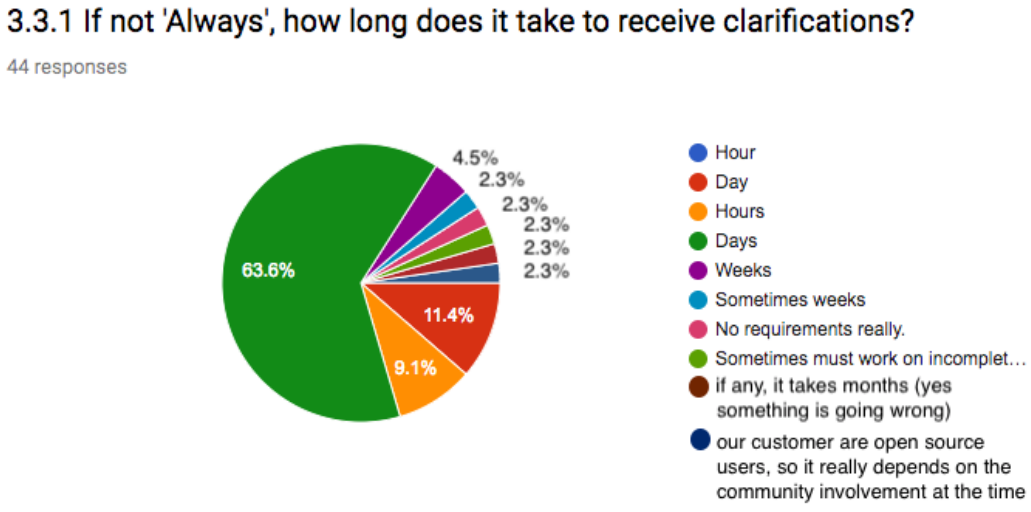

FIGURE B.14: Speed of customer's task clarifications

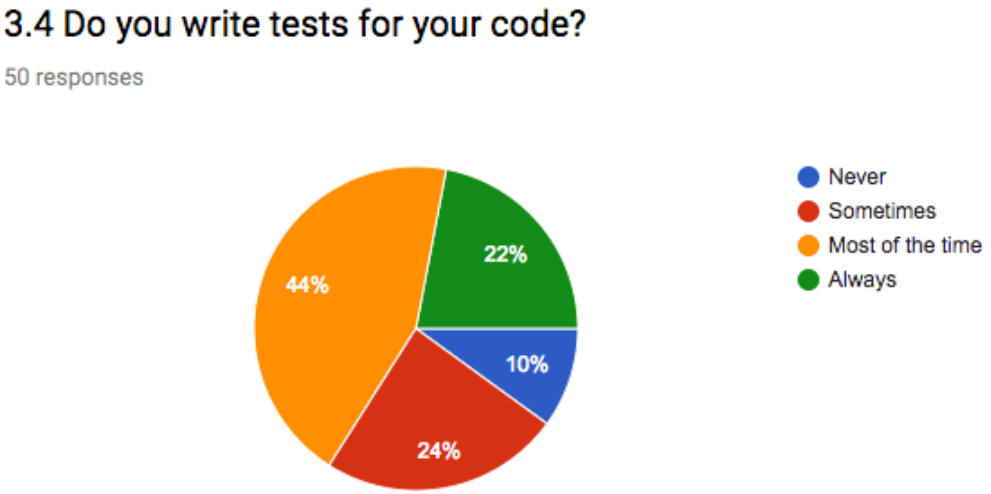

FIGURE B.15: Percentage of participant's writing tests

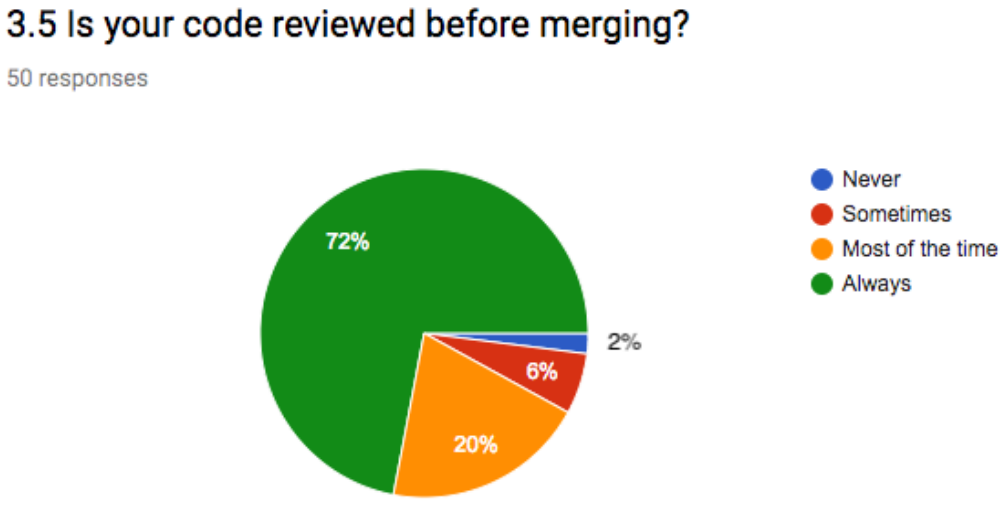

FIGURE B.16: Percentage of those who have code reviews

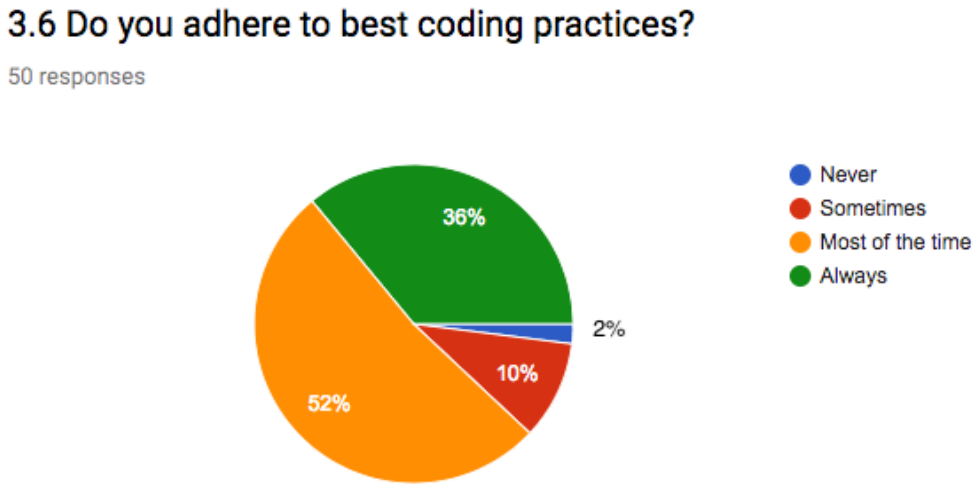

FIGURE B.17: Percentage of those who adhere to best coding practices

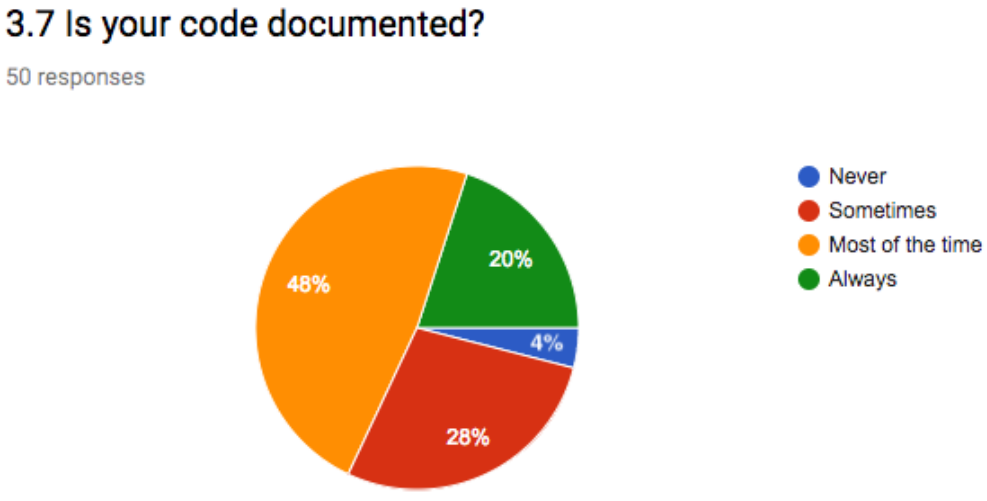

FIGURE B.18: Percentage of those who document their code

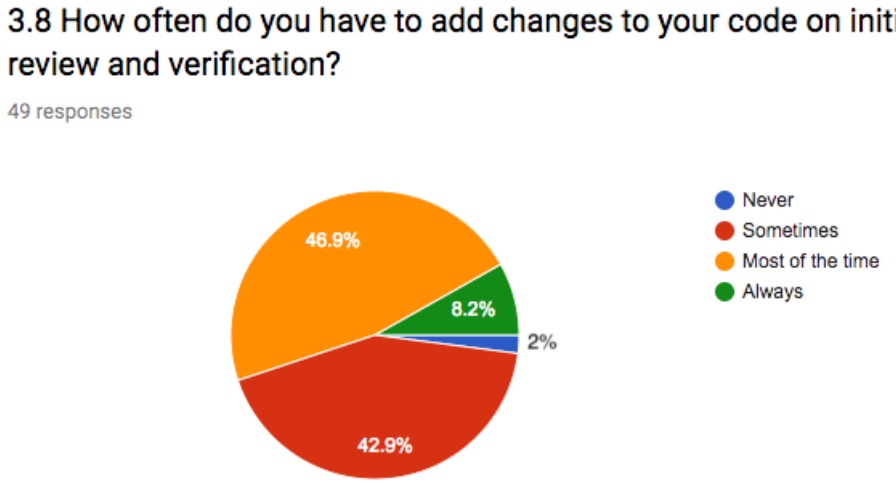

FIGURE B.19: Percentage of those who add changes to code on initial code review and verification

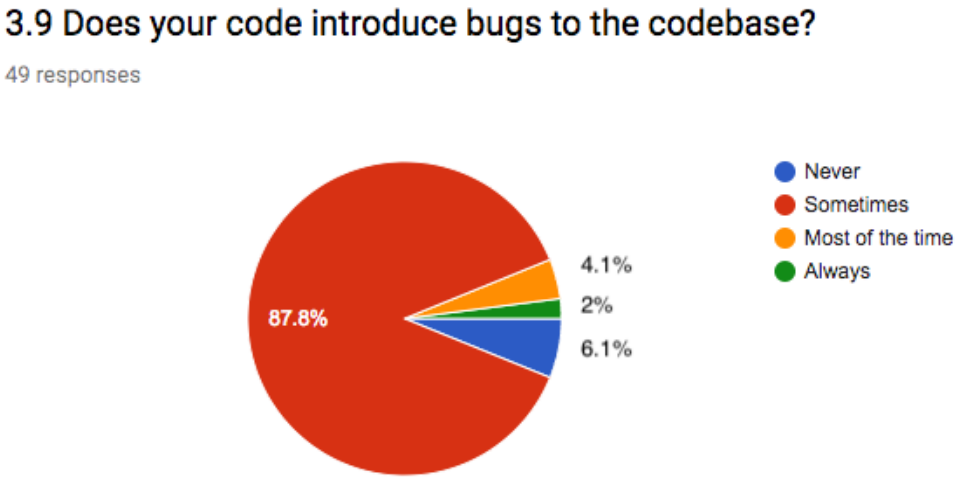

FIGURE B.20: Percentage of bugs introduction to codebase

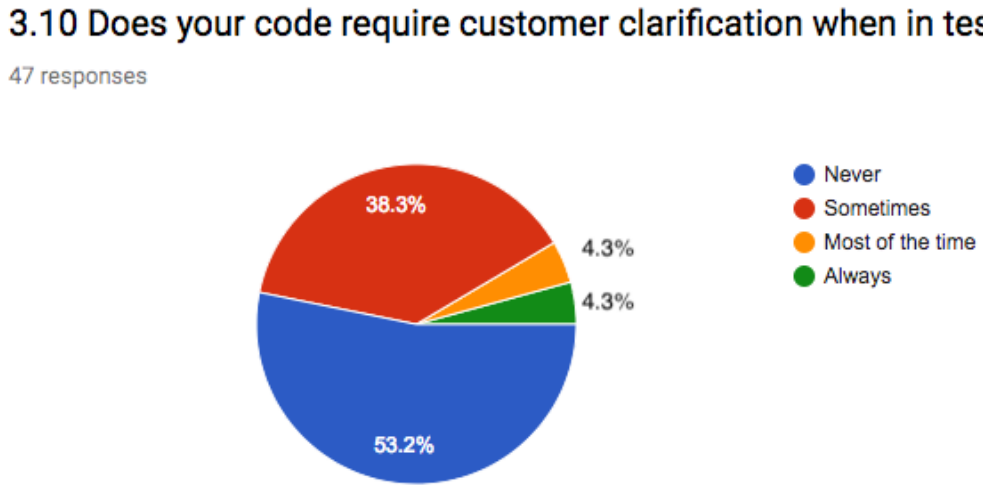

FIGURE B.21: Customer verification when task is in test

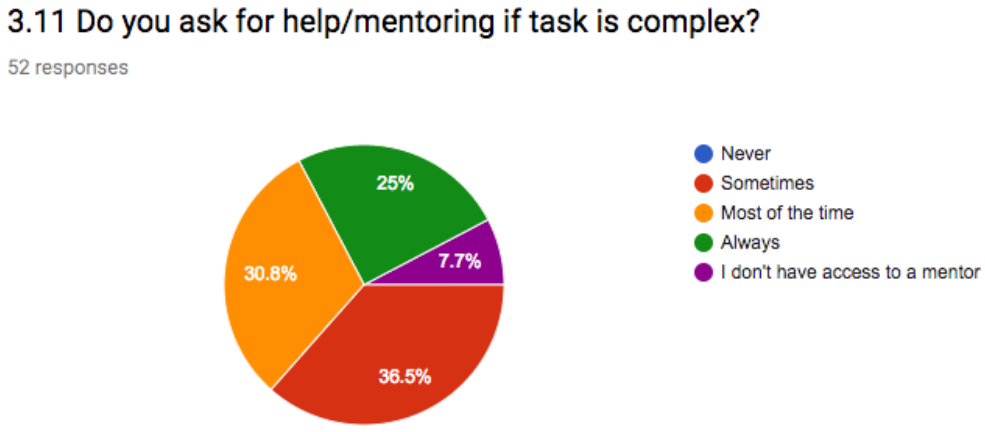

FIGURE B.22: Percentage of those who ask for help/mentoring when task is complex

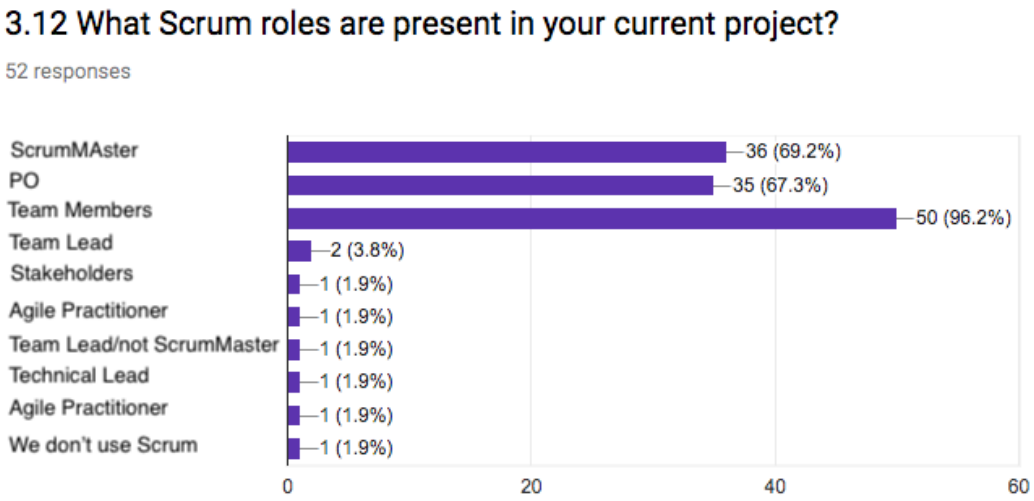

FIGURE B.23: Scrum roles present in participant's project

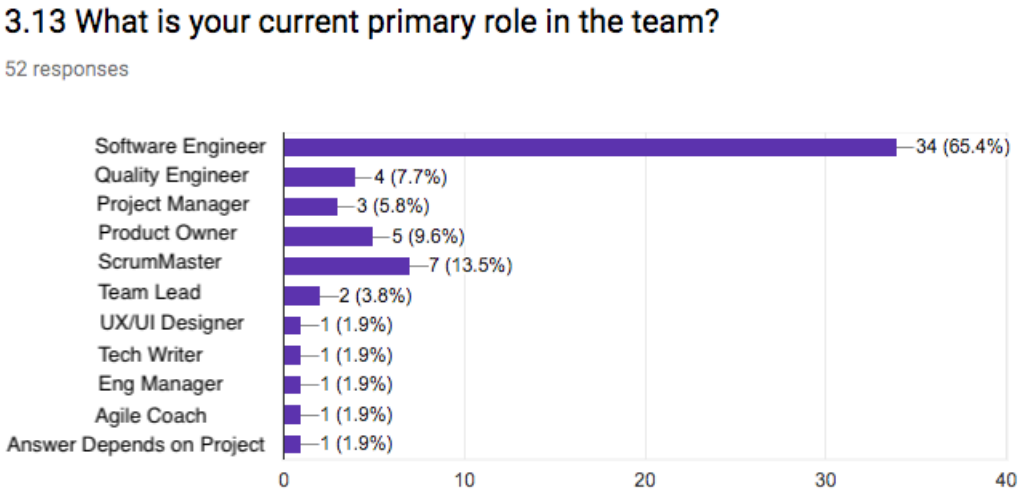

FIGURE B.24: Participant's primary role in the team

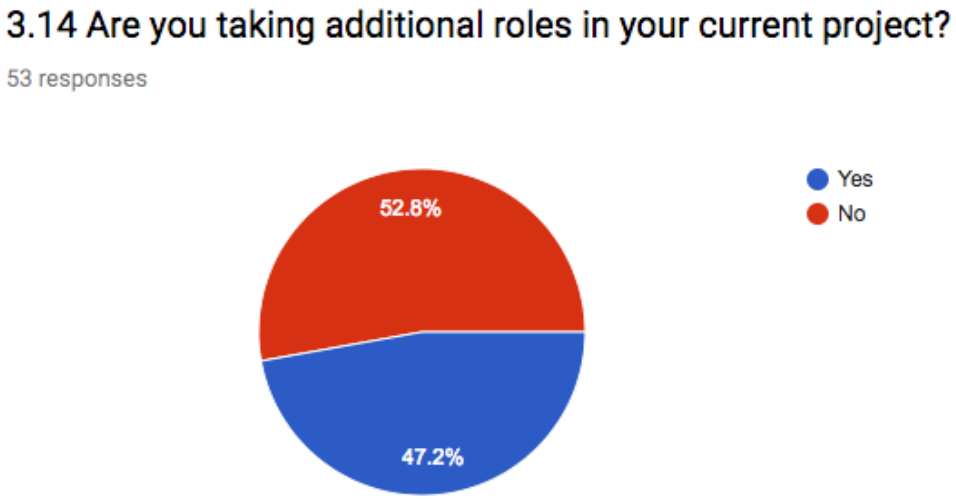

FIGURE B.25: Percentage of participant's taking additional role(s) in project

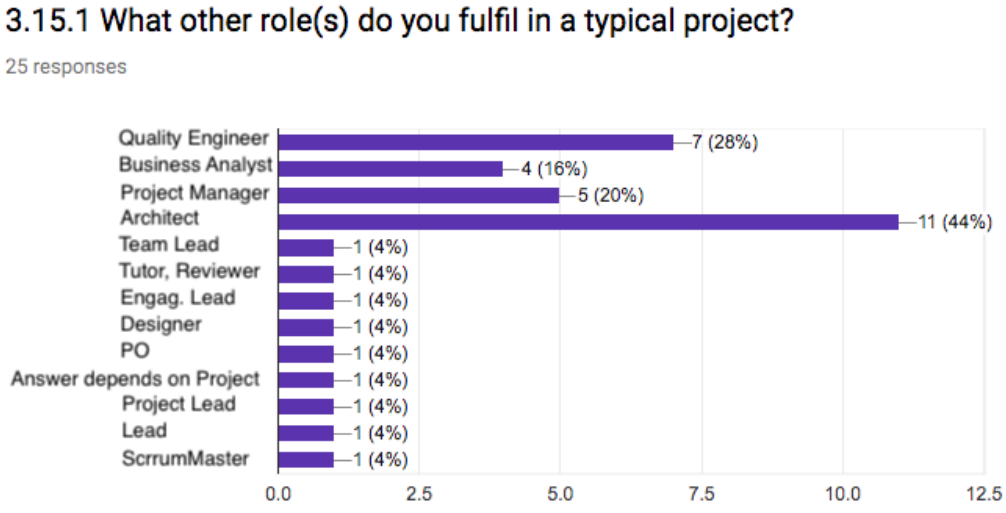

FIGURE B.26: Participant's additional role(s) in a typical project

FIGURE B.27: Participant's skills to cover all additional roles in project

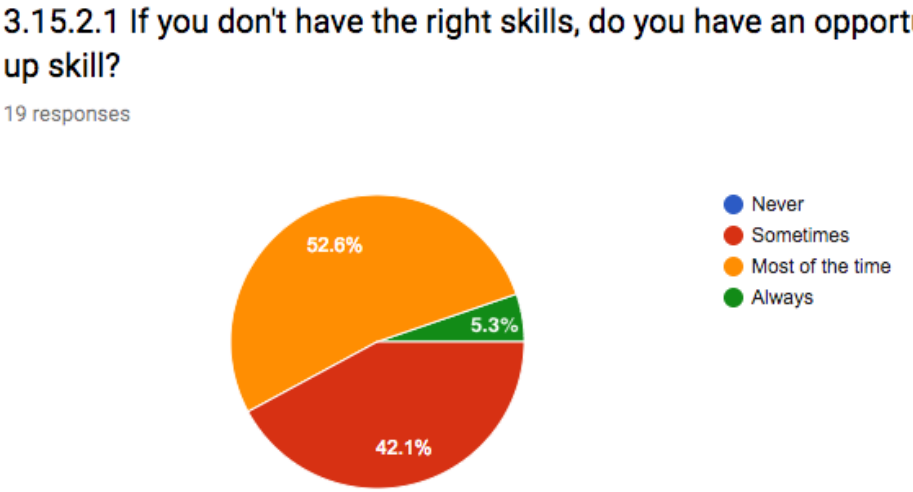

FIGURE B.28: Possibility to up skill to perform additional roles

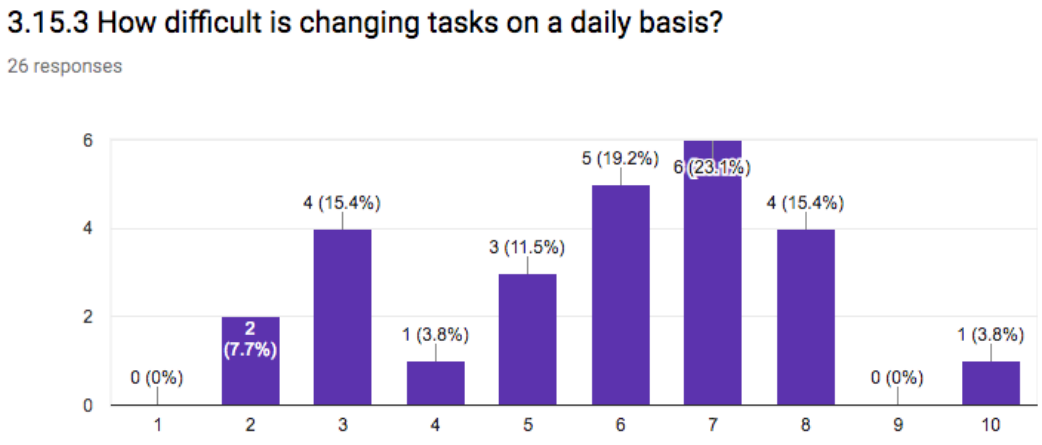

FIGURE B.29: Difficulty levels for changing tasks on daily basis

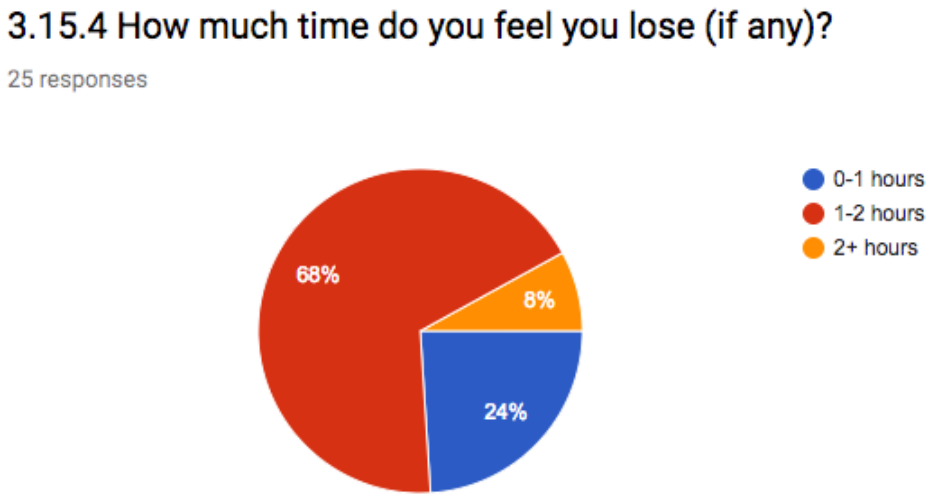

FIGURE B.30: Time loses when changing tasks daily

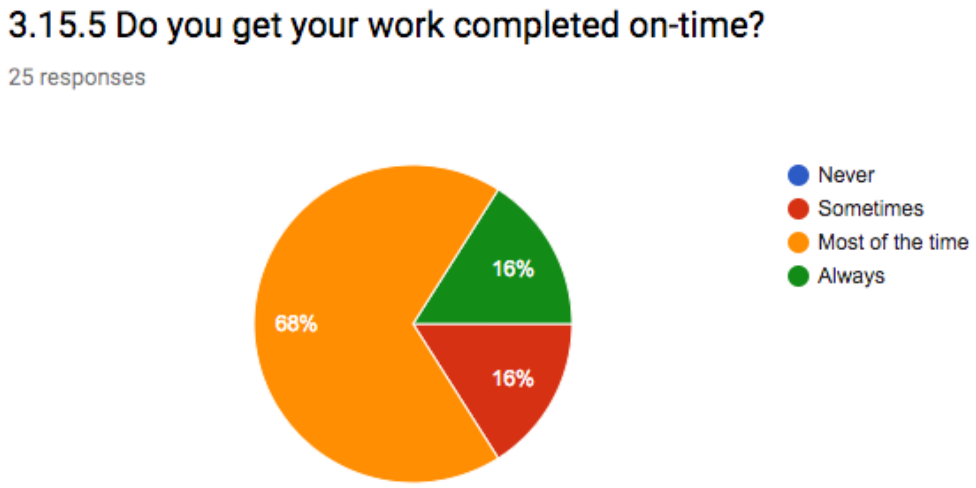

FIGURE B.31: Percentage of participant's work completion on-time when fulfilling multiple roles in project

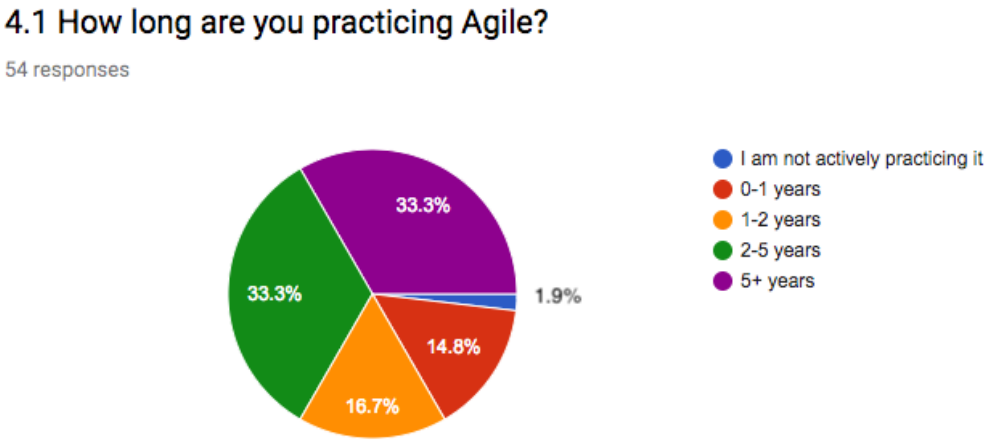

FIGURE B.32: Length of participant's Agile practice

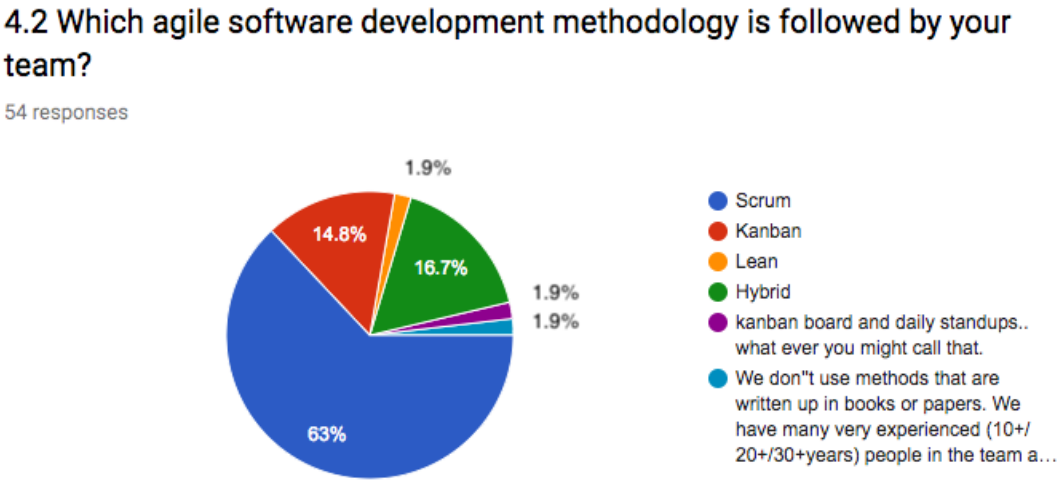

FIGURE B.33: Agile methodologies followed by participants

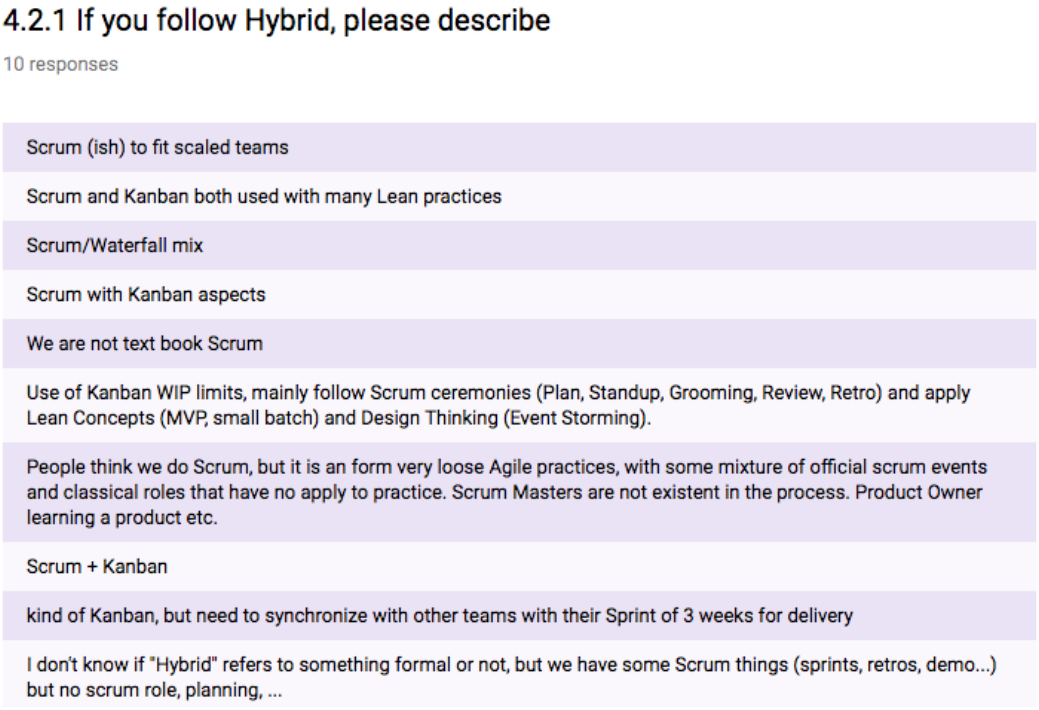

FIGURE B.34: Hybrid methodology followed

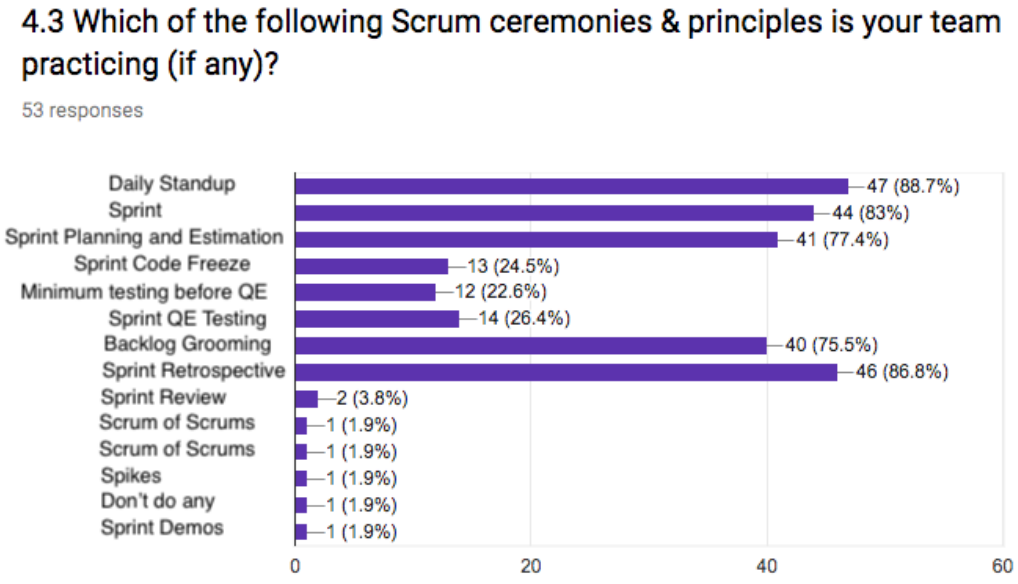

FIGURE B.35: Scrum practices used by teams

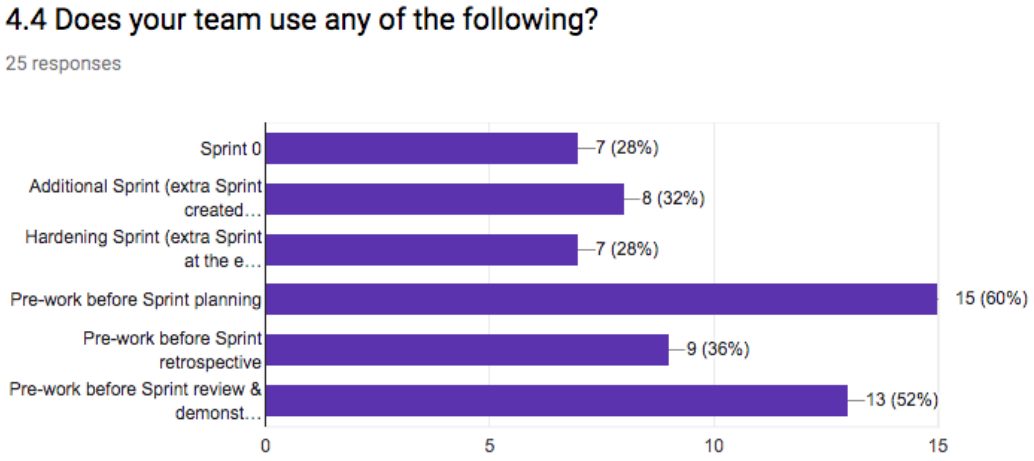

FIGURE B.36: Additional Scrum practices used by teams

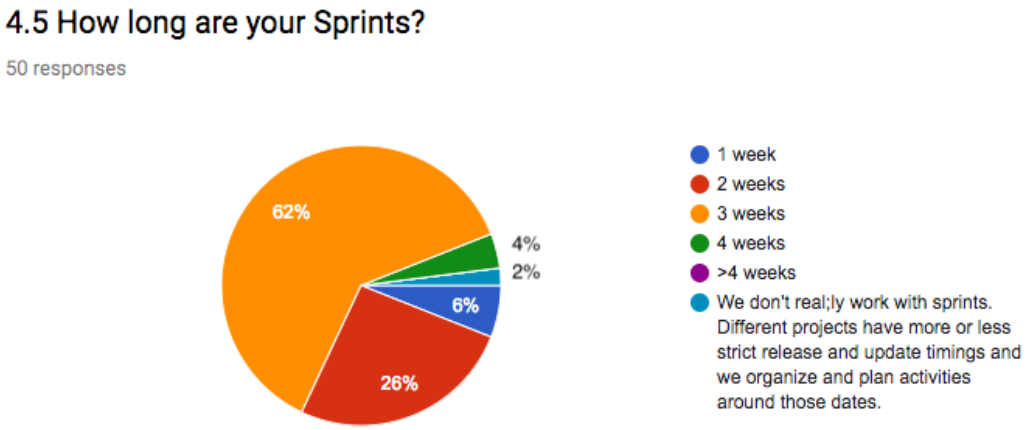

FIGURE B.37: Sprint duration across different size teams

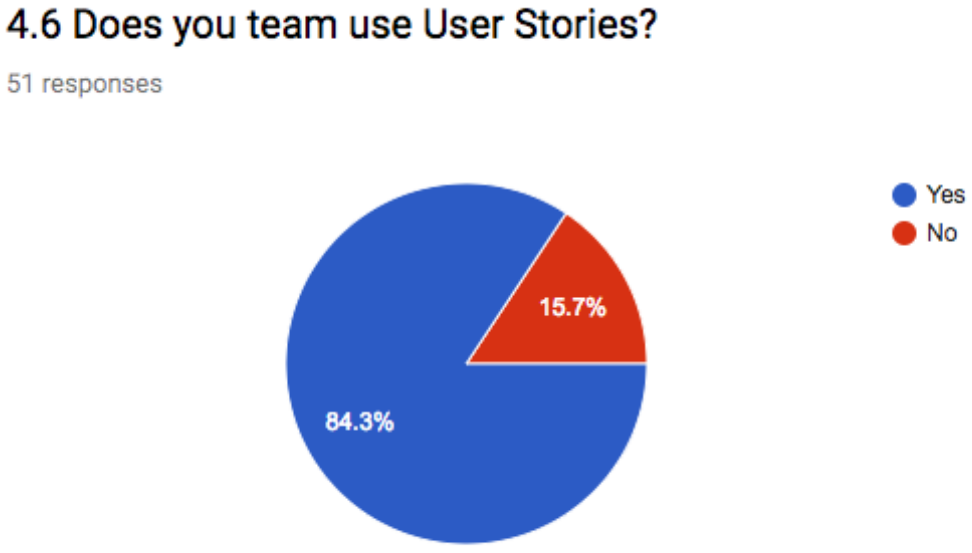

FIGURE B.38: Percentage of User Stories used by participants

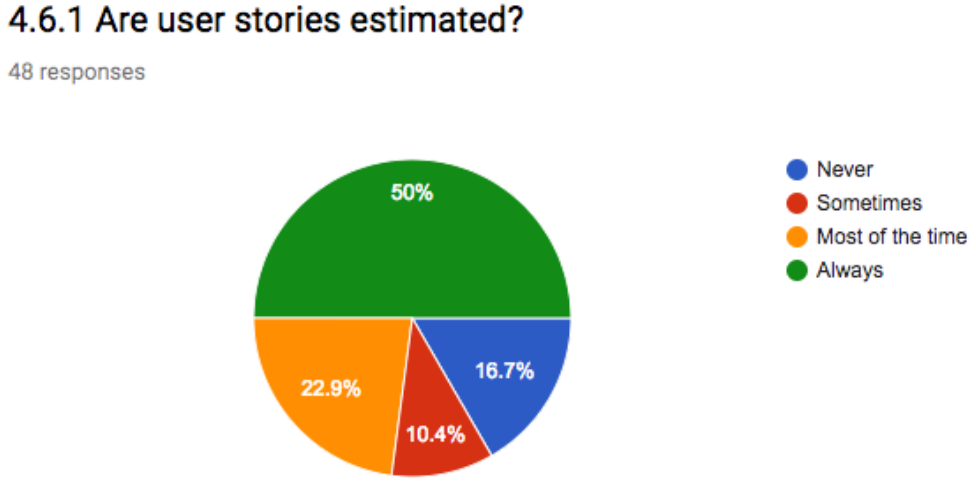

FIGURE B.39: Percentage of estimating User Stories

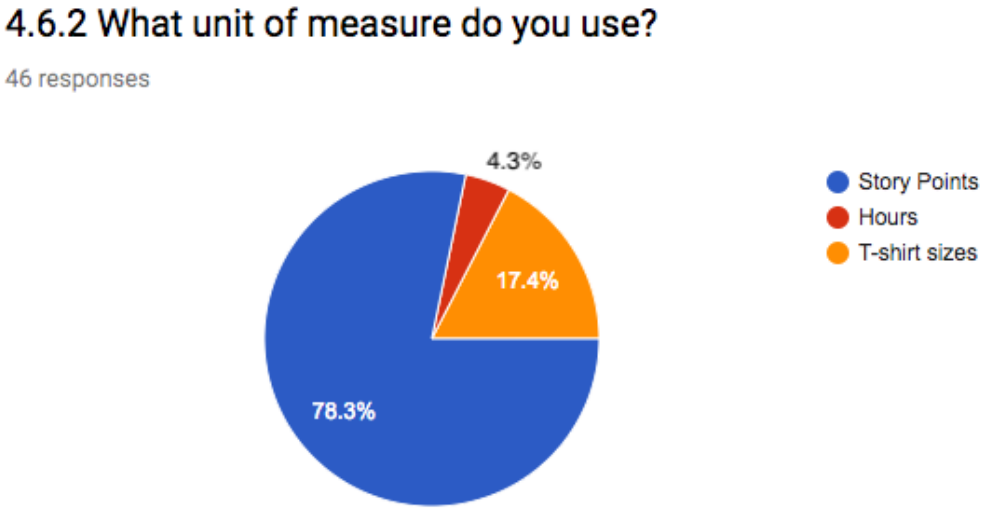

FIGURE B.40: Units of measure for features used by participants

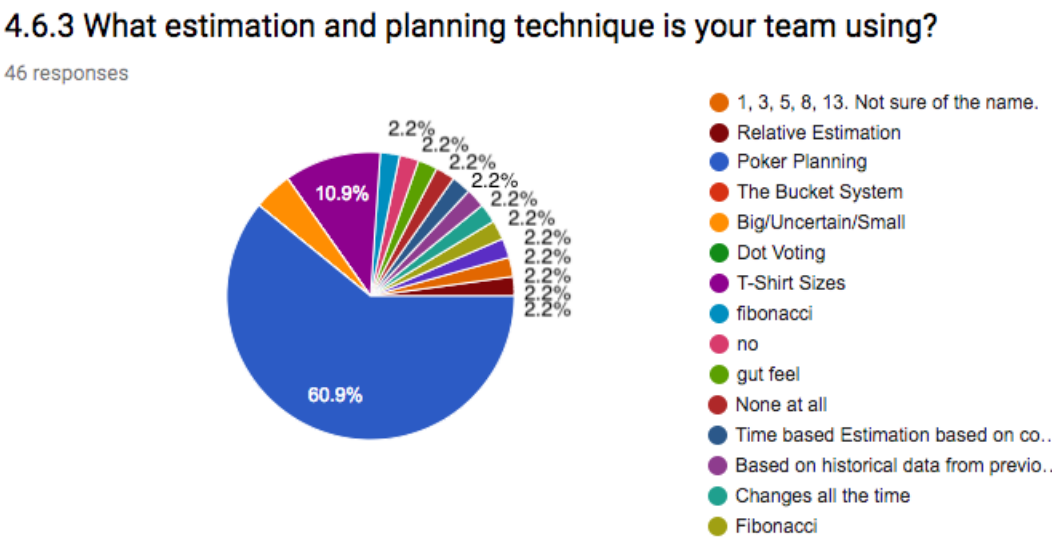

FIGURE B.41: Work estimation methods used

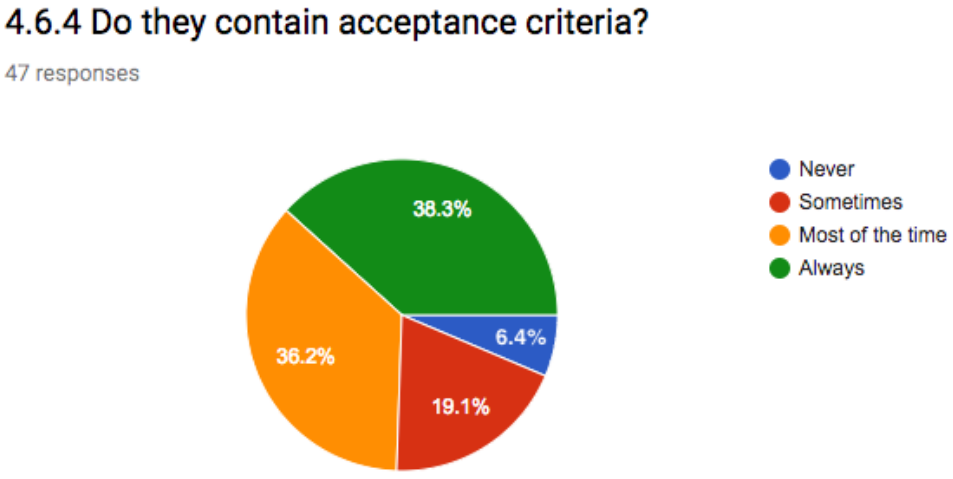

FIGURE B.42: Percentage of work items containing acceptance criteria

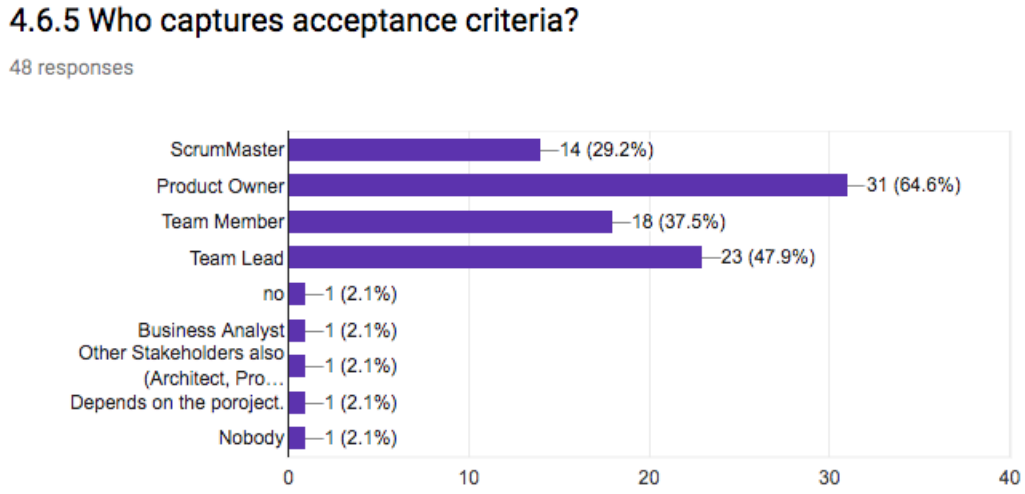

FIGURE B.43: Capturing acceptance criteria

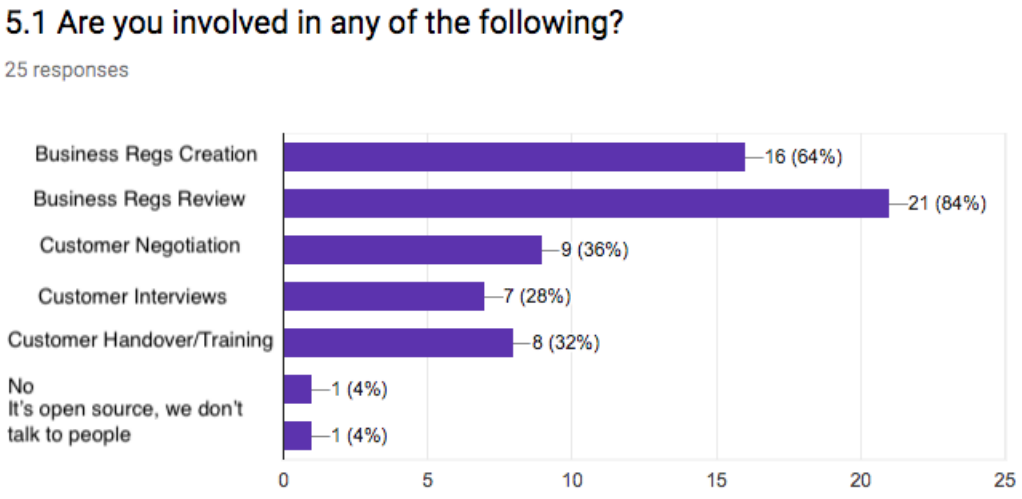

FIGURE B.44: Involvement in software requirements creation, clarification etc.

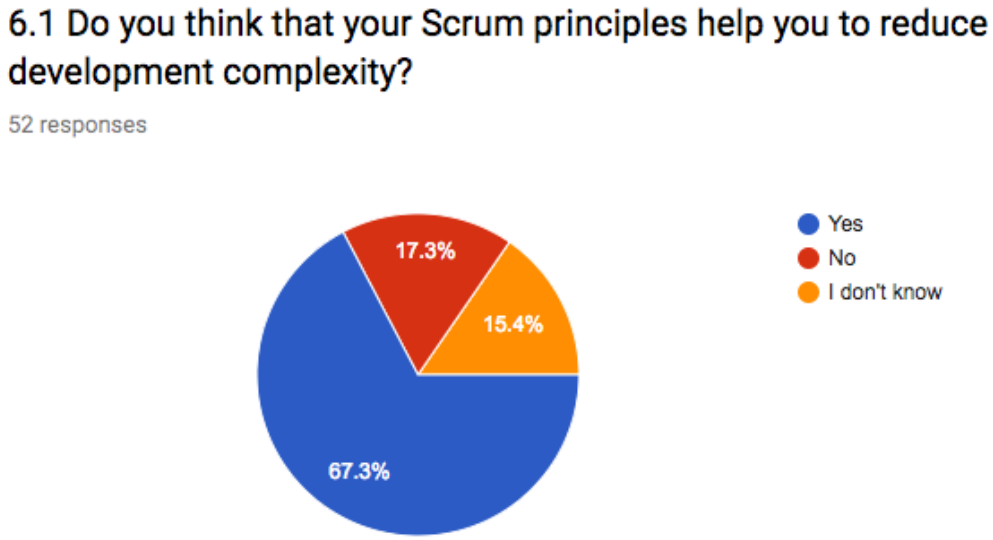

FIGURE B.45: Scrum principles helping to reduce development complexity

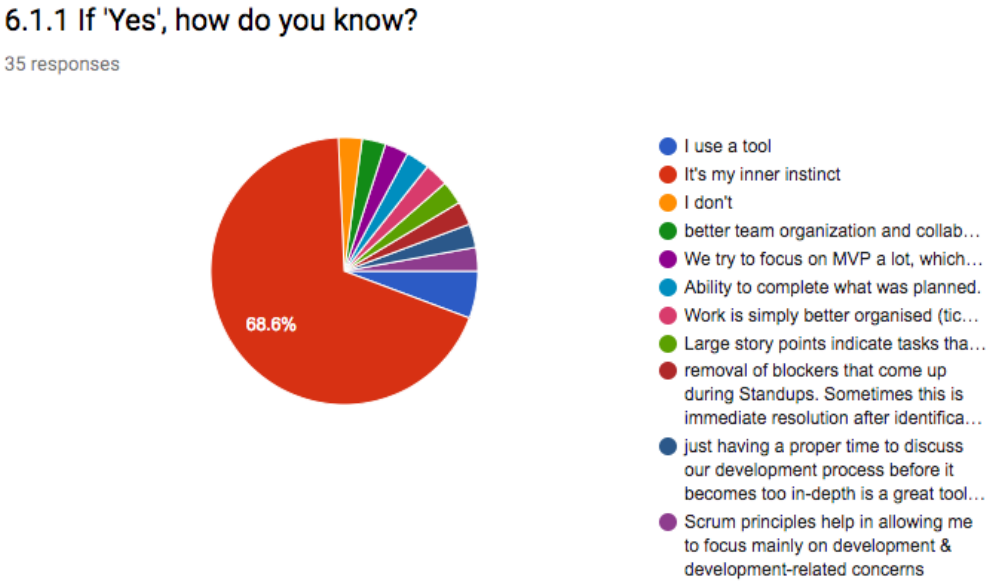

FIGURE B.46: Results from follow up question

6.1.1.1 If you use a tool, please specify what tool do you use?

7 responses

jira
JIRA analysis and some statistical analysis
Trello (Not great or even good for Agile tracking and planning...!)
JIRA, Confluence, BitBucket
JIRA
Trello
For development complexity? None. Use JIRA for bug/ task tracking though.

FIGURE B.47: Tools used for verifying Scrum reduction of development complexity

6.2 How frequently does change request happen in your project (if any)?

49 responses

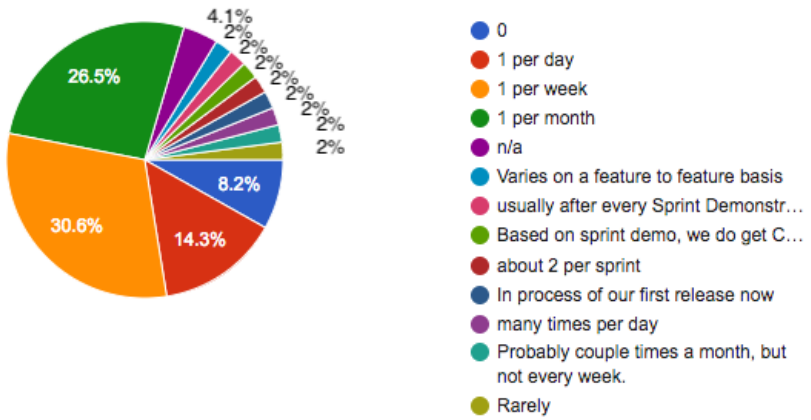

FIGURE B.48: Frequency of change requests in project

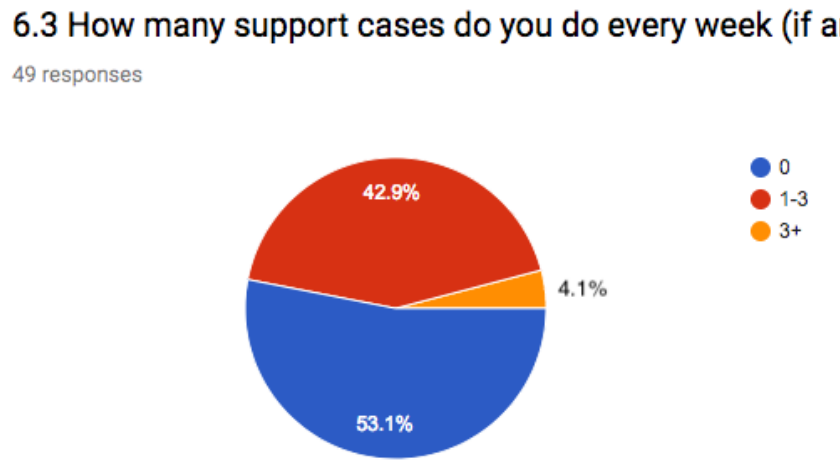

FIGURE B.49: Number of support cases to worked on

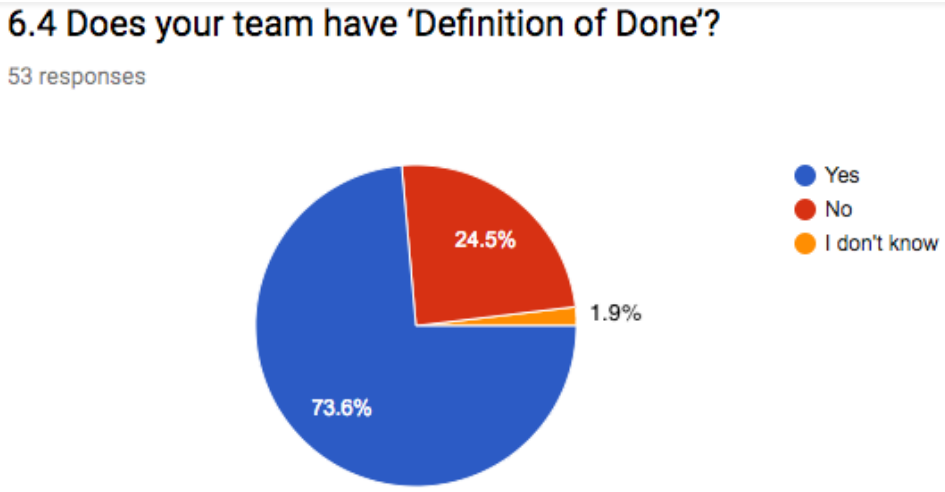

FIGURE B.50: Usage of 'Definition of Done'

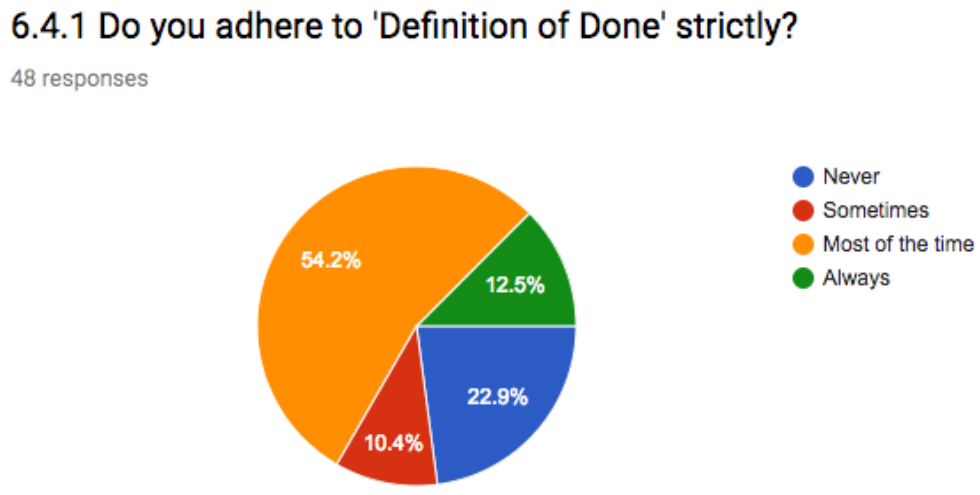

FIGURE B.51: Level of adherence to 'Definition of Done'

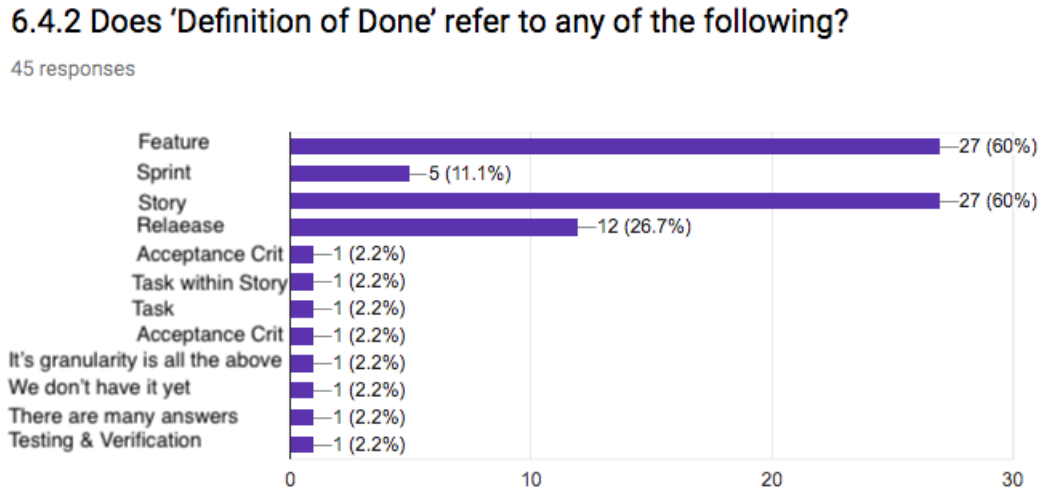

FIGURE B.52: Reference of 'Definition of Done' to work items

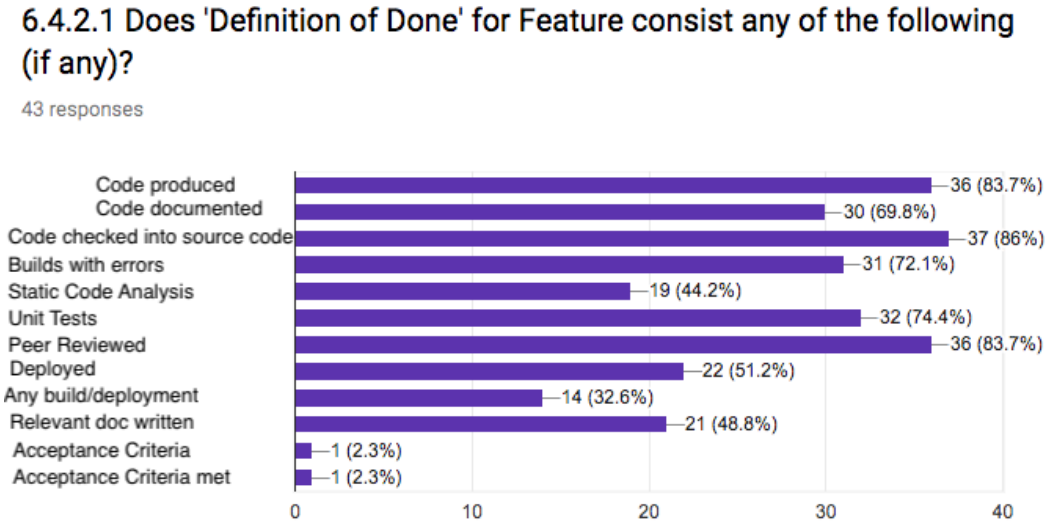

FIGURE B.53: Use of 'Definition of Ready'

6.4.2.2 Are there any elements of 'Definition of Done' for Feature you disagree with?

42 responses

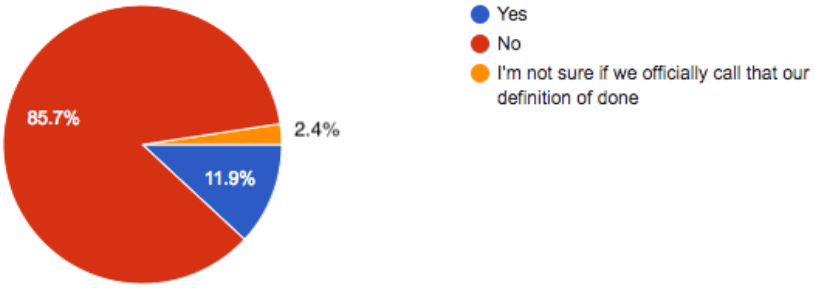

FIGURE B.54: Disagreement of elements of 'Definition of Done' for Feature

6.4.2.3 If you disagree with 'Definition of Done' for Feature, what elements do you disagree with and why?

5 responses

We should include static code analysis tools and automated feature tests.
should include documentation and system test managed by QE
Way too rigid way of thinking. Every case can be different. Not agile at all.
We don't actually use the term "Definition of Done" but we have some criteria for moving cards to Complete
in some places we have code coverage as a metric of done, i'm not convinced this is a good metric to have in place at this level of resolution.

FIGURE B.55: Disagreement of elements of 'Definition of Done' for Feature

6.4.3 Does 'Definition of Done' consist any of the following?

38 responses

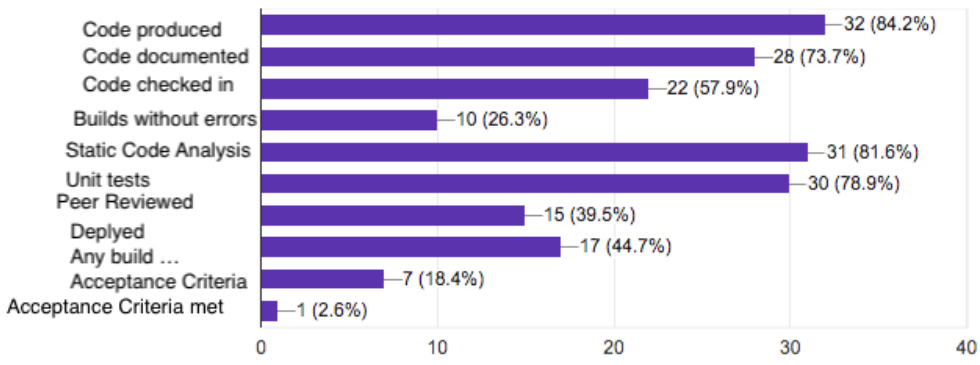

FIGURE B.56: Elements of 'Definition of Done'

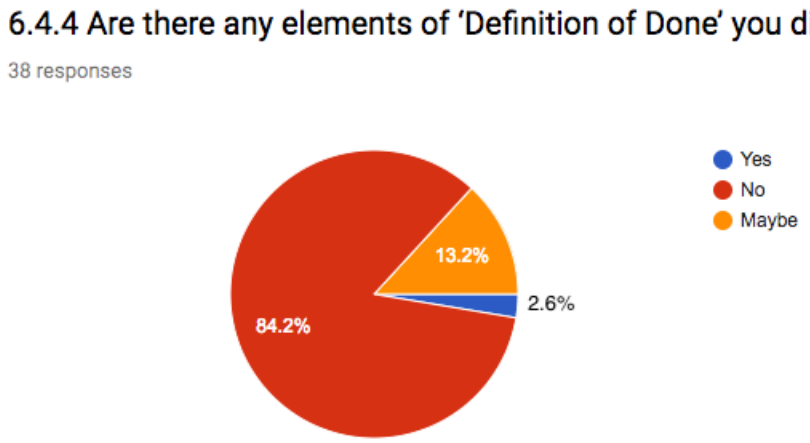

FIGURE B.57: Disagreement of elements of 'Definition of Done'

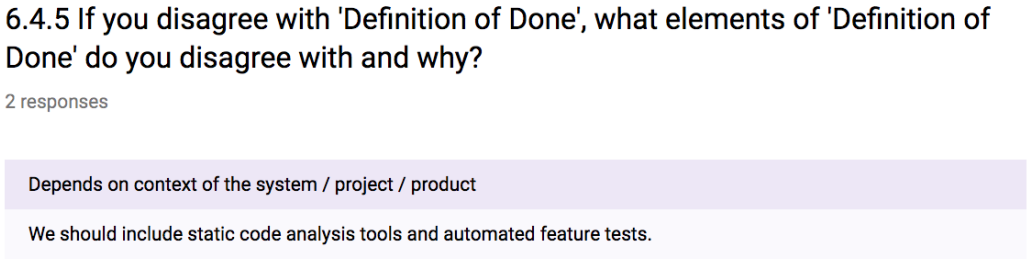

FIGURE B.58: Disagreement of elements of 'Definition of Done'

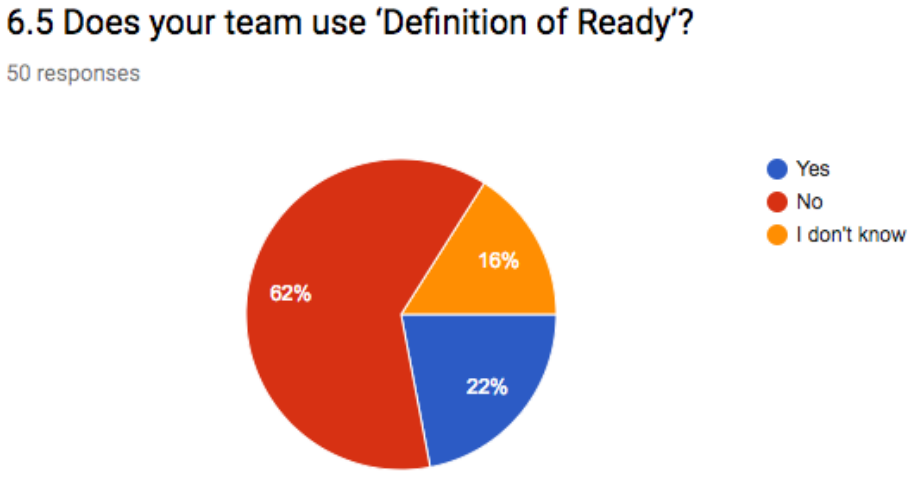

FIGURE B.59: Use of 'Definition of Ready'

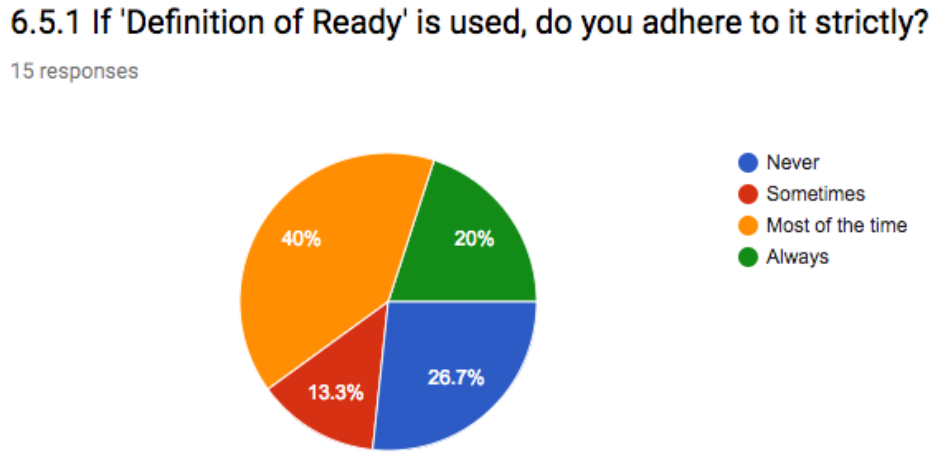

FIGURE B.60: Level of adherence to 'Definition of Ready'

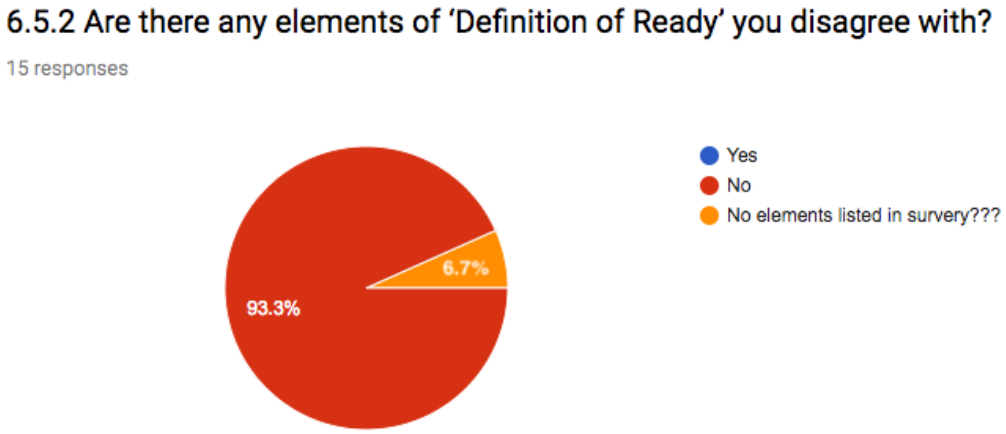

FIGURE B.61: Disagreement of elements of 'Definition of Ready'

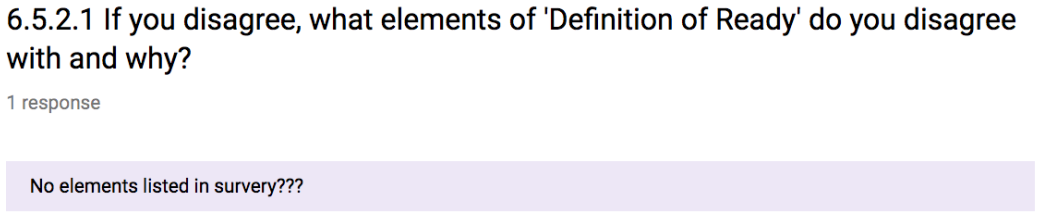

FIGURE B.62: Disagreement of elements of 'Definition of Ready'

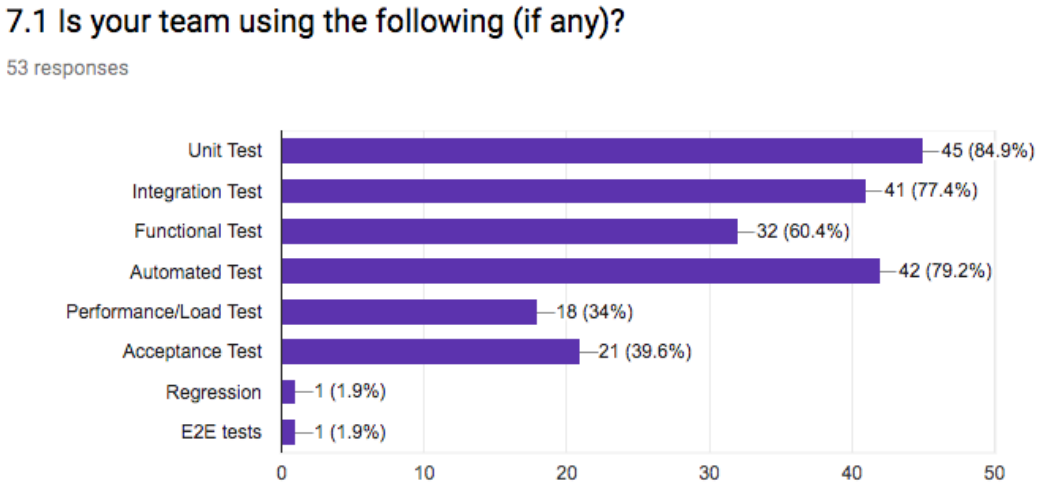

FIGURE B.63: Use of testing

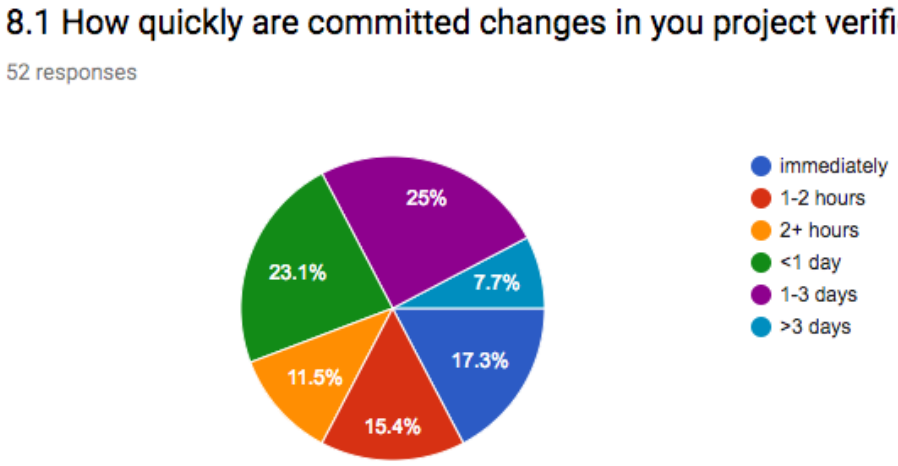

FIGURE B.64: Speed of work verification

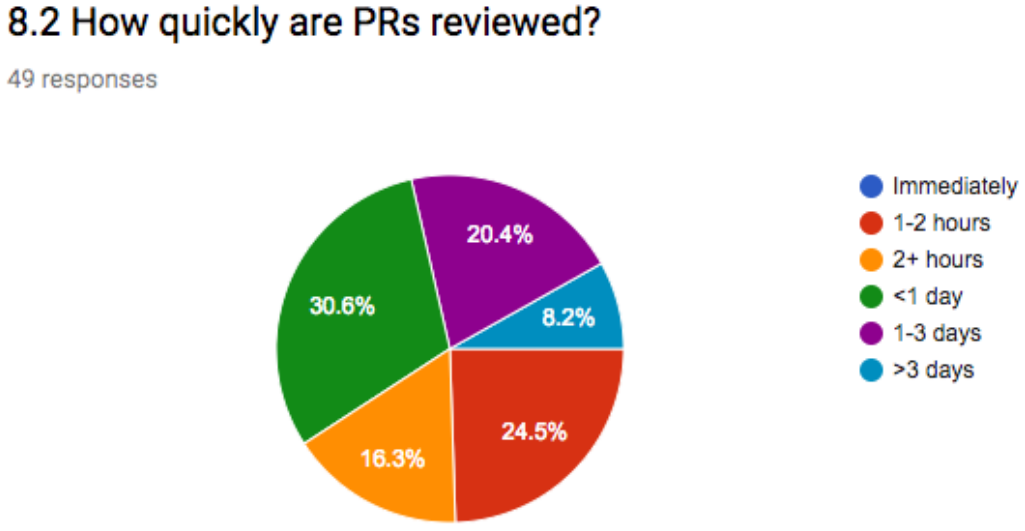

FIGURE B.65: Speed of PR reviews

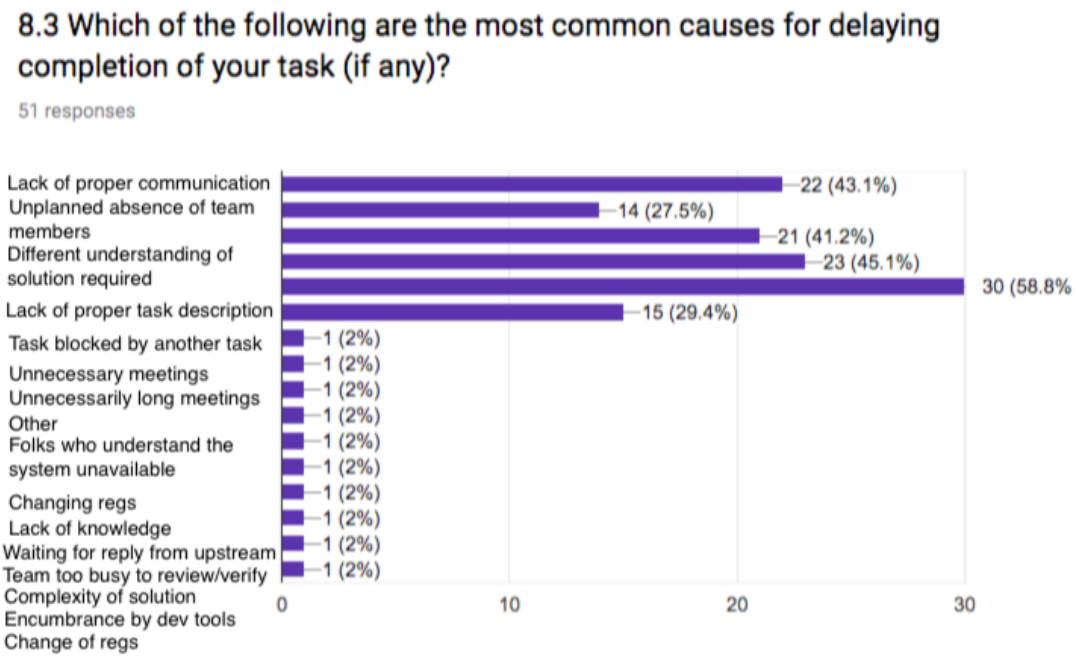

FIGURE B.66: Reasons for task completion

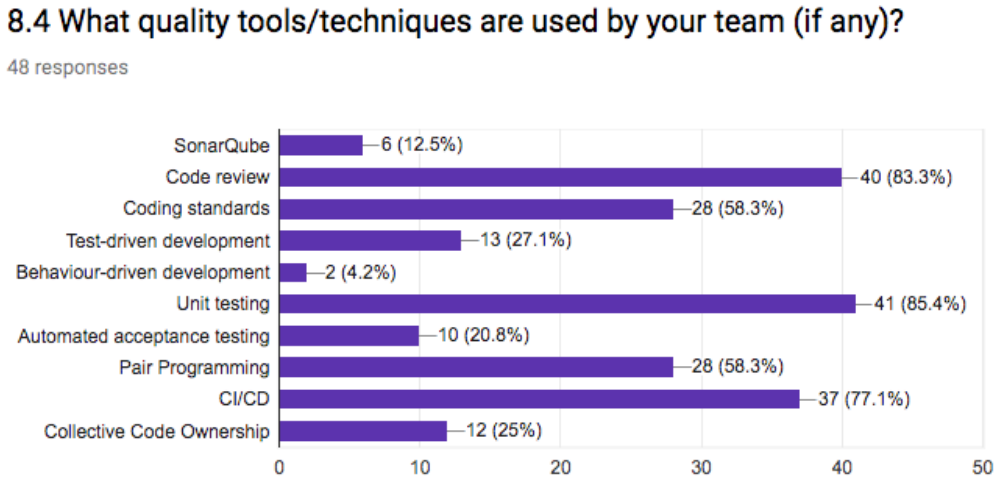

FIGURE B.67: Quality tools used

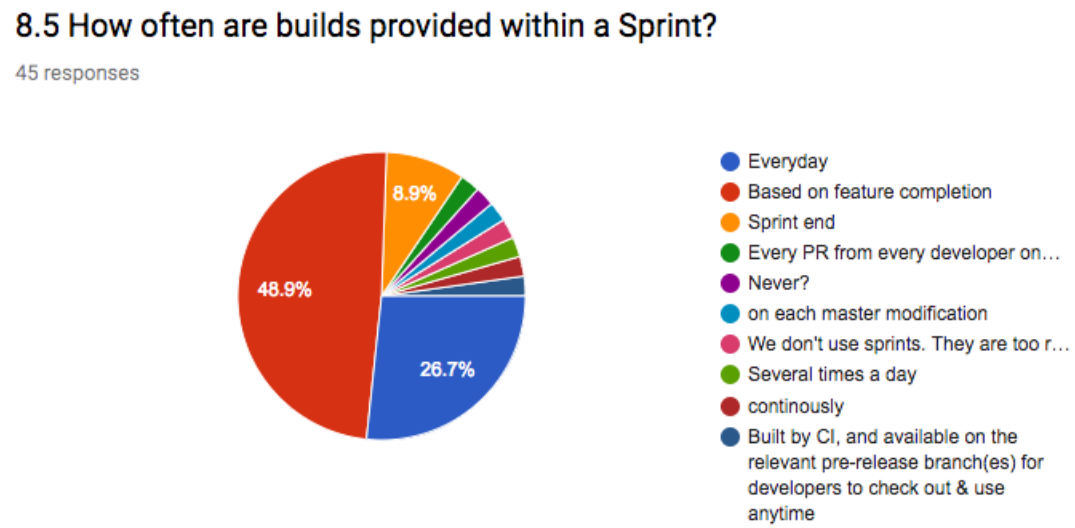

FIGURE B.68: Builds release within Sprint

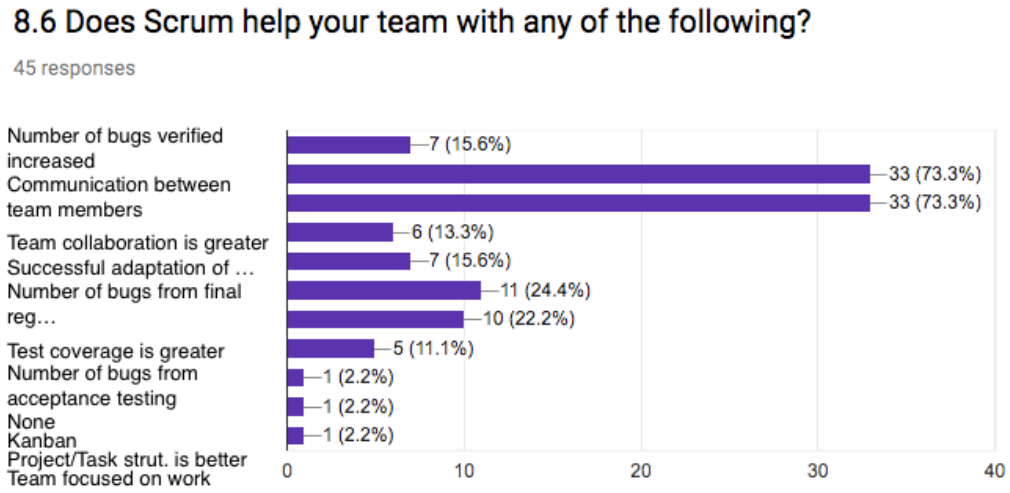

FIGURE B.69: Assistance of Scrum in various aspects of quality

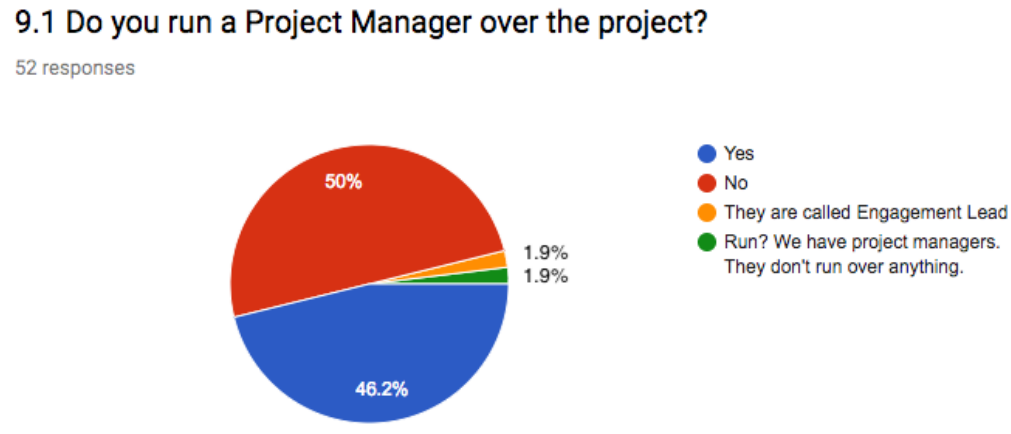

FIGURE B.70: Involvement of PM in project

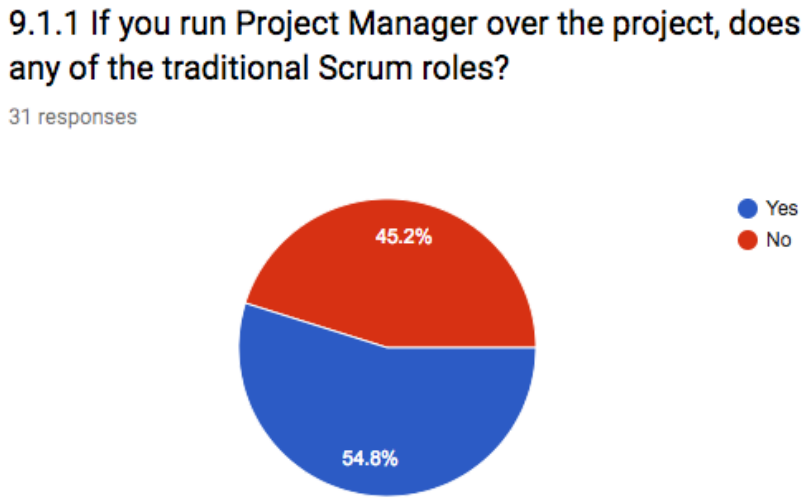

FIGURE B.71: PM taking traditional Scrum roles

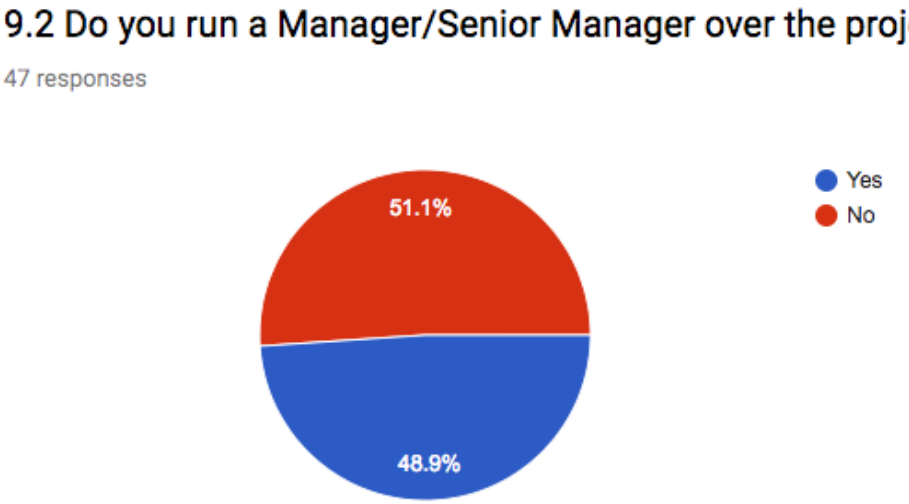

FIGURE B.72: Involvement of Manager/Senior Manager in project

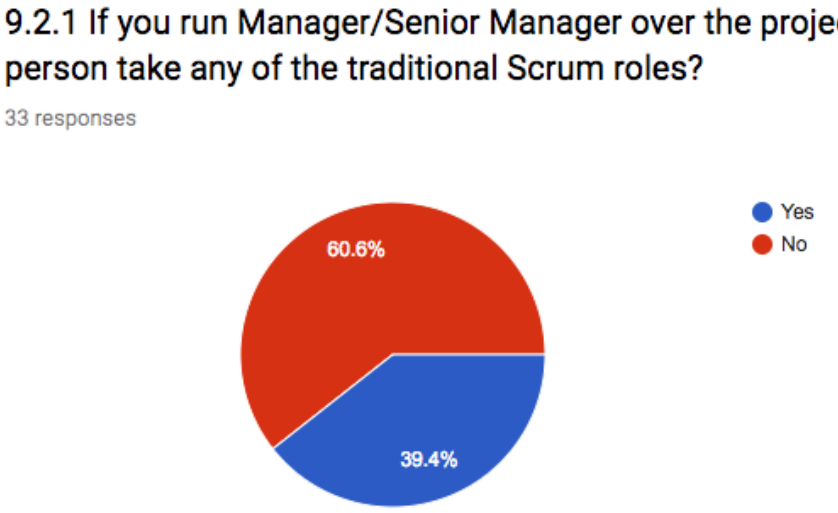

FIGURE B.73: Manager/Senior Manager taking traditional Scrum roles

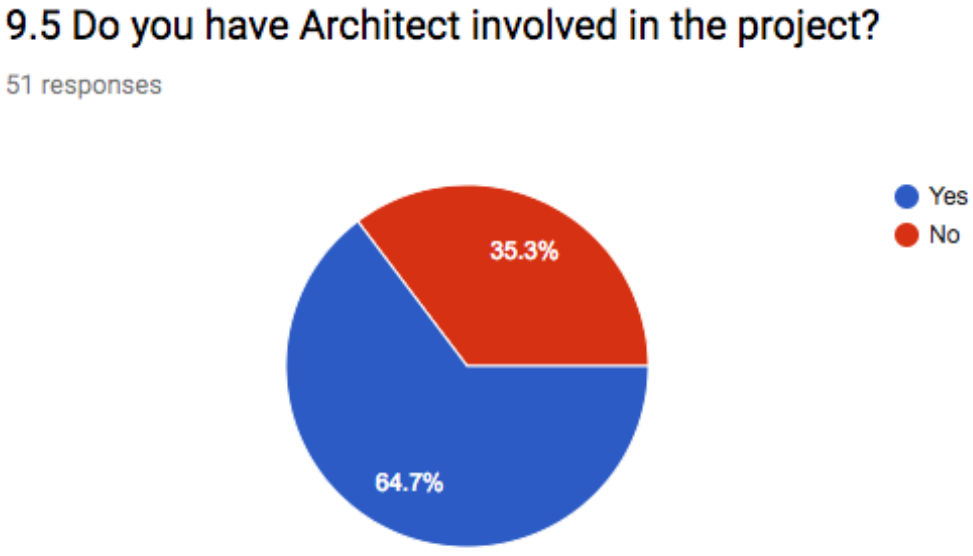

FIGURE B.74: Involvement of Architect in project

FIGURE B.75: Involvement of Sales team in project

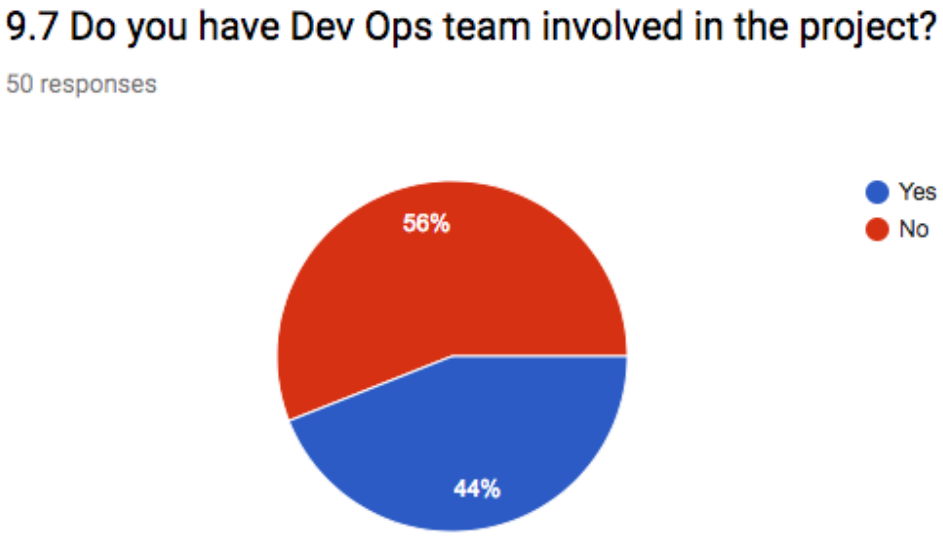

FIGURE B.76: Involvement of DevOps in project

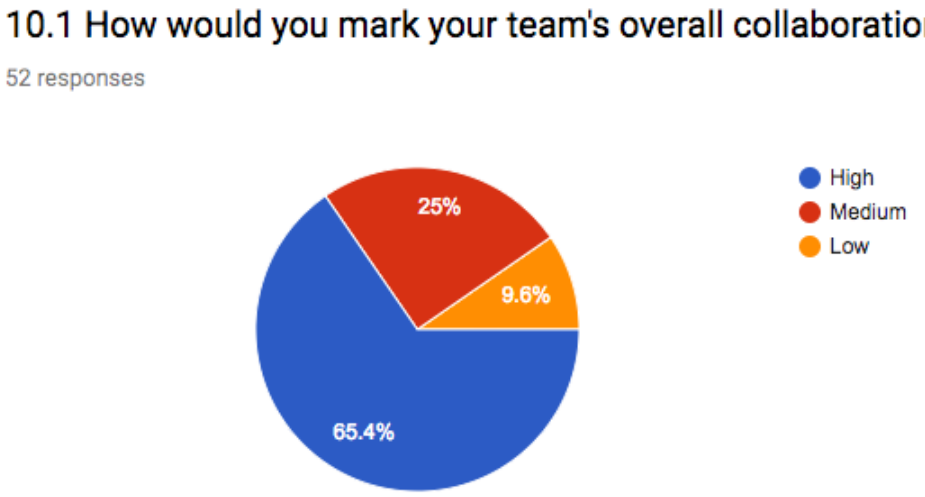

FIGURE B.77: Overall team collaboration

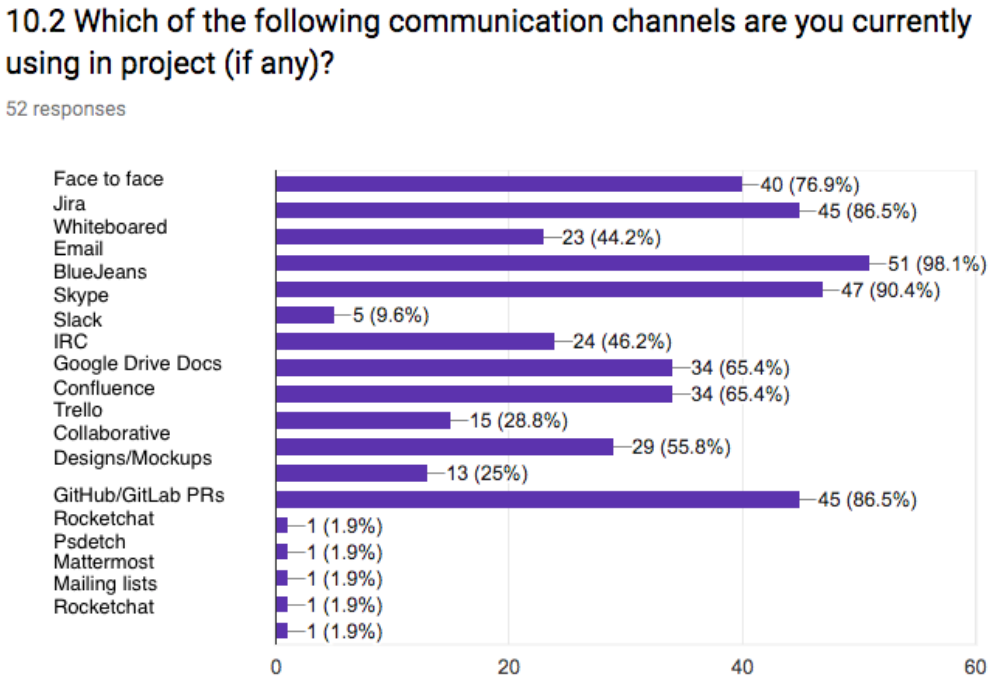

FIGURE B.78: Commonly used communication channels

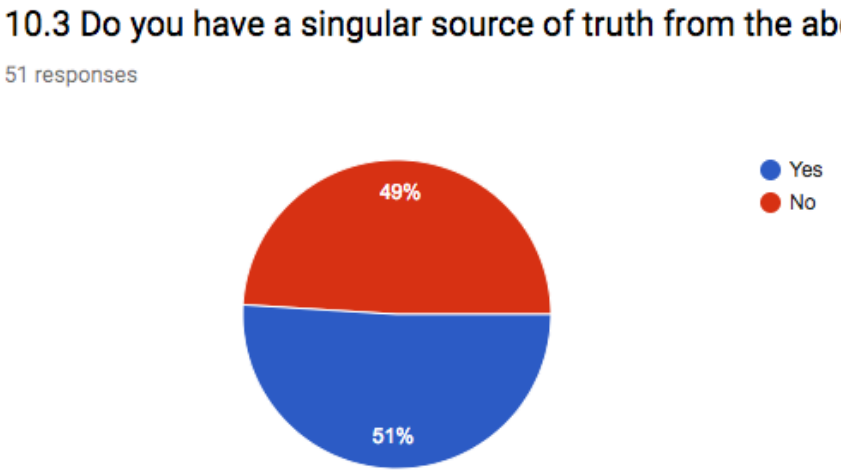

FIGURE B.79: Trust percentage in the communication channels

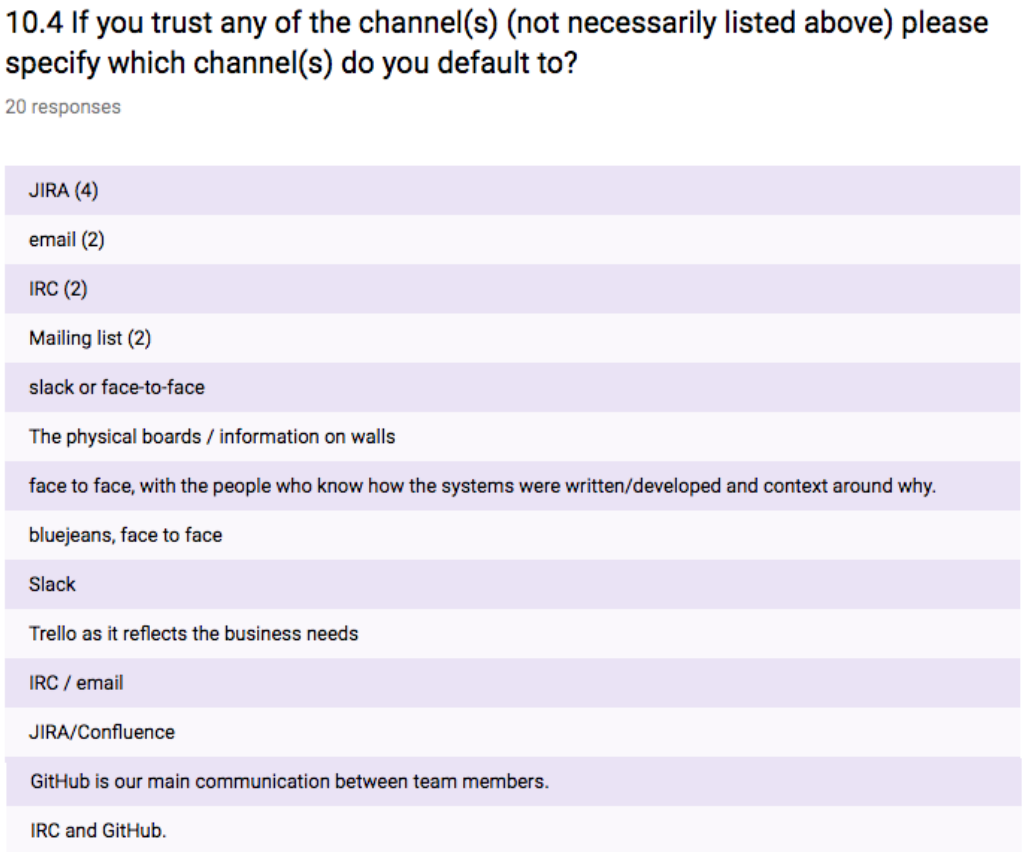

FIGURE B.80: List of default channels

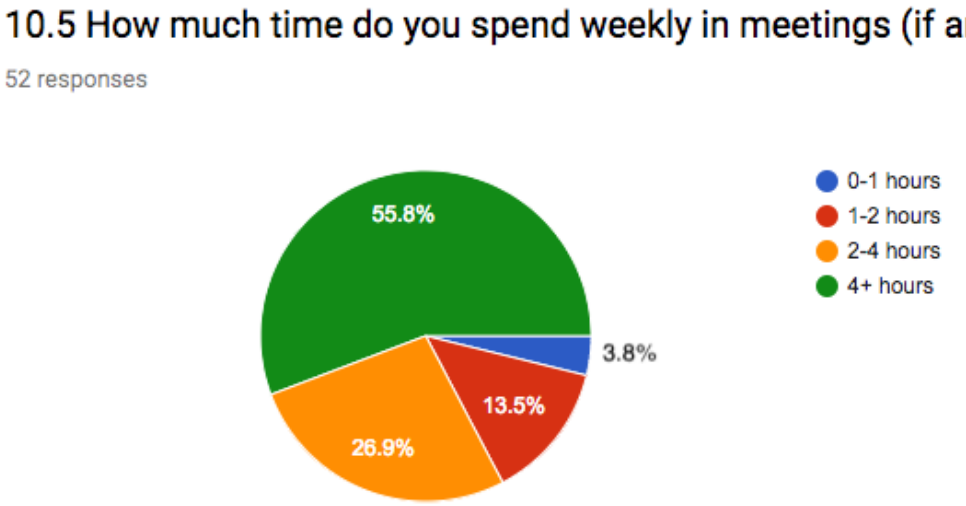

FIGURE B.81: Time spent in meetings weekly

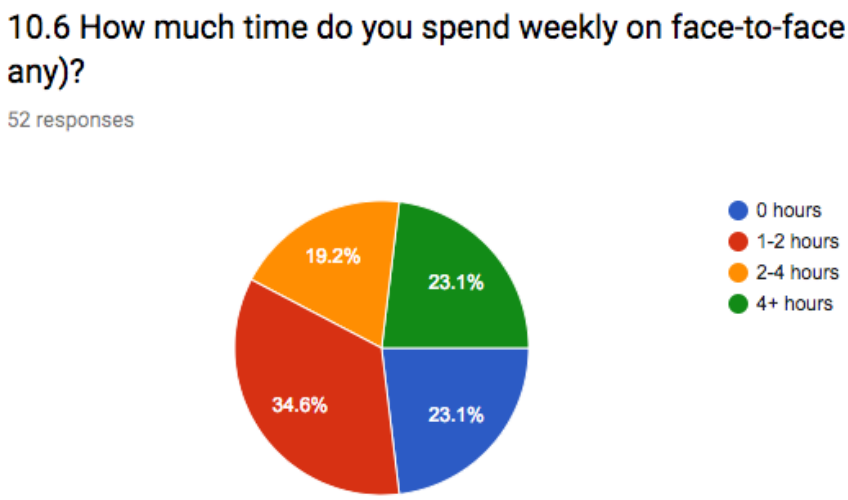

FIGURE B.82: Time spent in face to face meetings weekly

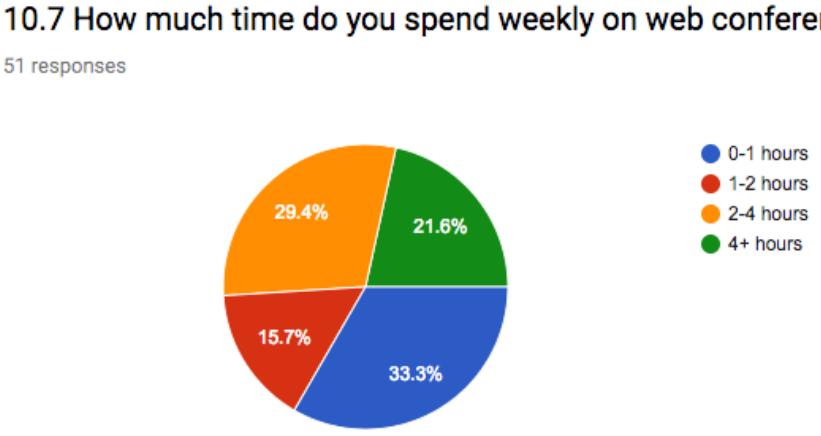

FIGURE B.83: Time spent in web conference meetings weekly

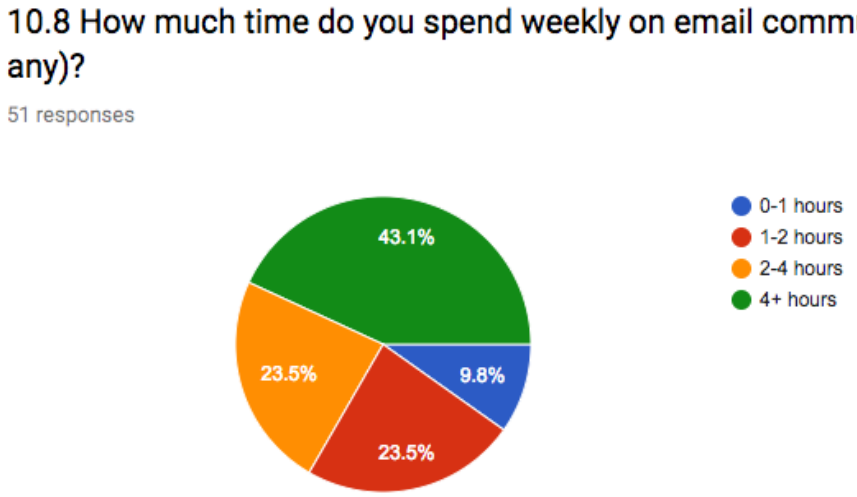

FIGURE B.84: Time spent on email communication weekly

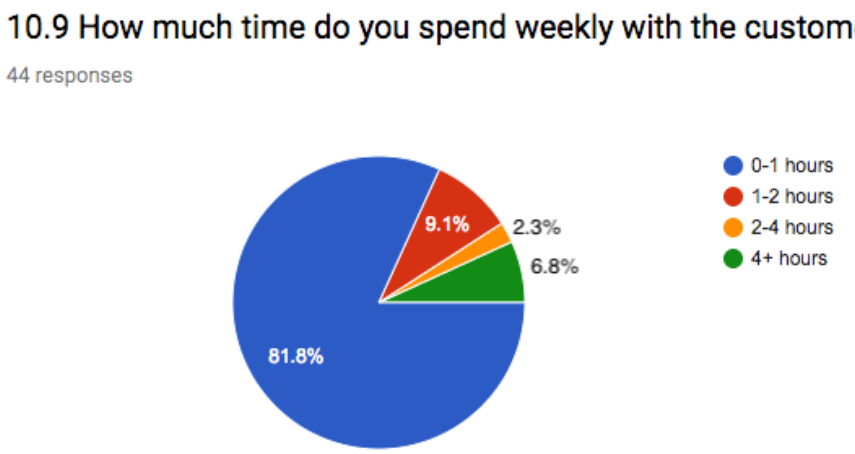

FIGURE B.85: Time spend with the customer weekly

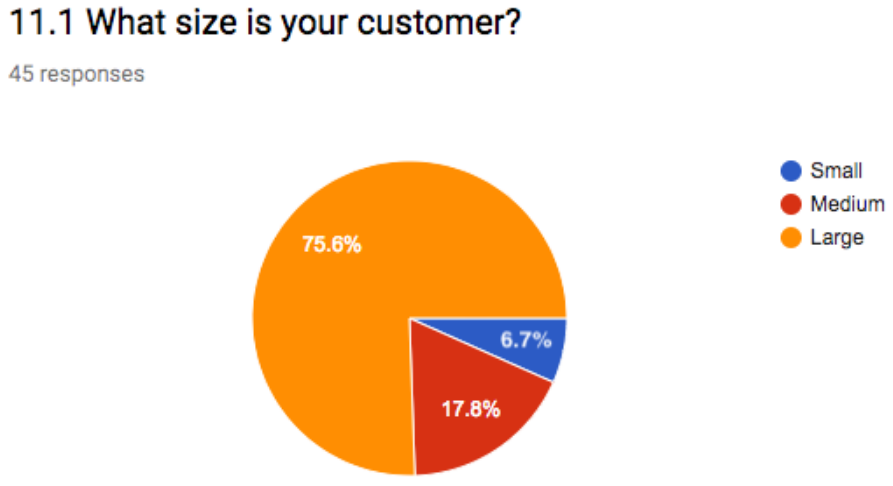

FIGURE B.86: Customer's size

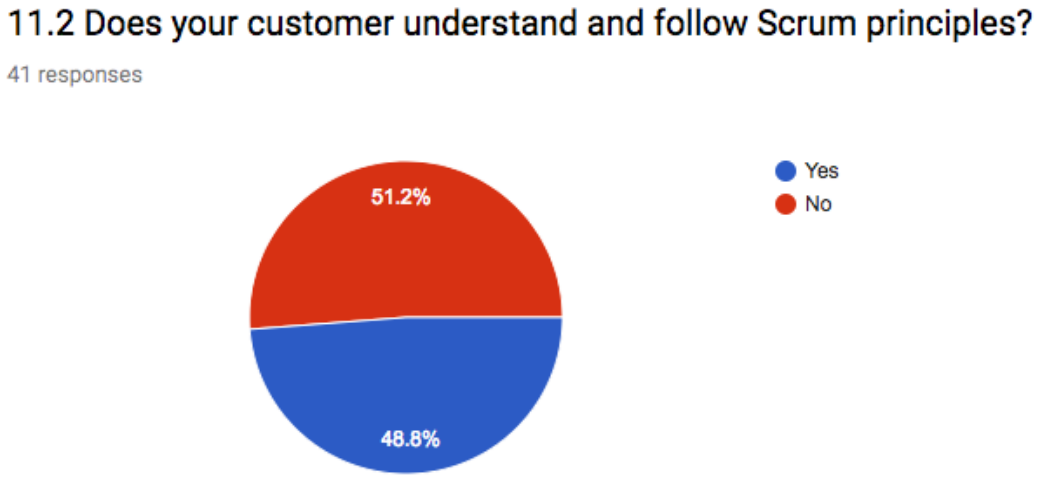

FIGURE B.87: Customer's knowledge of Scrum

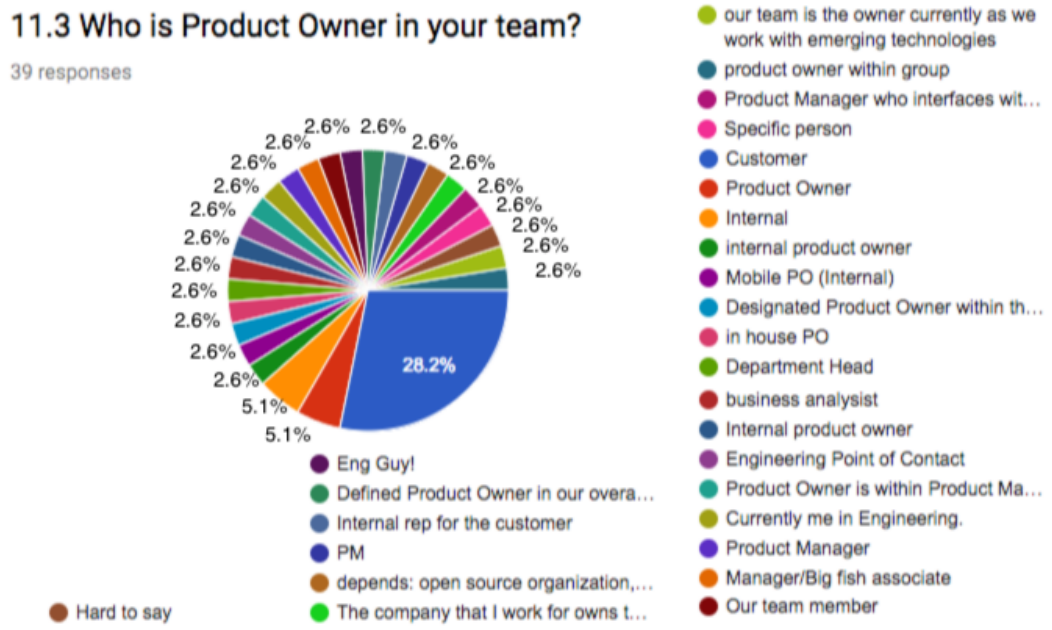

FIGURE B.88: Product Owners in projects

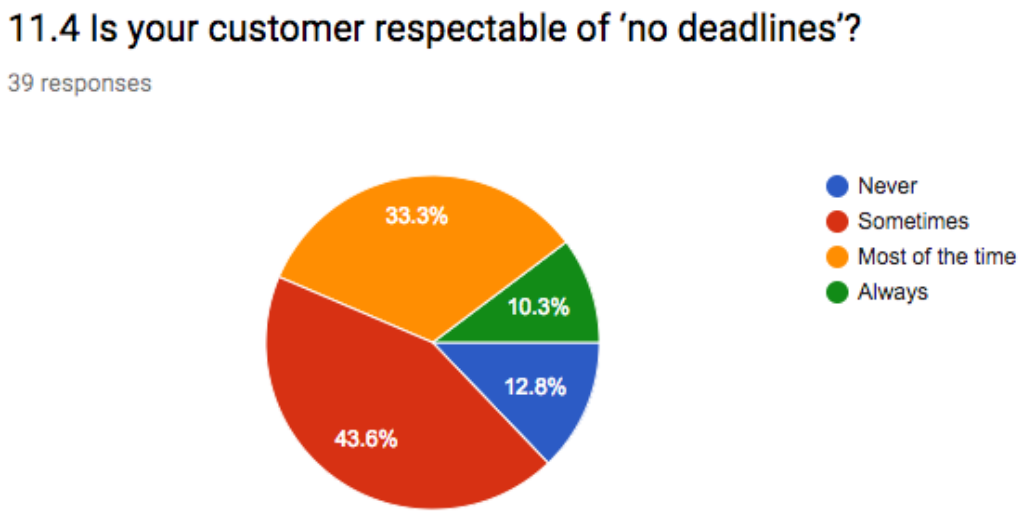

FIGURE B.89: Customer's respect of 'no deadlines'

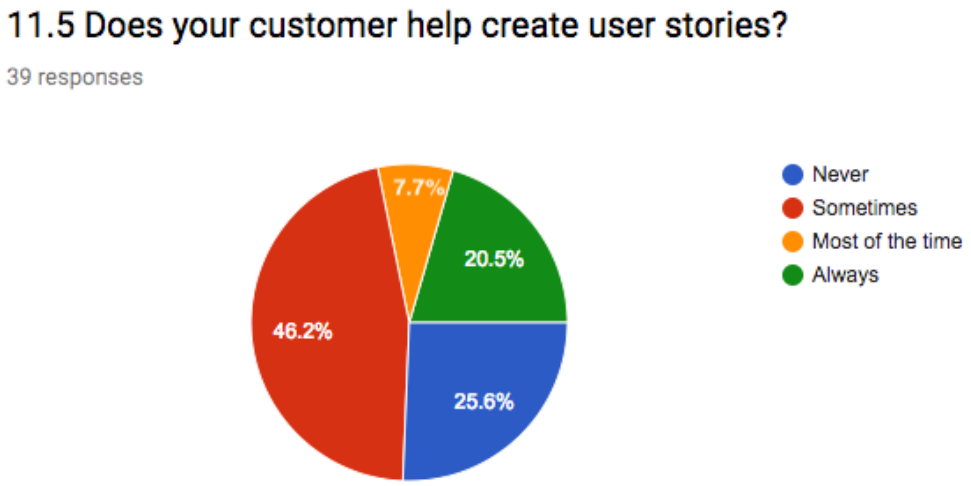

FIGURE B.90: Customer's involvement in User Stories creation

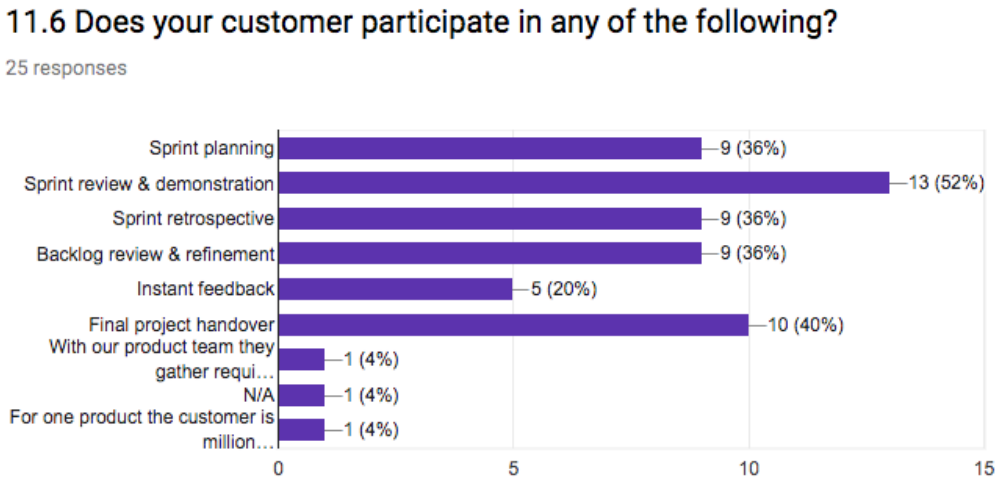

FIGURE B.91: Customer's participation in Scrum ceremonies

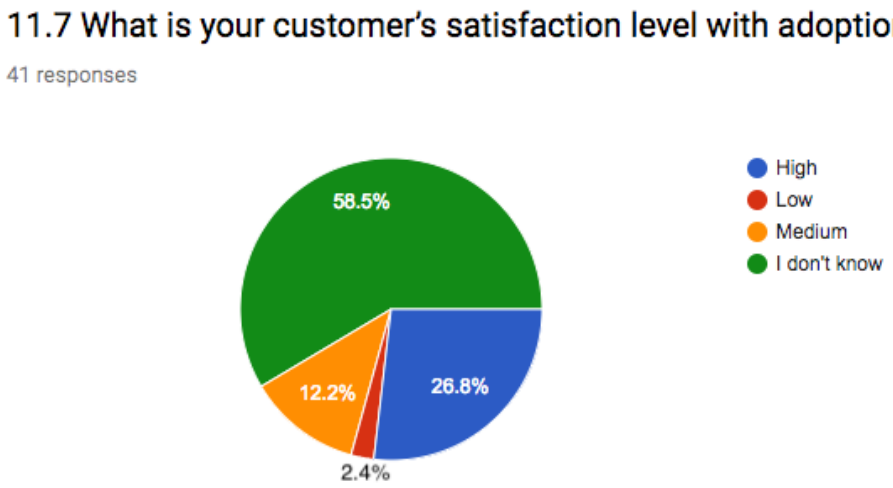

FIGURE B.92: Customer's satisfaction with Agile

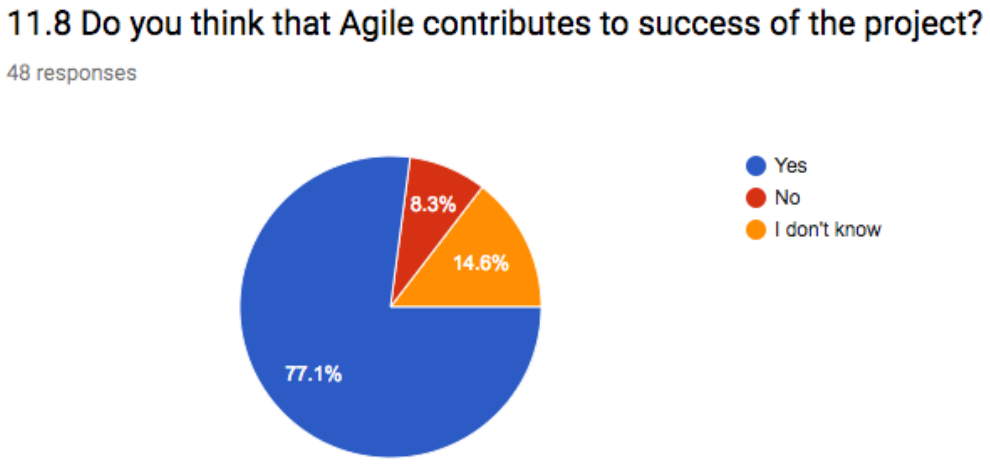

FIGURE B.93: Participant's rating of projects success with Agile

Appendix C

Scrum Ceremonies, Principles and Roles

The four traditional and commonly used Scrum ceremonies and principles are as follows:

- **Daily Scrum** - The Daily Scrum is the opportunity for the team members to meet daily, provide individual update on work completed, work in progress and any blockers interfering with the Sprint goal. Daily Scrum (also known as Daily Standup) is not a status update meeting, but a meeting focused on highlighting any impediments.

  It's the Scrum Master's responsibility to help the Development Team to stay focused on delivering the work planned prior to start of the Sprint by clearing these impediments.

  It's recommended that Daily Scrum should take no longer than 15 minutes which is enough time to allow the Development Team to answer three questions and listen to other participants answers. What did you do yesterday?, What will you do today?, and Are there any impediments in the way? are the three questions. Any other business should be discussed as part of a different meeting.

  Absence of Product Owner in Daily Standup meetings is optional.

- **Sprint Planning** - Sprint Planning is designed for creating a list of work items known as Sprint Backlog for the Development Team to deliver in the Sprint. Sprint Planning takes place prior to start of a new Sprint and should not last longer than 4 hours for a two week Sprint.

  This meeting should be attended by Product Owner, Scrum Master and Development Team. Product Owner is responsible for keeping the Product Backlog up to date and prioritized with the most desired functionality listed at the top of the Backlog prior to Sprint Planning session.

  Keeping the Product Backlog in order means having acceptance criteria, requirements, and clarifications to any questions or assumptions addressed for the Development Team to be able to estimate the work for upcoming Sprint.

- **Sprint Review/ Demo** - The Sprint Review also known as Sprint Demo is the showcase of all work completed in the last Sprint. Product Owner, Scrum Master and Development Team are expected to attend the meeting which should not last longer than two hours for a two week Sprint. Demonstrated software should be functional and meet definition of done.

  During Sprint Demo, it's important that Product Owner and stakeholders provide the team with a constructive feedback.

- **Sprint Retrospective** - The Sprint Retrospective is a communication platform for members of the team to reflect on the work completed in the last Sprint and identify any actions that would help them to improve in the future Sprints.

  The meeting should not be longer that 1.5 hours for a two week Sprint and it should focus on the three aspects: What went well over the last Sprint?, What didn not go so well?, What could we do differently to improve? All actionable results should be collected from the participants including Scrum Master, the Development Team and/or Product Owner and assigned to them.

The three roles of the Scrum framework are as follows:

- **Product owner** - Product Owner represents the client and the business and therefore holds the vision for the product. He/ she is responsible for keeping the Product Backlog up to date and prioritized every Sprint.

- **Scrum Master** - Scrum Master assists the team with the best use of Scrum to build the product. He/ she is responsible for making sure that any blockers that Development Team is facing are cleared. Coach, facilitator, counselor, impediment-remover, mediator best describe the role of Scrum Master.

- **Development Team** - Development Team is a group of cross-functional, self-organizing and motivated team members responsible for building the product as per Product Backlog specification. Development Team consists of developers, QA/ QE, designers and other technical roles.

Appendix D

Barriers to Small Scale Scrum

The following barriers to Small Scale Scrum are ordered with the most important at the top of list:

1. **Improper communication** - Clear communication provides clear direction and it can be supported by the use of suitable communication channels. Sufficient task description, the same understanding of solution required, willingness to initiate or participate in requirements clarifying meetings, discipline for the avoidance of assumptions and efficient communication of task related progress. They all contribute to successful communication within and outside of the team. The benefit of effective communication is genuine motivation, team unity and common vision resulting in high productivity, happy team members and the willingness to go the extra mile.

2. **Lack of knowledge sharing** - Knowledge sharing makes problem-solving experiences reusable helping individuals to save time and prevent them from reinventing known solutions, enables better and faster decision making, stimulates innovation and growth, improves delivery to customers and reduces the loss of know-how.

3. **Multifunctional roles** - Changing tasks on daily basis proves to be difficult and is associated with a loss of time required on getting up the speed with the project and codebase prior to any implementation of change requests. Multifunctional roles require ability to act as Business Analyst, Developer, Tester and Delivery Manager as one. Structured roles can fundamentally improve productivity and quality of delivered solution.

4. **Lack of or poor planning** - Failure to plan work properly is a waste of time and money. Lack of or poor planning is firmly associated with increased risk of the project failing. Progressive planning should result in less time required for planning as the project matures. Planning is critical to the success of any project and should be carefully prepared with the customer.

5. **Lack of or poor team organization** - Role definition is an integral part of team organization and accounts for much more free-flowing and effective communication and quicker responsiveness to issues.

6. **Lack of or poor work prioritization** - Work prioritization has a whole range of benefits. It reduces stress and increase productivity, helps create room to check errors, helps avoid procrastination and keeps the team motivated.

7. **Inexperienced team members** - Inexperienced team members should be supported through training and mentoring for the best results. Best practices and

processes governing development, testing and delivery should be clearly communicated.

8. **Unavailability of mentoring** - Well structured and professionally delivered mentoring program can greatly benefit mentors, mentorees and organization. On the other hand, mentoring with the wrong qualifications, profile or expectations can create a long-lasting damage. In mentoring, it's important to allow mentorees to choose their mentor. Mentoring can be delivered online and onsite with a structured learning plan focused as possible to learning goals.

9. **Lack of commitment** - Lack of commitment between team members is a sign that the individuals haven't investigated the issues properly. This is not about reaching a final conclusion, but rather buying-in despite of doubts and disagreements from other team members. There are number of ways to build commitment in teams including making sure that team members feel valued, avoiding forcing individuals into working on a project, aligning roles and responsibilities around team members strengths and interests, building trust, challenging to avoid boredom, encouraging creativity.

10. **Lack of teamwork/ team cohesion** - A lack of leadership, the presence of disruptive personalities, lack of proper training, lack of defined goals, lack of incentive, not taking into account team members strengths and weaknesses, fear of failure, tot enough team meetings, individuals don't feel included are common reasons why teamwork fail. Teamwork can be improved by taking some critical steps such as leading by example, building up trust and respect, encouraging socializing, cultivating open communication, precisely outlining roles and responsibilities, organizing team processes, setting goals, recognizing good work, allowing team members to actively take part in decision-making, maintaining the balance of work, meeting regularly, avoiding micromanagement, creating space, giving regular feedback and celebrating successes.

11. **Lack of motivation** - Lack of motivation within the team can be caused by a number of factors including ineffective communication, disorganization of roles, lack of direction, poor work organization and prioritization, unavailability of mentors, lack of trust and transparency and lack of knowledge sharing just to name a few.

12. **Lack of or poor quality delivery** - Poor software quality delivery can be caused by a number of factors such as lack of domain knowledge, lack of technology knowledge, Unrealistic planning, badly engineered software, just to name a few. Some of these could be addressed by providing access to domain experts, training developers in different application technologies, planning properly, re-factoring portions of the code,

13. **Lack of trust & transparency** - Lack of trust and transparency can have a very negative impact on performance or even destroy the team, but it can be prevented. Simply communicating (telling people what's going on) would be considered as good start, listening, showing care and recognizing could increase trust and transparency.

14. **Unsuitable project management tools and processes** - Selecting a suitable project management tool e.g Jira and keeping the processes simple could help address this issue.

Appendix E

Daily Scrum Questions

The following are the three common questions for Daily Scrum meetings ***Agile Alliance (2018)***[52] which focus on completion of tasks:

1. **What have you completed since the last meeting?**
2. **What do you plan to complete by the next meeting?**
3. **What is getting in your way?**