CHALLENGES IN TRANSITIONING FROM WATERFALL TO SCRUM

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Rini Iyju

June 2015

COMMITTEE MEMBERSHIP

TITLE:	Challenges In Transitioning From Waterfall to Scrum
AUTHOR:	Rini Iyju
DATE SUBMITTED:	June 2015

COMMITTEE CHAIR:	Dr. Davide Falessi, Ph.D. Associate Professor of Computer Science
COMMITTEE MEMBER:	Dr. Gene Fisher, Ph.D. Professor of Computer Science
COMMITTEE MEMBER:	Dr. Timothy Kearns, Ph.D. Professor of Computer Science

ABSTRACT

Challenges In Transitioning From Waterfall To Scrum

Rini Iyju

The goal of this thesis is to investigate critical issues and challenges that occur during the transition from a traditional software development methodology such as Waterfall to Scrum. During the last decade, Scrum has gained a vast success in software development due to its lightweight character and efficient way of handling the challenges of increased market speed, change and product complexity.

This thesis is based on a sequential exploratory mixed methods research model, which uses both qualitative as well as quantitative research methods to investigate the problem. The rationale for this is that neither method is sufficient by itself to capture the trends and details of situations. When used in combination, both quantitative and qualitative methods complement each other and provide a more complete picture of the research problem.

There are six main results from this thesis. First the main challenges are identified. Second, they are ranked based on their frequency of occurrence and thirdly, based on their importance. Fourth, the correlation between the frequency of occurrence of challenge and their importance is measured. This thesis also examines the varied perspectives of Scrum Coaches and Scrum Practitioners regarding the frequency of challenges as the fifth result and regarding the importance of challenges as the sixth result.

ACKNOWLEDGMENTS

God – In God I trust; for through him, all things are possible.

Varun Abraham – For being my life's companion and a wonderful husband. I am thankful to God for bringing us together.

Aaron Abraham – For being my bundle of joy.

Dr. Davide Falessi – I would like to express my sincere gratitude you for stepping in as my advisor at a critical point in my thesis, for the continuous support of my study and research, for your immense knowledge, patience, motivation and enthusiasm.

Dr. Clark S. Turner – For giving me the freedom to explore on my own and at the same time providing me with guidance to recover when my steps faltered.

Dr. Gene Fisher– I am thankful for all your help, support, and advice especially when I was new to Cal Poly. I am also thankful that I could always walk into your office and ask for help with anything and everything.

Dr. Timothy Kearns – I am thankful to you for your insightful comments at different stages of my research. These were thought provoking and helped me focus my ideas.

Dr. Heather Smith – Thank you for unraveling the wonderful world of statistics and for always being available for help and advice.

My wonderful family- Thank you for being with me through everything and making my dream of getting my master's degree a reality.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

1.1. Foreword

Agile software development methodologies are becoming increasingly prevalent in the industry today [1]. Companies are moving to agile methodologies like Scrum from traditional processes like waterfall because the technology marketplace demands a high responsiveness to change. In order to compete in the global economy, companies must move quickly to provide solutions to a client base that has more and more choices available to them. Agile approaches promise faster delivery of working code, higher quality and a more engaged development team that can deliver on its commitments [2]. Traditional waterfall, with its long phases and heavy investment in 'big up-front design' lacks the flexibility to swiftly respond to the market [2]. In a waterfall model of software development, it may be difficult to change requirements during the process because it causes huge problems. Agile processes focus on a more incremental, non-bureaucratic method that focuses on delivering value and reflecting business needs.

Agile processes also provide a larger return on investment by decreasing the investment in inventory, decreasing operating expenses and increasing throughput [3]. In other words, agile methods save cost by eliminating the time and money spent on designing an entire system, which may be outdated before it is implemented and may have numerous pieces that are never coded. By adopting an Agile Methodology, software development teams are able to provide rapid value delivery to customers, resulting in greater profits [9]. The transition to agile process is a growing trend that will

have lasting effects on the industry. It is a different way of work, one that requires greater communication and cooperation from its participants and greater leadership from its managers [18].

The flexibility and the responsiveness that agility offers has caused a noticeable reduction in the use of traditional methods, and companies are increasingly adopting agile methods such as eXtreme Programming (XP), lean software development, Crystal, Scrum etc. [1].

All of these methods share an iterative and incremental approach, but Scrum remains the most popular method [2]. Scrum is an agile software development process that focuses on project management practices [3]. It has gained considerable popularity in large companies. The term Scrum comes from rugby [4], where the players have to work together to take the ball and pass it to each other through the different rows to win. In addition to the name, many strategies from rugby were adapted and used in the Scrum development process, for instance the team integration strategies and keeping the same core team members during the project [4].

Scrum relies on a fixed cadence of iterative cycles called sprints. Each sprint begins with a planning meeting and ends with the demonstration of a potentially shippable product. Scrum is characterized by a high level of feedback and transparency, both within the team and outside it. Its short cycles and collaborative nature makes it ideal for projects with rapidly changing requirements [5].

The main motivation for software companies using Scrum is being able to adapt when the requirements are not predictable [5]. It relies on ceremonies and meetings with the client and within the development team such as daily Scrum

meetings and sprint reviews, and it uses techniques that facilitate the visualization and communication of the workload [6]. The process itself allows for building the product in a more progressive way by planning small parts that fit on previous parts and getting them approved by the project stakeholders before moving on to the next step [7].

1.2. Statement Of The Problem

Scrum is one of the easiest frameworks to understand; yet it is one of the hardest frameworks to implement well [8]. Scrum's inherent simplicity can make people believe that it is easy to do, but the reality is not so. Scrum goes against what has been learned through many years of implementing traditional software development. To do Scrum right, the fundamental way of developing software has to be changed [3].

The attractiveness of agile software development methods is undeniable, but being agile is not an easy task. It presents many challenges in terms of team management and perception of the Scrum roles [12] and challenges due to adopting a new culture, attitude and practices [14].

In addition to that, there are challenges related to new communication arrangements with the team and the customer [15] as well as the ones related to customer involvement. There are also challenges related to the new decision making procedures [16].

A considerable amount of literature [17] [18] [19] has been published on the challenges related to Scrum, but very few of them point to the challenges that occur during a Scrum adoption and may hinder the completion of this transition. While the use

of Scrum keeps growing, more should be investigated about the challenges that occur during the transition and can last for years after adopting this method.

Often times, large companies fail to go through a successful Scrum transition in spite of investing considerable time and resources into it. But despite the growing popularity of Scrum and the unfortunate failures that some companies that adopt it have to face, there is very little scientific study on the challenges that software engineering teams face during a Scrum transition.

Thus, the problem that practitioners face is the deceiving simplicity of Scrum. The Scrum framework is easy to understand, but difficult to implement. Organizations that undergo a Scrum transition are not aware of the challenges that they are most likely to be faced with as they go through a transition from waterfall to Scrum. This thesis thus aims to study this problem and understand and prioritize the challenges that software teams face when they transition from a Waterfall software development life cycle to a Scrum methodology.

1.3. Aim Of The Study

This thesis is an attempt to close the gaps in the present knowledge about the challenges that occur during a Scrum transition. It focuses on uncovering the main challenges that software teams face when they transition from a traditional Waterfall model to a Scrum methodology.

While there are numerous case studies at individual organizations, there are few [27] if any that look into challenges that software engineering teams encounter from a

more holistic point of view. This study aims at contributing to the knowledge gap in this field.

1.4. Research Questions

This primary focus of the thesis can be broken down into the following five research questions.

R.Q. 1: What Are The Most Frequent Challenges?

This question aims to find the most frequent challenges that software development teams in large organizations face while transitioning from a traditional Waterfall Software development model to Scrum.

R.Q. 2: What Are The Most Important Challenges?

This question aims to find the most important challenges that software development teams in large organizations face while transitioning from a traditional Waterfall Software development model to Scrum.

R.Q. 3: Is There Correlation Between The Frequency And Importance Of Challenges?

This question aims to find whether there is a correlation between the frequency and importance of challenges that software development teams in large organizations face while transitioning from a traditional Waterfall Software development model to Scrum.

R.Q. 4: Regarding Frequency Of Challenges, Do Coaches Agree With Practitioners?

This question aims to find whether the views of Scrum coaches and general Scrum practitioners are in alignment with regards to the most frequent challenges faced during a Scrum transition.

R.Q. 5: Regarding Importance Of Challenges, Do Coaches Agree With Practitioners?

This question aims to find whether the views of Scrum coaches and general Scrum practitioners are in alignment with regards to the most important challenges faced during a Scrum transition.

In the above questions, frequency refers to the number of occurrences of the effect and importance refers to the size of the effect. By answering the above research questions, this thesis aims at providing an improved, empirically based and more precise understanding of challenges faced during a Scrum transition.

1.5. Significance To The Field

The Scrum framework is easy to understand, but difficult to implement. Organizations that undergo a Scrum transition are not aware of the challenges that they are most likely to be faced with as they go through a transition from waterfall to Scrum. Successful research in this area can make companies that are about to undergo a Scrum transition aware of these challenges and be prepared for them by taking preventive steps.

1.6. Structure Of The Paper

The following chapters of the thesis will not only attempt to answer the research question but also discuss the Scrum methodology. The goal of this thesis is to investigate

the problems faced by software companies when transitioning from waterfall to an agile methodology. Scrum is considered to be the representative methodology here. Chapter 2 is the Background section, which sets a theoretical foundation for the topic with the intent of developing a framework for empirical study. Chapter 3 outlines the Research Methodology. Chapters 4 and 5 contain the qualitative research section and the quantitative research section respectively. Chapter 5 also includes the results and discussion section of the findings of this thesis. The objective of chapter 5 is to analyze the empirical research from the previous section and form meaningful conclusions, which are presented in chapter 6.

CHAPTER 2. BACKGROUND AND RELATED WORK

In this section, the literature related to software development methodologies are examined. The focus of this review is to identify and understand the challenges of using Scrum and building an understanding on what previous research has addressed so far.

2.1. Software Development Life Cycle

The software development life cycle is a term used in software engineering, systems engineering and information systems to describe a process for planning, creating, testing and deploying an information system [17]. A considerable amount of literature has been published on the characteristics of software development projects. These studies show that software development projects are very complex and need a high level of integration to succeed [18].

Another feature of software development projects is the considerable level of uncertainty involved [19]. The unpredictability of both the tasks and the software that software engineers perform make the necessary resources and the duration of the project hard to estimate [20].

There is also considerable uncertainty when it comes to defining requirements. It is hard to foresee all the requirements at the start of a project. That is why software development projects require a lot of flexibility to manage the changes and adjust to the changing requirements as the project develops [21].

2.2. Software Development Methods

A software development methodology in software engineering is a framework that is used to structure, plan and control the process of developing an information system. They can be classified into two main types depending on the constraints that they encounter which are cost, quality and time related issues. In this regard, literature acknowledges two distinct methods of software development, which are traditional and agile methods [22].

2.2.1. Traditional Software Development Methods

The traditional software development methods are based on a plan-driven approach using a standard development process built mainly around a waterfall model [23]. This requires an exhaustive and fully documented set of requirements [24], and focus on an elaborate planning phase [25].

As per traditional software development methods, development should be managed by following a number of well-defined steps. There are distinct goals for each phase of development where each phase is completed before the next one and there is no going back to the previous phase.

The goal of these methods is to focus on the process and proceed through the steps one after the other with a focus on the project milestones [27]. This brings inflexibility to the development process and goes against the rapidly changing environment that software development belongs to [28].

These methods are highly effective when it comes to well defined projects with fixed requirements [29]. But, they are based on predictable goals and are not suitable for changing requirements. Thus, they are becoming increasingly unsuitable to the

rapidly changing technologies and requirements of today's world. A new way of doing projects had to be adopted which are called agile methods.

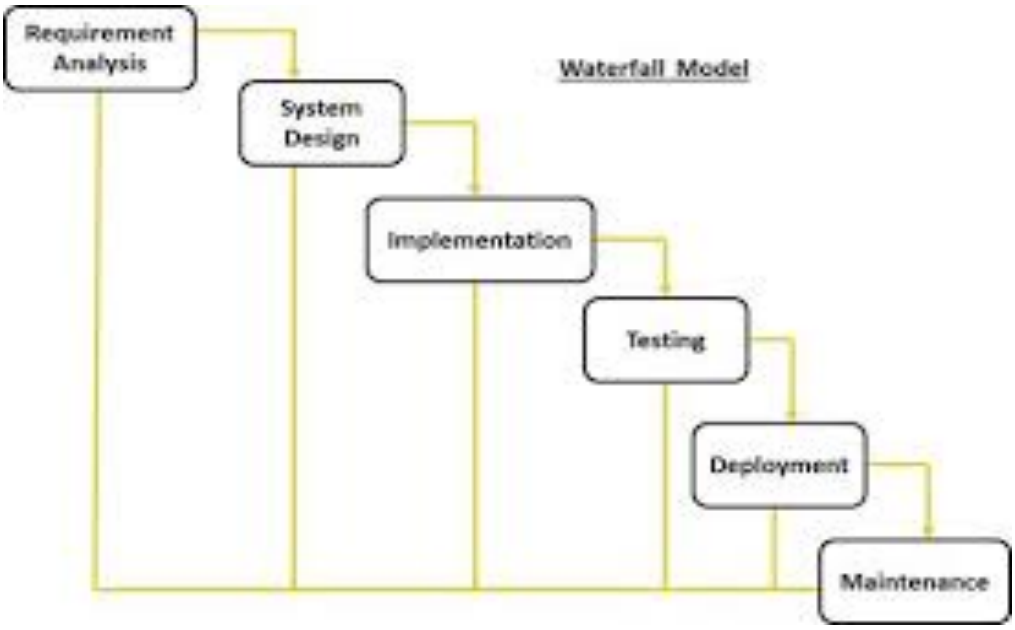

Fig 1: Waterfall Software Development Lifecycle

2.2.2. Agile Software Development Methods

Agile is an attribute often associated with animals like the big cats. Alert and responsive, quick and well coordinated in movement [46]. These are also features that can be associated with agile software development methods. The Scrum framework is considered to be one of the most influential and popular agile methods, so in the following sub chapter, an overview of agile development and its principles are described.

Agile Development as a term was introduced in 2001 during a two day meeting between seventeen people gathered at Snowbird Ski Resort in Utah [2]. The people gathered here were representatives from various disciplines in software development

trying to establish common ground. Ken Schwaber and Jeff Sutherland, the founding fathers of the Scrum framework were two of the people gathered. The outcome of the summit was the Manifesto for Agile Software Development, which has a vastly influential role software development in today's world.

The Agile Manifesto is based on four values and twelve principles:

2.2.2.1. The Value Set Of Agile Software Development [2]

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

- *Individuals and Interactions* over *Processes and Tools*
- *Working Software over Comprehensive Documentation*
- *Customer Collaboration over Contract Negotiation*
- *Responding to Change* over *Following a Plan*

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

2.2.2.2. Principles Behind The Agile Manifesto [2]

- 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 time scale.

- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they 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.

2.3. What Is Scrum?

Scrum is a framework for managing software development. It is based on iterative development cycles called Sprints. Sprints last two to four weeks and follow each other. The results of the development activities in each sprint are potentially shippable increments. In Scrum, development activities are broken down into small sub parts that can be done within a sprint. This relationship between time and workload establishes a

common commitment in the development team. It also sets the scene for formal events and meetings in the Scrum framework [3].

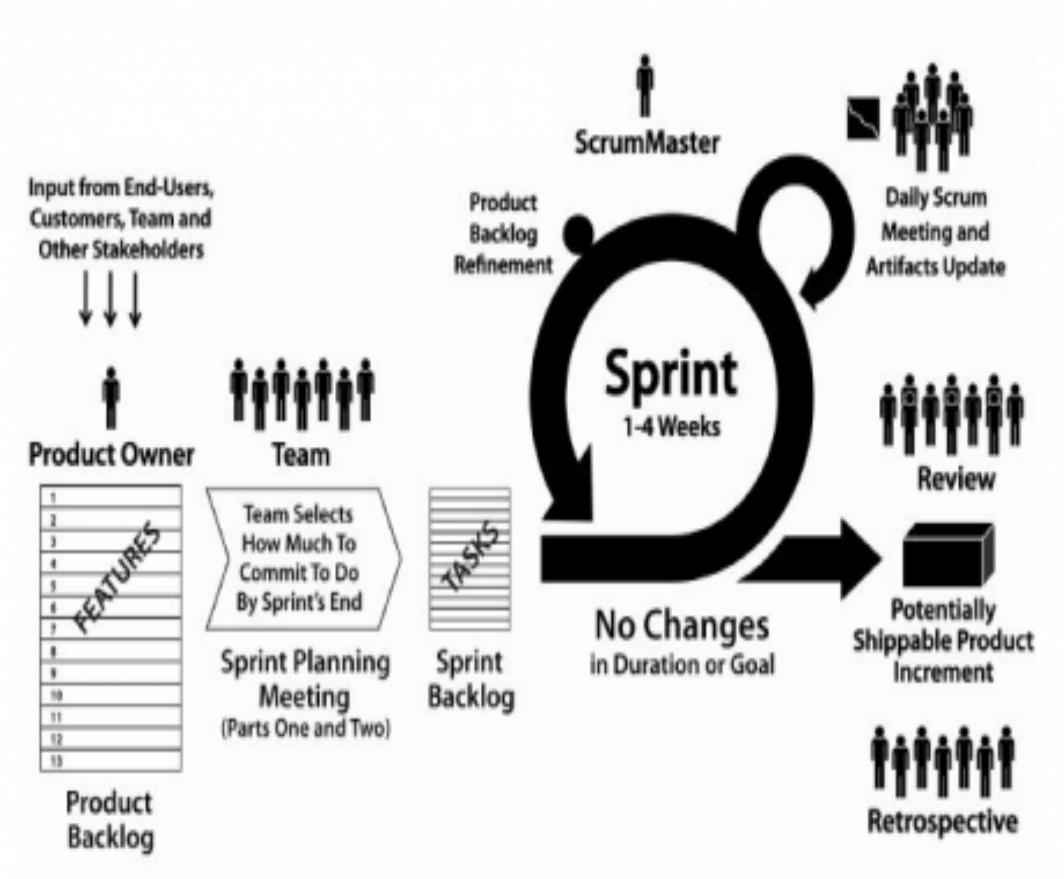

Fig 2: Scrum software development methodology

The following description of the Scrum roles, artifacts and ceremonies is taken from The Scrum Guide, 2014 [3].

2.4. Scrum Roles

Ideally, each scrum team should be composed of 5 to 9 team members functioning in different capabilities. Each scrum team should have a product owner, a scrum master and the development team.

2.4.1. Product Owner

The product owner owns the product in development. He or she represents the full product responsibility and is accountable for making decisions that will drive development in the direction that will create maximum value for the customer.

2.4.2. Scrum Master

The Scrum Master is responsible for maintaining a flow in the development process in accordance with the instructions of the Scrum framework. This means facilitating daily meetings with the team and making sure that the team has a common understanding of the product vision, the overall goals and the development tasks in the current sprint.

2.4.3. Development Team

The Development Team consists of the people who do the development and who have the responsibility of delivering the agreed upon results at the end. The size of the development team varies from three to seven persons and they are usually co-located to ensure clear and continuous communication amongst the team members. During a sprint, the development team is only working on tasks that lead directly to the goal of that particular sprint.

2.5. Scrum Artifacts

Scrum Artifacts consist of the Product Backlog, the Sprint Backlog and the Burn Down Chart.

2.5.1. The Product Backlog

The Product Backlog consists of all the functions, features, requirements and enhancements that constitute the changes to be made to the product in future releases. The Product Backlog is maintained by the Product Owner and is a dynamic document that evolves as product development progresses.

The product backlog may change all through the development lifecycle and functions as a document for later versions of the same product. The development team and the product owner continuously groom the backlog by adding items and making changes requested by the project stakeholder as development progresses.

2.5.2. The Sprint Backlog

The Sprint Backlog is a tool for the development team for making visible the work that the team identifies as necessary in order to meet the sprint goal. It is a dynamic document that develops throughout the sprint and has just enough detail for the development team to establish an overview of the remaining work and remaining time to do the work. It assists the team in planning the ongoing sprint and makes the work of the team transparent to the Product Owner.

2.5.3. Burndown Charts

The Burndown chart shows the total remaining work hours in one sprint on a daily basis. Teams become self aware of how much work they have done and how much work remains to be done.

2.6. Scrum Ceremonies

The Scrum ceremonies consist of the Daily Scrum meeting, the Sprint Planning Meeting and the sprint Review meeting.

2.6.1. Sprint Planning Meeting

Every Sprint begins with a Sprint Planning Meeting. Here, the Scrum Team, which comprises the Product Owner, the Scrum Master, and the Development Team plan the next sprint by deciding which tasks and activities have to be taken into the Sprint Backlog from the Product backlog.

2.6.2. Daily Scrum Meeting

The Daily Scrum is a 15-minute meeting held each day by the development team. It is placed in a fixed time slot and is thereby meant as a permanent and regular event occurring on a daily basis. The purpose of the daily Scrum is to synchronize the work and adjust the chosen approach to reaching the Sprint Goals. It is the Scrum Master's responsibility to facilitate the meeting and make sure that every Development team member makes himself heard.

2.6.3. Sprint Review

The Sprint Review is an informal meeting that is conducted at the end of every Sprint. It is usually two to four hours long where team members present an incremental working model of the product, which was completed in that Sprint to the product

owner and others. The discussions and considerations at the Sprint Review usually function as a basis for the Product Owner's input at the next Sprint Planning Meeting.

CHAPTER 3. METHODOLOGY OVERVIEW

3.1. Foreword

The purpose of this chapter is to ensure the clarity of the research process by presenting the research methods used for collecting and analyzing the research material.

3.2. Mixed Methods Research

To study and answer the research question, a mixed methods [4] approach was used which is a procedure for collecting, analyzing and integrating both qualitative and quantitative research data at some stage of a research process within a single study [12]. This study uses a sequential exploratory mixed method design, consisting of two distinct phases. It involves the collection of both qualitative (open-ended) and quantitative (closed-ended) data in response to research questions or hypothesis. It includes the analysis of both forms of data. The procedures for both qualitative and quantitative data collection and analysis need to be conducted. This includes adequate sampling, sources of information and data analysis steps. The two forms of data are integrated in the design analysis through merging, connecting or embedding the data. [12].

The rationale for mixing both types of data is that neither quantitative nor qualitative methods are sufficient by themselves to capture the trends and details of situations. When used in combination, both quantitative and qualitative methods complement each other and provide a more complete picture of the research problem [4]. Mixed methods research design draws on both qualitative and quantitative research and

minimizing the limitations of both approaches. At a practical level, it is a sophisticated, complex approach to research and provides a useful strategy to explain qualitative results with a quantitative follow up data collection and analysis.

3.2.1. Exploratory Sequential Mixed Methods Design

An exploratory sequential mixed methods design is one in which the researcher first begins by exploring with qualitative data, analyzing the data and then using the findings in a second quantitative phase. The intent of this strategy is to develop better measurements with specific samples of populations and to see if data from a few individuals in the qualitative phase can be generalized to a large sample of the population in the quantitative phase [12].

In this design, the qualitative text data is collected and analyzed first, while the quantitative, numeric data is collected and analyzed later. This second sequential analysis phase explains or elaborates on the qualitative results obtained in the first phase. The qualitative approach provides a general picture of the research problem by reducing the scope of the problem and paves the way for quantitative analysis, which in turn provides statistical results by exploring participant's views in greater depth.

CHAPTER 4. QUALITATIVE RESULTS & ANALYSIS

4.1. Qualitative Data Collection

The qualitative data collection was conducted by means of theme-centered interviews. The theme-centered interview allows interviewees to develop their special point of view in detail. The common theme in these interviews was the challenges that software development teams face when they transition from a traditional software development process to Scrum. The focus here is on the individual person and his or her experiences and opinions concerning the topic. An open conversation situation is established during the interview. Interviewees are given the opportunity to unfold and explain what is important to them in regards to the topic.

This chapter presents the data collected through interviews with six Scrum coaches. Each coach had over five years of experience leading Scrum transitions at large companies. They had worked with several Scrum teams and dealt with many challenges, which came in the way of successful Scrum adoption. Four of these coaches were based out of the United States; one was from France and one was from New Zealand. Each coach was asked to highlight the top challenges, which they felt were in the way of a successful transition from waterfall to Scrum.

4.2. Qualitative Data Analysis: Grounded Theory

The challenges, which were identified by interviewing the Scrum Coaches, were converged into a set of common themes. These themes were formed on the basis of open coding, active coding and selective coding which together comprise grounded theory.

4.2.1. Open Coding

Open coding refers to the initial phase of the coding process in the grounded theory approach. The initial stage of data analysis is called open coding because the process opens up the text of data in order to uncover the ideas and meanings it holds.

The process of coding begins with the collection of raw data, in this case - interviews. The intent of open coding is to break down the data into segments in order to interpret them. As many ideas and concepts as possible are developed without concern for how they will ultimately be used. Data segments are compared so that they may be grouped together as examples of the same concept or differentiated to form new ones.

4.2.2. Axial Coding

Axial coding follows open coding. In this step, data is reassembled so that relationships between them can be identified more readily. Axial coding is the phase where concepts that begin to stand out are refined and relationships among them are pursued systematically. Major categories begin to emerge at this stage.

4.2.3. Selective Coding

Selective coding is the last phase of analysis in the grounded theory approach to qualitative data. In this phase, explanations of phenomena begin to emerge and the analyst selects a central category.

4.2.4. Grounded Theory

This section presents how the data was examined to analyze the challenges. First the

method used for the analysis is defined, and then the process used to execute the analysis is described.

Scrum Coach	Open coding	Active coding	Selective coding
Scrum Coach D	*"I have often seen programmers poorly speaking of agile because for instance they never lost as much time in meetings than since we started that agile thing. If team members have this general opinion, they are likely to be right. Either the meetings are too frequent or too long, or they are not adding the expected value."*	Meaningful Scrum meetings	Scrum Meetings
Scrum Coach E	*When done well, retrospectives are often the most beneficial ceremony a team practices. When done poorly, retrospectives can be wasteful and a pain to attend. A retrospective whose action items are not acted on quickly becomes a meeting that people will not have respect for.*	Retrospectives without improvement	Scrum Meetings
Scrum Coach F	*These meetings can quickly turn messy if they are not managed effectively. They could also turn into*	Poorly executed Scrum Meetings	Scrum Meetings

	mentally and physically exhausting arguments without the moderation of a Scrum Master.		
Scrum Coach A	*An underpowered product owner lacks decision-making power. Not surprisingly, this caused delays and eroded the team's confidence in the product owner. Ensure that the product owner is fully empowered and receives support and trust from the right person."*	Underpowered product owner	Product Owner and Scrum Master related challenges
Scrum Coach E	*The selection of a new team's Scrum Master can impact the success or failure of the team's Scrum adoption. Choose the wrong person, and the team could face an uphill struggle. Choose the right person and the team will have an incredible head start in adopting Scrum.*	Choosing the right Scrum Master	Product Owner and Scrum Master related challenges
Scrum Coach D	*In both cases, being both product owner and scrum master is also likely to be a big draw on one person's time, leading to possible sacrifices in either their leading of*	One person being both the Scrum Master and Product owner	Product Owner and Scrum Master related challenges

	the team or in having up to date information on the product		
Scrum Coach B	*"This role is very important. Sometimes we get a great person, but often something is still wrong. Sometimes the person does not have enough decision-making authority, sometimes the person is not trained to be a Product Owner; sometimes the Product Owner is not interested in attending Sprint goals demonstration etc. At other times, the Product Owner may be focused on project deadlines, leading to a continuous process of scope negotiation just to meet such deadlines. That causes technical debt to increase continuously thereby making it more difficult to add features as the project evolves."*	Problems with the product owner	Product Owner and Scrum Master related challenges

Table 1: Grounded theory coding of Scrum challenges

4.3. Qualitative Results: List Of Challenges

The 21 challenges that resulted as a result of analysis using grounded theory are listed below. At this point in my thesis, this list is in no particular order. This list of challenges will be serve as input for quantitative section of my research where they will

be used to answer the research questions. A complete list of the interview transcripts with the six Scrum coaches can be found in Appendix B.

- Changing the organizational culture to support Scrum practices
- Breaking the waterfall mentality that everything needs to be sequentially completed
- Managing business expectations for deadlines
- Changing the management style from command and control to leadership and collaboration
- Lack of Agile compliant performance evaluation
- Lack of top-level executive support
- Challenges with distributed teams
- Lack of effective Scrum training
- Not understanding the values and principles behind Scrum
- Difficulty in creating empowered self-organizing teams and fostering the team mentality
- Resistance to change from team members
- Ineffective Product Owner or Scrum Master
- Fear in developers caused by skill deficiencies
- The need for developers to be a master of all trades
- Attempting to scale Scrum at the start
- Poorly executed Scrum meetings (Daily stand-up meetings, Retrospectives, Sprint
- Review, Sprint Planning etc.)
- Difficulty in adopting Engineering practices (Test Driven Development, Pair

- Programming, Continuous Integration, Refactoring etc.)
- Difficulty in grooming, estimating and managing the Product backlog
- Difficulty in determining sprint velocity
- Difficulty in delivering potentially shippable increments at the end of each sprint
- Agreeing on the definition of done

CHAPTER 5. QUANTITATIVE APPROACH: RESULTS AND DISCUSSION

5.1. Foreword

The quantitative research is the second part in the second part in the two-part analysis process in a sequential exploratory mixed methods design. The purpose of this mixed methods sequential exploratory study was to identify the key challenges that software teams face when they transition from a traditional software development methodology to Scrum. After gathering and analyzing responses for the challenges that software professionals' face when they transition from a traditional software development model to Scrum, this chapter analyzes the results by means of a quantitative survey. The aim of this chapter is to find answers to the five research questions.

This chapter is structured as follows: First the survey design and procedure is explained, followed by the statistical tests for quantitative data analysis. The chapter ends with a results and discussion section devoted to each of the twelve questions in the survey, which includes both the demographic questions as well as the research questions. Unless otherwise noted, all of the survey documents and data can be found in Appendix C.

5.2. Survey Design And Procedure

This subsection explains in each detail the step-by-step procedure involved in designing the questionnaire.

5.2.1. Tools For Gathering Data

There are many options to gather responses to a survey such as mailing or calling. However, one of the most cost effective and efficient method to gather data is by means of a web survey. A web survey is when a respondent is asked to visit a web site and respond to a survey questionnaire [38]. Since, the subjects in question have a high rate of Internet use, web surveys make the process fast and effective [39]. The advantages of a web survey include the flexibility of designing a questionnaire with logic and graphics where responses can be recorded directly into a database and are reported quickly [39]. The tool that will be used for the survey is SurveyMonkey.

5.2.2. Population Of Interest

The subjects for the survey questionnaire will be selected from industry representatives in the software industry. The survey is particularly aimed at professionals who have transitioned from a traditional waterfall software development process to Scrum. The participants were randomly selected from the LinkedIn group 'Scrum Practitioners' which has a membership of over fifty thousand. Three hundred Scrum practitioners were randomly selected from this entire population. They were individually contacted on LinkedIn and repeated appeals were made to them to complete the survey.

5.2.2.1. Determining The Sample Size

The goal of this section is to determine a target sample size value 'n' during the planning phase of the questionnaire design.

The equation to yield a representative sample for large sample proportions is [65]:

$$n_0 = \frac{Z^2 pq}{e^2}$$

Fig 3: Formula for calculating the sample size

Here,

n = sample size,

Z = Confidence Level. In this case Z= 1.96 for a 95% confidence interval),

e = Confidence Interval : A confidence interval is an indicator of your measurement's precision. It is also an indicator of how stable your estimate is, which is the measure of how close your measurement will be to the original estimate if you repeat your experiment. Here, e = 8

p = Degree of Variability. It is the estimated proportion of an attribute that is present in the population, and

With a population size of 48,000, confidence interval of 8 and a confidence level of 1.96, the target sample was obtained as n= 150 respondents.

5.2.2.2. Simple Random Sampling

Simple random sampling is a sampling method in which every eligible individual in the population has the same chance of being selected. In this method, a representative sample from the population provides the ability to generalize to a

population. Here, a list of eligible individuals is available and a random selection scheme is used to select a sample of individuals.

5.2.3. Human Subjects Committee And Consent Web Page

When a survey participant accesses the web survey, the first web page will be a page of consent. Respondents are informed on the first page that they are going to participate in an industry survey questionnaire about the challenges that software teams face when they transition from a traditional process to Scrum. The consent form and survey questionnaire can be found in Appendix C and were reviewed and approved by Dr. Steve Davis, Chairperson of the California Polytechnic State University - Human Subjects Committee. The page of consent included a description of the study, how long the questionnaire would take, how the responses would be kept confidential, how to contact the researchers and how to contact the Human Subjects Committee. The participant was then asked to answer the first question, which asked for his/her willingness to participate in the survey. If the participant consented by clicking yes, he/she was taken to the second page of the survey. However, if the participant indicated an unwillingness to participate, he/she was redirected to the exit page.

5.2.4. Survey Questionnaire

The industry survey questionnaire consists of 12 questions. The first seven questions are used to collect participant demographics. The seventh question also categorizes the survey participant as a Scrum coach with five or more years of

coaching experience or not. The different Scrum challenges collected by means of qualitative interviews are categorized into three main categories, namely organization and management related challenges, people related challenges and process related challenges. The survey participants are requested to rate each challenge based on their personal experience on a five-point Likert scale, with options varying from very frequently, frequently, occasionally, rarely and never. Likert scale responses are used to provide a sense of how the respondent feels about a particular statement or question, and the strength of that feeling [38]. The participants are then finally asked to rank the top 5 challenges from 1 to 5 with rank 1 corresponding to the highest challenge and in that order.

5.2.5. Data Observation And Tests For Validity

Statistics is the science of collecting, analyzing, presenting and interpreting data and hypothesis tests are a form of statistical inference that uses data from a sample to draw conclusions about a population parameter [48].

This section will provide a brief background on the Spearman's rank correlation coefficient which is used for analyzing the results obtained from the survey.

5.2.5.1. Spearman's Rank Correlation Coefficient

Correlation is a statistical technique used to find if two variables are related to each other. Spearman's rank correlation coefficient is a non-parametric measure of correlation, using ranks to calculate the correlation. If a change in one variable brings about a change in the other, they are said to be correlated[18]. To calculate the

Spearman's coefficient, all the data should be in terms of ranks. It tries to assess the relationship between ranks without making any assumptions about the nature of their relationship. Hence, it is a non – parametric measure, a feature which has contributed to its popularity and widespread use. It is denoted by rs.[18]

Correlation Coefficient: The numerical value of the correlation coefficient, r_s, ranges between -1 and +1. The correlation coefficient is the number indicating how the scores are related to each other.

$$r_s = 1 - \frac{6\sum d^2}{n(n^2 - 1)}$$

Fig 4: Formula For Calculating Spearman's Coefficient

where d = difference in paired ranks and *n* = number of cases.

Interpretation: In general,

$r_s > 0$ implies positive agreement among ranks

$r_s < 0$ implies negative agreement among ranks

$r_s = 0$ implies no agreement among ranks

The closer r_s is to 1, better the agreement while the closer that r_s is to -1, the stronger the agreement in the reverse direction.

Also, the below table provides a rule of thumb for interpreting the size of the correlation coefficient.

Size of correlation	Interpretation
.90 to 1.00 (-.90 to -1.00)	Very high positive(negative) correlation
.70 to .90 (-.70 to -.90)	High positive(negative) correlation
.50 to .70 (-.50 to -.70)	Moderate positive(negative) correlation
.30 to .50 (-.30 to -.50)	Low positive(negative) correlation
.00 to .30 (-.00 to -.30)	Negligible correlation

Table 2: Interpreting Spearman's Rank Correlation Coefficient

5.2.5.2. P-value

The P-value answers this question: If there really is no correlation between X and Y overall, what is the chance that random sampling would result in a correlation coefficient as far from zero or further as observed in this experiment? [18]

If the P-value is small, then the idea that the correlation is due to random sampling can be rejected. If the P-value is large, the data do not give any reason to conclude that the correlation is real. This is not the same as saying that there is no correlation at all. This just means that there is no compelling evidence that the correlation is real and not due to chance.[18]

5.3. Pilot Study

A pilot study can be defined as a small study to test research protocols, data collection instruments, sample recruitment strategies and other research techniques in preparation for a larger study. A pilot study is conducted to identify potential problem

areas and deficiencies in the research instruments prior to implementation during the full study [36].

For these reasons, a pilot study was conducted on a group of 12 participants before conducting the full study. Results from this study were used strictly for feedback and improvement and were not used for data analysis. Some survey questions were eliminated and others redesigned based on the feedback from this pilot study. For example, the participants of the pilot study reported that it was a time consuming process to rank all twenty one challenges in terms of their importance and that given the choice, they would rather skip this question. So this question was redesigned and the survey participants only had to rank the top five challenges and not the top twenty one.

5.4. The Survey

The survey was administered online and accessed through the URL. Potential participants were randomly selected from a LinkedIn group of 'Scrum Practitioners'. The group had 48,000 members at the time. Of this 300 participants were short-listed using random selection. These participants were then individually contacted and requested to complete the survey. The procedure was complicated because of the multiple follow-ups required to ensure a good response rate. In the first week, only 120 participants responded. Multiple reminders requesting survey participation were sent to survey participants. The data collection took place between Feb 21 and March 9, 2015.

5.5. Response Rate

Three hundred randomly selected Scrum Practitioners were contacted to participate in the survey. Each participant was contacted four times over a time span of three weeks and requested to complete the survey. At the end of three weeks, 210 responses to the survey were received. This constitutes a response rate of seventy percent. After reviewing and filtering the recorded responses, 202 were considered valid and were considered for analysis.

5.6. Demographics

Demographic based questions were asked to help understand the diversity of the subjects. The participants were asked the number of years they had been involved with software development, the size of their organization, current position, whether or not they had experienced a Scrum transition and whether or not they have had Scrum coaching experience for five or more years.

5.6.1. Software Development Industry Experience

Of the valid responses, a total of 178 participants answered this question. My goal was to include a wide distribution of respondents with various years of industry experience since an entry-level employee may have different insights from a more experienced employee. This goal was met, as there is a wide range in the years of experience of survey participants from just a few years to over twenty years of software development experience. The median number of years of industry experience was 9 and the mean number of years of industry experience was 8.

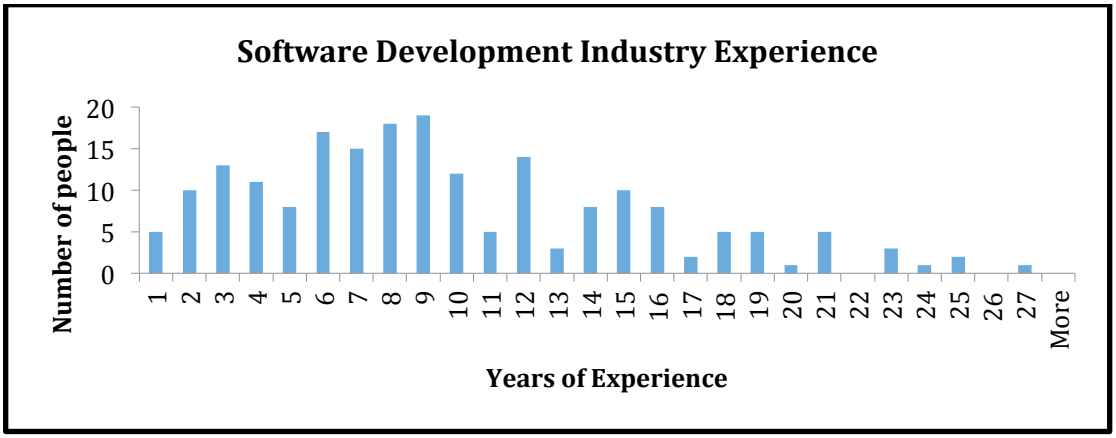

Fig 5: Survey responses to software development industry experience

5.6.2. Size Of The Organization

The size of an organization is a major factor in determining the challenges faced during a major organizational change such as the adoption of Scrum. So, this question categorized survey responses into three categories, which are small sized organizations with less than one hundred employees, medium sized organizations with between one hundred to one thousand employees and large organizations with more than one thousand employees. An overwhelming majority of the survey respondents were from large sized organizations,which fits well with my research goals.

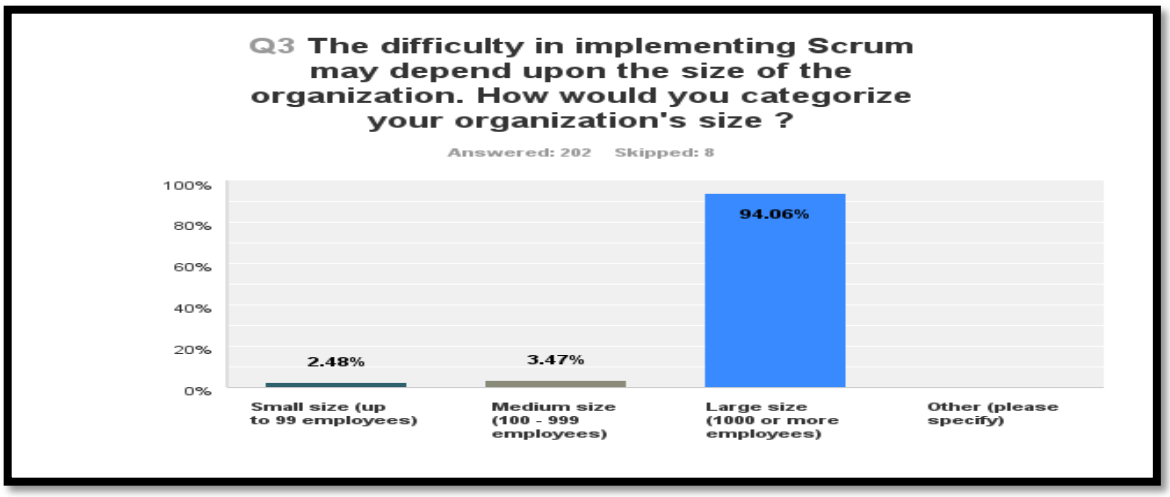

Fig 6: Survey responses to size of the organization

Three hundred randomly selected Scrum Practitioners were contacted to participate in the survey. Each participant was contacted four times over a time span of three weeks and requested to complete the survey. At the end of three weeks, 210 responses to the survey were received. This constitutes a response rate of seventy percent. After reviewing and filtering the recorded responses, 202 were considered valid and were considered for analysis.

5.6.3. Current Position

Of the valid responses, a total of 201 responses indicated the individual's position at the company. Of these, 150 were development team members, 14 were Product Owners, 14 were Scrum Masters and 19 were Scrum coaches. This constitutes 75 percentage, 7 percentage, 7 percentage and 10 percentage respectively of the total survey participants. My goal to receive diverse responses so as to analyze the data in separate ways was met.

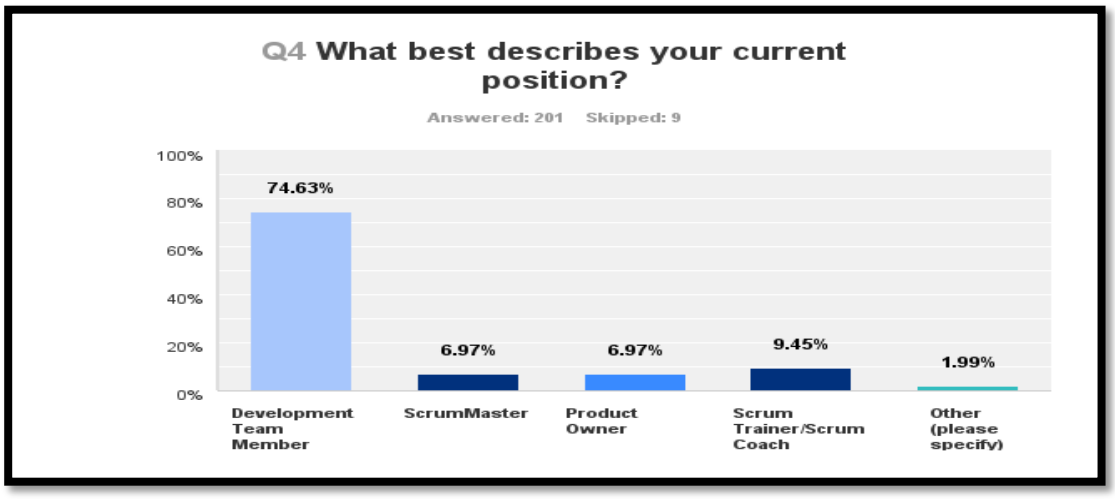

Fig 7: Survey responses to current position

5.6.4. Principal Industry

Of the valid responses, a total of 82 responses indicated the principal industry of the organization. An overwhelming majority of the respondents were from the information technology industry. Other industries represented were advertising and marketing, aerospace, automotive, education, financial services, government, healthcare and pharmaceuticals, manufacturing, non-profit, retail, telecommunications and insurance.

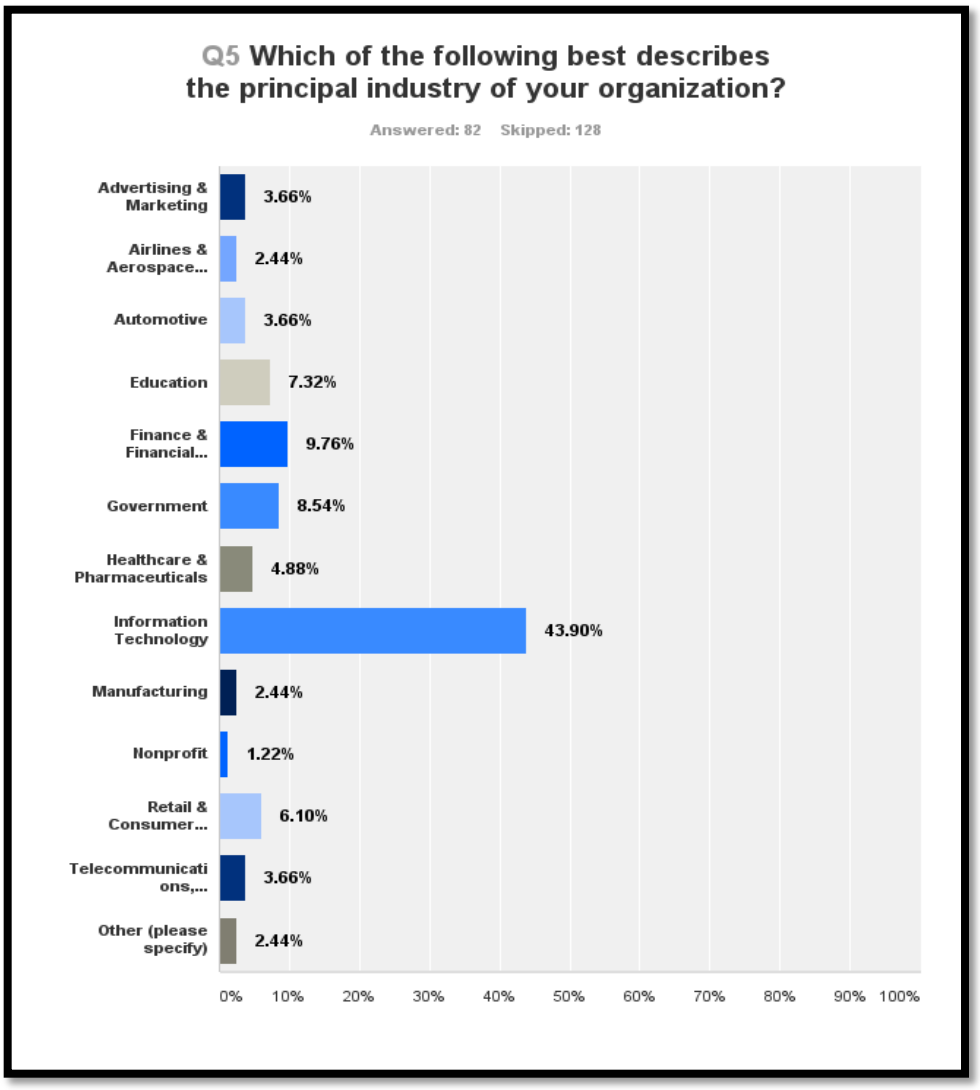

Fig 8: Survey Responses To Principal Industry Of Your Organization

5.6.5. Scrum Transition Experience

This question was asked to make sure that the respondents to the survey had experienced a Scrum transition. Of the 202 respondents to this question, 186 answered positively.

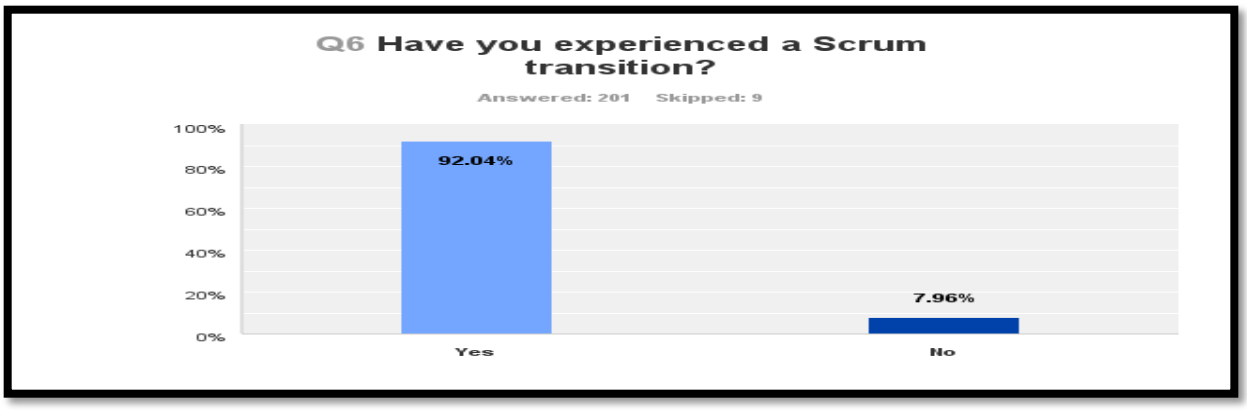

Fig 9: Survey responses to Scrum transition experience

5.6.6. Scrum Coaching Experience

In the qualitative research section of the thesis, all the interviewees had more than five years of Scrum coaching/training experience. So, the purpose of this question was to identify the Scrum Coaches who had at least five years of Scrum coaching/training experience. A total of 30 respondents answered positively to this question. This data can be used to correlate the findings from both the qualitative and the quantitative studies.

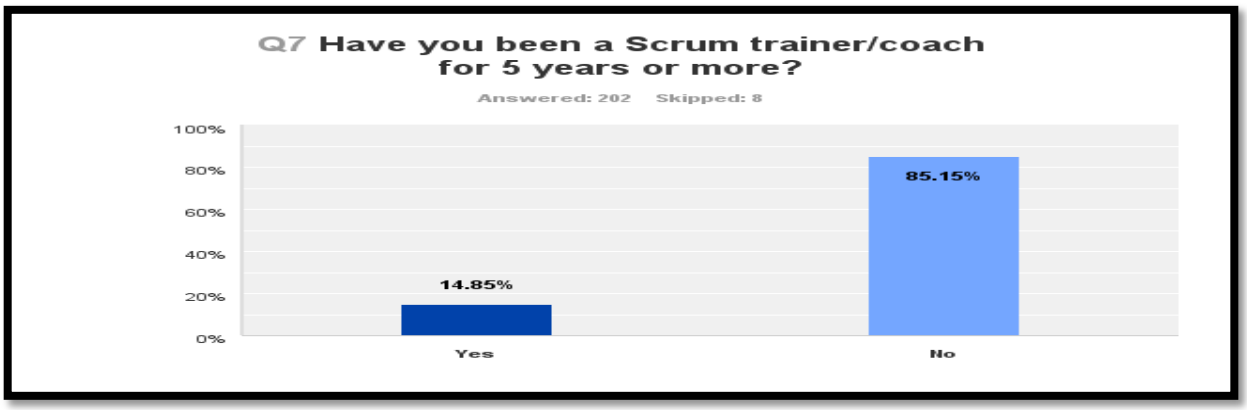

Fig 10: Survey responses to Scrum coaching experience

5.7. Research Question 1: *What Are The Most Frequent Challenges?*

The subsection examines results related to research question 1: *What are the most frequent challenges that software development teams face during a transition from a traditional waterfall software development model to Scrum in large organizations?*

5.7.1. Introduction

Frequency as the rate at which something occurs or it is repeated over a particular period of time in a given sample. So, equating this to the case at hand, the frequency of occurrences of a challenge refers to the number of occurrences of the challenge. This section will follow the following structure: First the question is introduced, then the method of analysis is discussed followed by results and ends with a discussion of the results.

5.7.2. Methodology

The survey Participants were asked to rate the twenty-one challenges identified by means of qualitative analysis using the Likert scale. The participants were asked to choose a rating for each challenge on a Likert Scale based on their personal experience. The scale had values ranging from very frequently, frequently, occasionally, rarely and never. The weighted average for each challenge would then be calculated based on the rating that each challenge received on the Likert Scale. A lower weighted average represented a higher rank and vice versa (The highest rank was 1). The standard deviation was also calculated which provides a measure of the extent of deviation for the group as a whole or how spread out the numbers are.

5.7.3. Results

A total of 170 responses in total were received for this question. Table 2 shows the number of responses received for each challenge in each category. The weighted average was calculated for each of these challenges by assigning a weight of 1 to very frequently, 2 to frequently, 3 to occasionally, 4 to rarely and 5 to never. A lower weighted average thus pointed to a more frequently occurring challenge. The standard deviation provides a measure of the extent of deviation for the group as a whole or how spread out the numbers are:

SCRUM TRANSITION CHALLENGES (Frequency)	VERY FREQ (1)	FREQUENTLY (2)	OCCASIONALLY (3)	RARELY (4)	NEVER (5)	AVG RANK	STD DEV
Changing the organizational culture to support Scrum practices	93	71	6	0	0	1.49	44.57
Difficulty in determining sprint velocity	96	52	19	1	0	1.55	40.73
Difficulty in adopting Engineering Practices	90	64	11	2	1	1.57	40.85
Breaking the waterfall mentality that everything needs to be sequentially completed	77	85	7	0	0	1.59	43.27
Poorly executed Scrum meetings	83	74	8	3	0	1.59	41.21
Difficulty in delivering "potentially shippable increments" at the end of each sprint	85	67	11	3	0	1.59	39.78
Changing the management style from "command and control" to "leadership and collaboration"	68	67	26	7	0	1.83	32.37
Difficulty in creating empowered self-organized teams and fostering team mentality	62	81	20	3	3	1.84	35.74
Challenges with distributed teams	49	87	20	8	4	1.99	34.65
Fear in developers caused by skill deficiencies	44	90	25	8	1	2	35.64
The need for developers to be a master of all trades	41	90	30	5	1	2.01	35.81
Not understanding the values and	41	90	28	9	1	2.05	35.12

principles behind Scrum							
Lack of effective Scrum Training	55	67	30	16	1	2.06	27.19
Managing business expectations for deadlines	42	78	38	9	1	2.1	30.53
Resistance to change from team members	46	74	34	12	2	2.11	28.51
Lack of agile compliant performance evaluation	49	68	37	11	4	2.13	26.54
Difficulty in grooming, estimating and managing the Product backlog	31	79	47	10	0	2.22	31.35
Ineffective Product Owner or Scrum Master	31	66	48	19	3	2.38	24.56
Attempting to scale Scrum at the start	13	48	69	25	12	2.85	24.62
Agreeing on the "Definition of Done"	18	33	51	42	24	3.13	13.31
Lack of top level executive support	10	22	34	57	46	3.63	18.66

Table 3: Scrum Transition Challenges Ordered In Terms Of Frequency

The histogram below in Figure 5 provides a visual interpretation of these challenges sorted by the most frequently occurring to the least frequently occurring challenge.

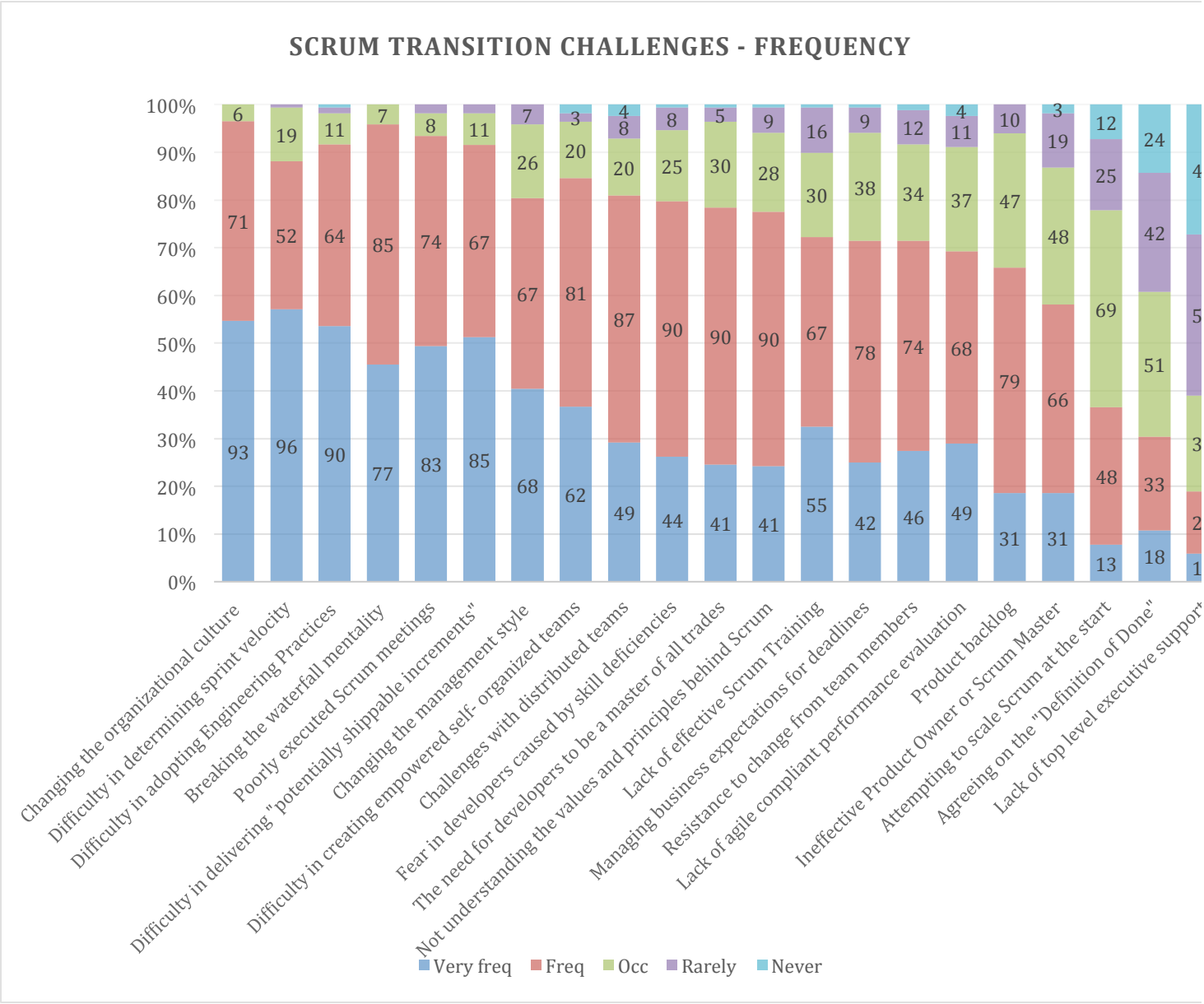

Fig 11: Scrum Transition Challenges Ordered In Terms Of Frequency Of Occurrence

5.7.4. Discussion

There are several interesting observations that are evident from the results obtained. The top six overall most frequent challenges had a high number of respondents who rated them as either “very frequently” or “frequently”. At the same time, they had a very low number of respondents who rated these challenges as either occasionally, rarely or never. So, these six challenges namely Changing the organizational culture to support Scrum practices, Difficulty in determining sprint velocity, Difficulty in adopting Engineering Practices, Breaking the waterfall mentality that everything needs to be sequentially completed, Poorly executed Scrum meetings and Difficulty in delivering "potentially shippable increments" at the end of each sprint have close to unanimous agreement with regards to their position at the top. Thus, companies undergoing a Scrum transition must be aware of these challenges and take counter measures to minimize their effect.

The three least frequent challenges namely Attempting to scale Scrum at the start, Agreeing on the definition of done and Lack of top-level executive support were given a rating of occasionally, rarely or never by a majority of participants. That means these were very infrequent.

Another interesting observation is that the statistical deviation, which provides an indication of how spread out the numbers are is the maximum for the more frequently occurring challenges and it gradually decreases until it reaches the minimum for the least frequently occurring challenge. What can be inferred from this is that the survey participants are in general agreement about the less frequently occurring challenges, but they have disagreements regarding the most frequent challenges.

5.8. Research Question 2: *What Are The Most Important Challenges?*

The second research question is*: What are the most important challenges that software development teams in large organizations face during a transition from a traditional Waterfall Software Development model to Scrum?*

5.8.1. Introduction

The importance of a challenge refers to the magnitude of the size of its effect. The aim of this subsection is to rank the Scrum transition challenges in terms of its importance and will follow the following structure: First the question is introduced, then the method of analysis is discussed followed by results and ends with a discussion of the results.

5.8.2. Methodology

The survey Participants were asked to pick the top five most important challenges from the list of the twenty-one challenges and assign them ranks from one to five based on their importance. The participants were asked to do so based on their personal experience during a Scrum transition. The weighted average for each challenge is then calculated based on this ranking. A lower weighted average represented a higher rank and vice versa. The standard deviation is then calculated which provides a measure of the extent of deviation for the group as a whole or how spread out the numbers are.

SCRUM TRANSITION CHALLENGES (Importance)	RANK 1	RANK 2	RANK 3	RANK 4	RANK 5	NOT RNK'D	AVG RANK	STD DEV
Determining sprint velocity	28	21	22	17	12	70	2.47	5.96
Difficulty in delivering "potentially shippable increments" at the end of each sprint	8	15	25	28	21	73	2.58	8.02
Poorly executed Scrum meetings	10	6	31	26	23	74	2.61	10.71
Difficulty in adopting Engineering practices	9	23	13	13	26	86	3.04	7.29
Changing the organizational culture to support Scrum practices	37	8	5	9	11	100	3.53	13.04
The need for developers to be a "master of all trades"	10	15	13	16	13	103	3.64	2.30
Challenges with distributed teams	8	13	5	9	9	126	4.45	2.86
Difficulty in creating empowered self-organizing teams and fostering the team mentality	5	2	8	8	7	140	4.94	2.55
Breaking the waterfall mentality that everything needs to be sequentially completed	9	13	2	0	5	141	4.98	5.26
Lack of effective Scrum training	10	6	3	2	8	141	4.98	3.35
Lack of agile compliant performance evaluation	8	6	6	1	6	143	5.05	2.61
Fear in developers caused by skill deficiencies	5	5	5	10	1	144	5.08	3.19

Managing business expectations for deadlines	4	7	5	3	6	145	5.12	1.58
Changing the leadership style from "command and control" to "leadership and collaboration"	7	7	6	5	0	145	5.12	2.92
Not understanding the values and principles behind Scrum	2	7	4	5	3	149	5.26	1.92
Ineffective Product Owner or Scrum Master	0	4	4	4	5	153	5.40	1.95
Resistance to change from team members	4	2	2	3	4	155	5.47	1.00
Lack of top level executive support	3	3	1	0	2	161	5.68	1.30
Attempting to scale Scrum at the start	0	2	3	1	2	162	5.72	1.14
Difficulty in grooming, estimating and managing the Product Backlog	2	3	6	8	4	147	5.19	2.41
Agreeing on the "Definition of Done"	1	2	0	2	1	164	5.79	0.84

Table 4: Scrum Transition Challenges Ordered In Terms Of Importance

5.8.3. Results

A total of 170 responses in total were received for this question. Table 3 shows the number of responses received for each challenge in each category. The weighted average was calculated for each of these challenges by assigning a weight of 1 to rank-1, 2 to rank-2, 3 to rank-3, 4 rank-4 and 5 to rank-5. Based on this ranking scale, the survey respondents only ranked five challenges from a list of twenty-one which according to them were the most important based on their Scrum transition experience. For the

purpose of calculating the weighted average, the remaining seventeen unranked challenges by each respondent were assigned a weight of six. The weighted average was then calculated for each challenge.

A lower weighted average thus pointed out to a more important challenge. The standard deviation for each challenge was also calculated and it provides a measure of the extent of deviation for the group as a whole or how spread out the numbers are.

The histogram below in Figure 6 provides a visual interpretation of these challenges sorted by the most frequently occurring to the least frequently occurring challenge.

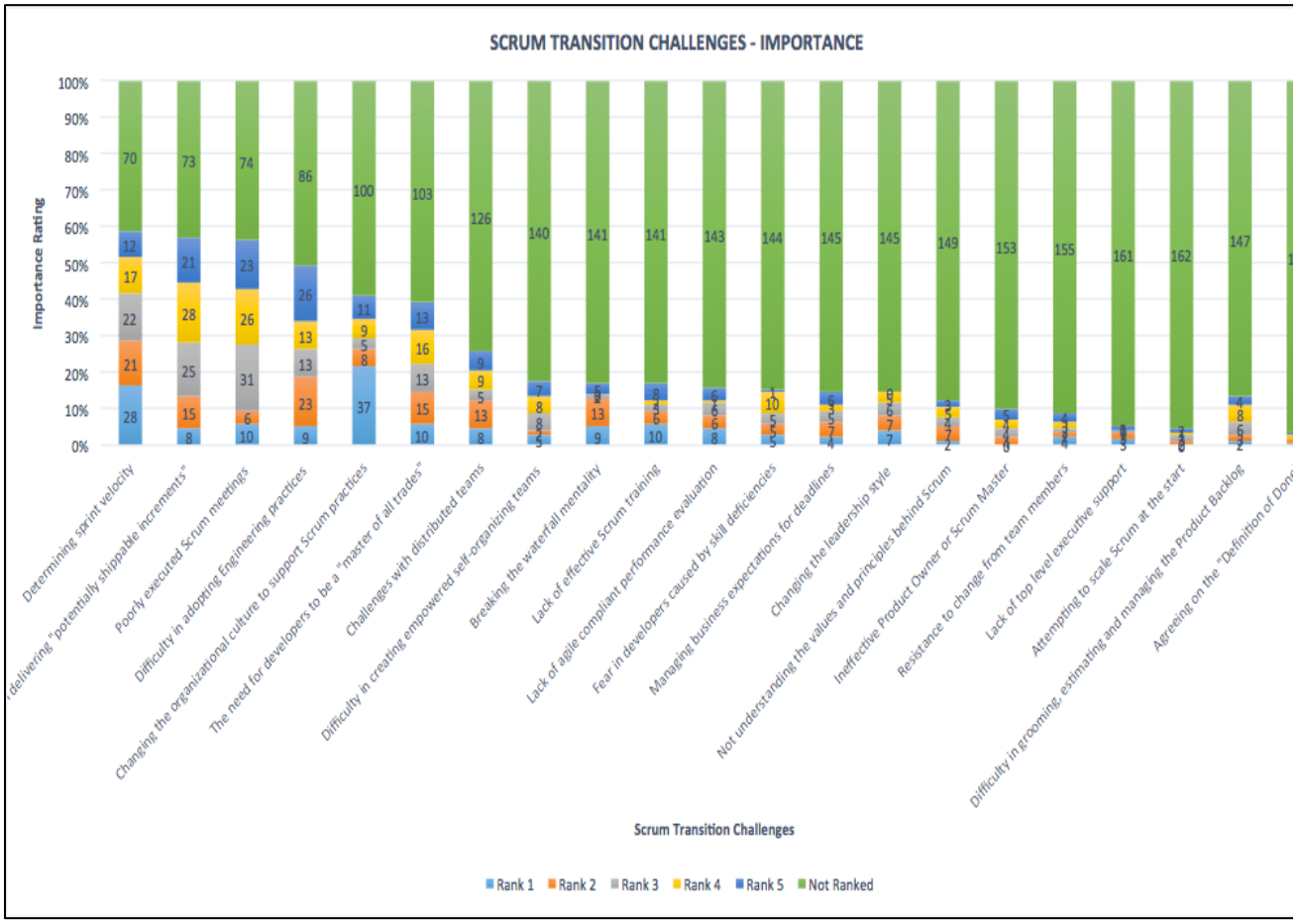

Fig 12: Ranking Of Scrum Challenges By Importance

5.8.4. Discussion

Based on the weighted average, the top six most important challenges that are encountered when transitioning from a traditional Waterfall model to Scrum are Determining sprint velocity, Difficulty in delivering "potentially shippable increments" at the end of each sprint, Poorly executed Scrum meetings, Difficulty in adopting Engineering practices, Changing the organizational culture to support Scrum practices and The need for developers to be a "master of all trades". Thus, companies undergoing a Scrum transition must be aware of these challenges and take counter measures to minimize their effect.

Based on the weighted average, the four least important challenges are Agreeing on the "Definition of Done", Difficulty in grooming, estimating and managing the Product Backlog, Attempting to scale Scrum at the start and Lack of top level executive support. An interesting observation is that the three least frequent challenges identified in the previous subsection are also in the list of the four least important challenges. Thus it would be fair to say that, of these twenty one challenges, these three challenges which are Agreeing on the "Definition of Done", Attempting to scale Scrum at the start and Lack of top level executive support are the least important as well as the least frequent.

The standard deviation for each challenge was also calculated and it provides a measure of the extent of deviation for the group as a whole or how spread out the numbers are. The least important challenge "Defining the definition of done" has the lowest standard deviation. This means that there was a very high agreement among survey participants that this was the least important challenge. Standard deviation was comparatively high for some challenges such as Poorly executed Scrum meetings with a

deviation of 8.02, Changing the organizational culture to support Scrum practices with a deviation of 10.71 and Changing the organizational culture to support Scrum practices with a deviation of 13.04. This indicates that survey participants disagreed about the ranks given to these participants. This could also mean that some respondents ranked these highly and some did not rank it at all which leads to the understanding some people are highly affected by this challenge while others are not.

5.9. Research Question 3: *Is There Direct Correlation Between Frequency And Importance Of Challenges?*

5.9.1. Introduction

This subsection will analyze results from research question 1and research question 2 to find if the Scrum transition challenges are correlated in terms of their frequency and their importance. In the following subsections, the method of analysis is discussed followed by the results and finally ends with a discussion of the results.

5.9.2. Methodology

To find an answer to this question, each Scrum transition challenge was given a rank based on its frequency of occurrence and in terms of its importance. A lower weighted average/average rank corresponded to a higher rank and vice versa. The Spearman's rank correlation coefficient and p-value is computed from the ranks given to the frequency and importance of the challenges. These values will give an indication of the correlation between frequency and importance.

5.9.3. Results

Each Scrum transition challenge was given a rank based on its frequency of occurrence. This was obtained by means of its weighted average calculated in Table 3. A lower weighted average/average rank corresponded to a higher rank and vice versa. In this way, the most frequent challenge was given a rank of one and the least frequent challenge was given a rank of twenty-one. In a similar way, each Scrum transition challenge was given a rank based on its importance. This was obtained by means of its weighted average calculated in table 4.

SCRUM TRANSITION CHALLENGES	FREQ (WA)	IMP (WA)	RANK-FREQ	RANK - IMP	AVG RANK	FINAL RANK
Determining sprint velocity	1.55	4.02	2	1	1.5	1
Changing the organizational culture to support Scrum practices	1.49	4.46	1	2	1.5	2
Poorly executed Scrum meetings (Daily stand-up meetings, Retrospectives, Sprint Review meeting, Sprint Planning Meeting)	1.56	4.58	3	4	3.5	3
Breaking the waterfall mentality that everything needs to be sequentially completed	1.58	5.36	4	8	6	4
Challenges with distributed teams	1.99	5.21	9	7	8	5
The need for developers to be a "master of all trades"	2.01	4.86	11	6	8.5	6
Changing the leadership style from "command and control" to "leadership and collaboration"	1.83	5.46	7	10	8.5	7
Difficulty in creating empowered self organizing teams and fostering the team mentality	1.84	5.53	8	13	10.5	8

Difficulty in delivering "potentially shippable increments" at the end of each sprint	2.85	4.52	19	3	11	9
Difficulty in adopting Engineering practices (Test Driven Development, Pair Programming, Continuous Integration etc.)	2.22	4.66	17	5	11	10
Lack of effective Scrum training	2.06	5.44	13	9	11	11
Fear in developers caused by skill deficiencies	2.00	5.52	10	12	11	12
Difficulty in grooming, estimating and managing the Product Backlog	1.59	5.65	6	16	11	13
Attempting to scale Scrum at the start	1.59	5.88	5	20	12.5	14
Lack of agile compliant performance evaluation	2.13	5.47	16	11	13.5	15
Not understanding the values and principles behind Scrum	2.05	5.63	12	15	13.5	16
Managing business expectations for deadlines	2.10	5.56	14	14	14	17
Resistance to change from team members	2.11	5.74	15	17	16	18
Ineffective Product Owner or Scrum Master	2.38	5.76	18	18	18	19
Lack of top level executive support	3.64	5.81	21	19	20	20
Agreeing on the "Definition of Done"	3.13	5.89	20	21	20.5	21

Table 5: Scrum Transition Challenges – Frequency V's Importance

A lower weighted average/average rank corresponded to a higher rank and vice versa. In this way, the most important challenge was given a rank of one and the least frequent challenge was given a rank of twenty-one.

An average rank was also calculated for each challenge by calculating the mean of these two ranks. This gave way to a final ranking of challenges taking the criteria of both frequency and importance into consideration. Thee results are shown in table 5.

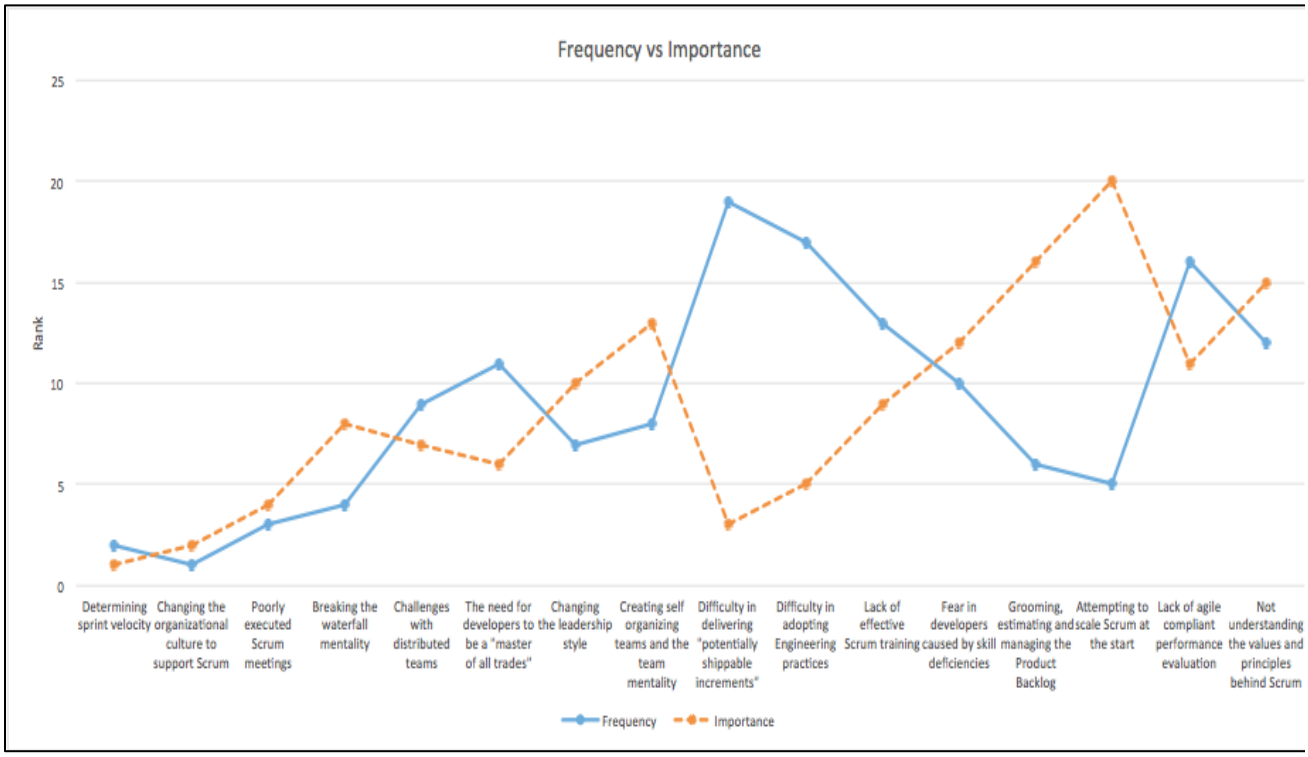

Fig 13: Frequency V's Importance

The line-graph above in Fig 13 provides a visual interpretation of the alignment between frequency and importance.

To calculate the correlation between frequency of occurrence of each challenge and importance of each challenge, the spearman's rank correlation coefficient was calculated.

Statistical measurement	Value
Spearman's rank correlation coefficient	0.4321
P-value	0.0487

Table 6: Statistical Measurements For R.Q. 3.

5.9.4. Discussion

The Spearman's rank correlation coefficient has values extending from -1 to 0 to +1 representing perfect negative correlation with +1, no correlation with 0 and perfect positive correlation with +1. The Spearman's rank correlation coefficient of 0.4321 indicates a positive medium correlation. The p-value needs to be interpreted as the probability of seeing the observed correlation if no correlation exists. Since, the p-value is very low (0.0487), we can conclude that it is very improbable that the observed correlation comes from a random effect. In other words, the observed correlation reliably represents actual correlation.

So, the most frequent challenges have a medium probability of also being the most important.

As observed from Fig13, the four challenges, which have the least correlation between frequency and importance, are Difficulty in delivering " potentially shippable increments" at the end of each sprint, Difficulty in adopting Engineering practices, Difficulty in grooming, estimating and managing the product backlog and attempting to scale at the start.

While Difficulty in delivering " potentially shippable increments" at the end of each sprint and Difficulty in adopting Engineering practices had a high importance ranking, they did not a have a high frequency ranking. This means that though these challenges were important, they were not very frequent. This leads to the conclusion that though the survey respondents agreed that these were important challenges, not many had to frequently face these challenges.

While Difficulty in grooming, estimating and managing the product backlog and attempting to scale at the start had a high frequency ranking, they did not a have a high importance ranking. This means that though these challenges were frequent, they were not very important. This leads to the conclusion that though the survey respondents agreed that these were not important challenges, they were frequently faced with these.

5.10. Research Question 4: *Regarding Frequency Of Challenges, Do Coaches Agree With Practitioners?*

The fourth research question is: In regards to the most frequent challenges faced during a Scrum transition, are the views of Scrum coaches and general Scrum practitioners in alignment? In the following subsections, the method of analysis is discussed followed by the results and finally ends with a discussion of the results.

5.10.1. Methodology

To find an answer to this research question, both Scrum coaches as well as Scrum practitioners who took the survey were asked to rate the Scrum transition challenge based on its frequency on the Likert scale. The scale had values ranging from

very frequently, frequently, occasionally, rarely and never. The weighted average for each challenge would then be calculated based on the rating that each challenge received on the Likert Scale. A lower weighted average represented a higher rank and vice versa (The highest rank was 1). The weighted average and the corresponding ranks are tabulated in table 7.

The Spearman's rank correlation coefficient and p-value is computed from the ranks given to the frequency of occurrence of each challenge by the Scrum Coach and the Scrum practitioner. These values give an indication of the correlation between frequency and importance.

5.10.2. Results

A total of 170 responses in total were received for this question. The respondents were categorized into two classes based on whether they were an experienced Scrum Coach or not. A Scrum Coach was categorized as experienced if he/she had at least five years of Scrum coaching experience. A total of twenty-eight experienced Scrum coaches answered this question. The table below lists weighted average of each challenge in terms of its frequency of occurrence as ranked by the 28 Scrum Coaches who took the survey and the 142 Scrum practitioners who took the survey. The weighted average was calculated for each of these challenges by assigning a weight of 1 to very frequently, 2 to frequently, 3 to occasionally, 4 to rarely and 5 to never. A lower weighted average thus pointed to a more frequently occurring challenge. Then each challenge is given a rank from one to twenty one in terms this average rank with one being the highest and twenty one the lowest.

SCRUM TRANSITION CHALLENGES	WGTD-AVG COACHES	WGTD-AVG PRACTITIONER	RANKING BY PRACTITIONER	RANKING BY COACHES
Difficulty in determining sprint velocity	1.86	1.49	1	10
Changing the organizational culture to support Scrum practices	1.14	1.56	2	1
Difficulty in delivering "potentially shippable increments" at the end of each sprint	1.64	1.58	3	6
Difficulty in adopting Engineering Practices	1.50	1.59	4	4
Poorly executed Scrum meetings (Daily stand-up meetings, Retrospectives, Sprint Review, Sprint Planning)	1.50	1.61	5	3
Breaking the waterfall mentality that everything needs to be sequentially completed	1.36	1.63	6	2
Changing the management style from "command and control" to "leadership and collaboration"	1.71	1.86	7	7
Difficulty in creating empowered self-organized teams and fostering team mentality	1.54	1.90	8	5
Fear in developers caused by skill deficiencies	2.00	2.00	9	13
The need for developers to be a master of all trades	2.07	2.00	10	14

Challenges with distributed teams	1.79	2.04	11	8
Managing business expectations for deadlines	2.21	2.08	12	17
Lack of effective Scrum training	1.93	2.09	13	11
Not understanding the values and principles behind Scrum	1.86	2.09	14	9
Lack of agile compliant performance evaluation	2.11	2.13	15	15
Resistance to change from team members	1.93	2.14	16	12
Difficulty in grooming, estimating and managing the Product backlog	2.18	2.22	17	16
Ineffective Product Owner or Scrum Master	2.36	2.39	18	18
Attempting to scale Scrum at the start	2.61	2.90	19	20
Agreeing on the "Definition of Done"	2.54	3.24	20	19
Lack of top level executive support	3.32	3.70	21	21

Table 7: Scrum Transition Challenges (Frequency) – Coaches V's Practitioners

To calculate the correlation between the views of Scrum coaches and general Scrum practitioners regarding the frequency of occurrence of Scrum transition challenges, the spearman's rank correlation coefficient was calculated.

Statistical Measurement	Value
Spearman's rank correlation coefficient	0.8481
P-value	0.0001

Table 8: Statistical Measurements For R.Q. 4.

The line-graph below in Fig 7 provides a visual interpretation of the alignment between frequency and importance.

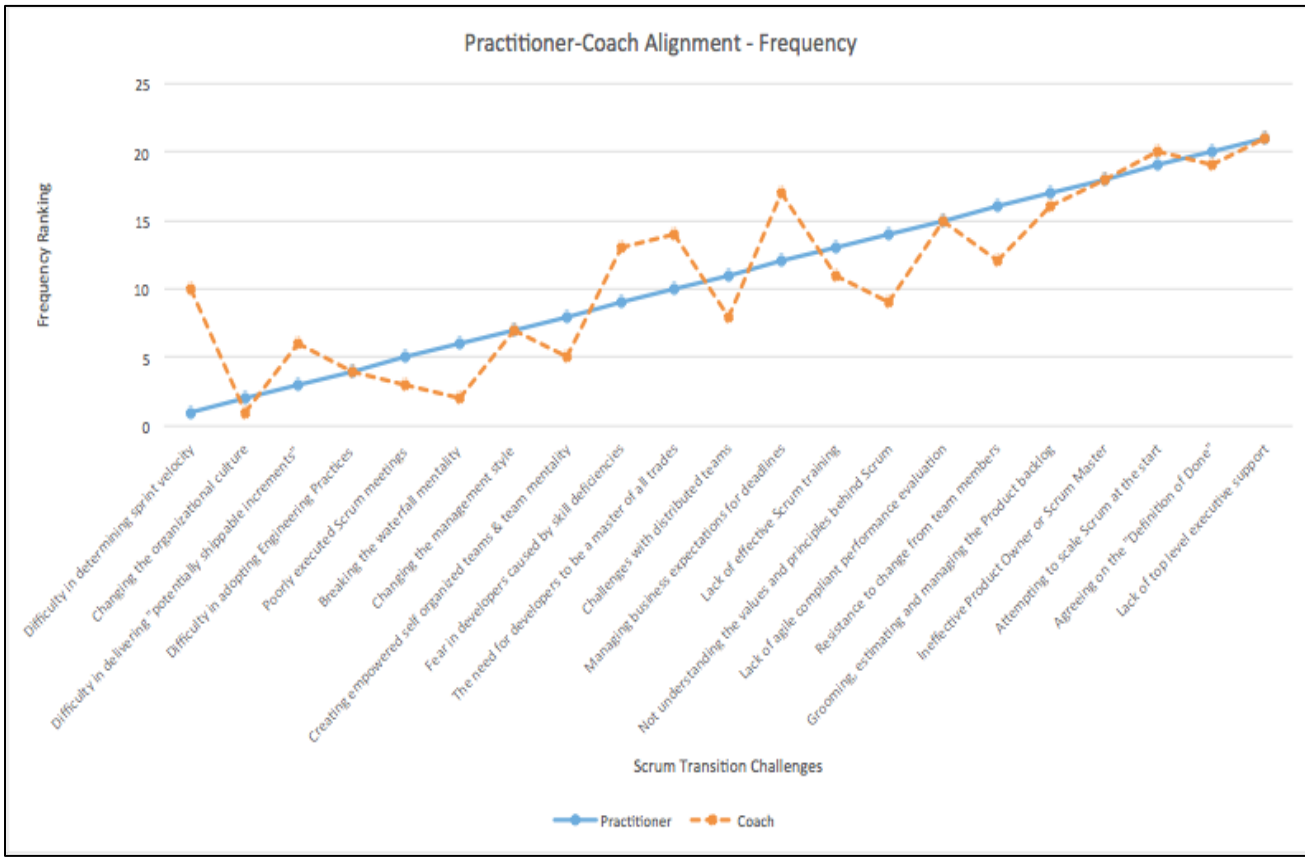

Fig 14: Practitioner V's Coaches – Alignment In Frequency

5.10.3. Discussion

The Spearman's rank correlation coefficient has values extending from -1 to 0 to +1 representing perfect negative correlation with +1, no correlation with 0 and perfect positive correlation with +1. The Spearman's rank correlation coefficient of 0.8481 indicates a high positive correlation. The p-value needs to be interpreted as the probability of seeing the observed correlation if no correlation exists. Since, the p-value is very low (0.0001), we can conclude that it is very improbable that the observed correlation comes from a random effect. In other words, the observed correlation reliably represents actual correlation.

So, to answer the research question, regarding the most frequent challenges, the views of Scrum coaches and Scrum practitioners have a high positive correlation.
One key result of answering this research question was with regards to the Scrum transition challenge " Determining Sprint Velocity". There was a high disagreement between Scrum coaches and Scrum practitioners in their responses to this challenge. Scrum practitioners ranked sprint velocity as their most frequently occurring challenge, while Scrum coaches only gave it a rank of ten. So, Scrum coaches should take note of this opinion put forward by Scrum practitioners and find solutions to the problem of "Determining Sprint Velocity" so as to facilitate smoother Scrum transitions.

According to responses collected from the Scrum Coaches who took the survey, the five most important challenges are:

- Changing the organizational culture to support Scrum practices
- Difficulty in delivering potentially shippable increments at the end of each sprint
- Breaking the waterfall mentality that everything needs to be sequentially executed

- Poorly executed Scrum meetings, which includes the Daily stand- up meetings, Retrospectives, Sprint Review meetings and the Sprint Planning Meeting.
- The need for developers to be a master of all trades

According to responses collected from the Scrum Practitioners who took the survey, the five most important challenges are:

- Determining the sprint velocity
- Poorly executed Scrum meetings, which include the Daily stand- up meetings, Retrospectives, Sprint Review meetings and the Sprint Planning Meeting.
- Difficulty in delivering potentially shippable increments at the end of each sprint
- Difficulty in adopting Engineering practices which include Test Driven Development, Pair Programming, Continuous Integration etc.)
- The need for developers to be a master of all trades.

5.11. Research Question 5: *Regarding Importance Of Challenges, Do Coaches Agree With Practitioners?*

The fifth and final research question is: In regards to the most important challenges faced during a Scrum transition, are the views of Scrum coaches and general Scrum practitioners in alignment? In the following subsections, the method of analysis is discussed followed by the results and finally ends with a discussion of the results.

5.11.1. Methodology

To find an answer to this research question, both Scrum coaches as well as Scrum practitioners who took the survey were asked to rate the Scrum transition

challenge based on its importance. The survey Participants were asked to pick the top five most important challenges from the list of the twenty-one challenges and assign them ranks from one to five based on their importance. The participants were asked to do so based on their personal experience during a Scrum transition. The weighted average for each challenge is then calculated based on this ranking. A lower weighted average represented a higher rank and vice versa.

The Spearman's rank correlation coefficient and p-value is computed from the ranks given to the importance of each challenge as indicated by the Scrum Coach and the Scrum practitioner. These values give an indication of the correlation between frequency and importance.

5.11.2. Results

A total of 170 responses in total were received for this question. The respondents were categorized into two classes based on whether they were an experienced Scrum Coach or not. A Scrum Coach was categorized as experienced if he/she had at least five years of Scrum coaching experience. A total of twenty-eight experienced Scrum coaches answered this question. The table below lists weighted average of each challenge in terms of its frequency of occurrence as ranked by the 28 Scrum Coaches who took the survey and the 142 Scrum practitioners who took the survey.

The weighted average was calculated for each of these challenges by assigning a weight of 1 to rank-1, 2 to rank-2, 3 to rank-3, 4 rank-4 and 5 to rank-5. Based on this ranking scale, the survey respondents only ranked five challenges from a list of twenty-one which according to them were the most important based on their Scrum transition

experience. For the purpose of calculating the weighted average, the remaining seventeen unranked challenges by each respondent were assigned a weight of six. The weighted average was then calculated for each challenge. A lower weighted average thus pointed out to a more important challenge.

The table below table lists weighted average of each challenge in terms of its importance as ranked by the 28 Scrum Coaches who took the survey and the 142 Scrum practitioners who took the survey. Then each challenge is given a rank from one to twenty one in terms this average rank with one being the highest and twenty one the lowest.

SCRUM TRANSITION CHALLENGES - IMPORTANCE	Practitioner	Practi - Rank	Coach	Coach-Rank
Changing the organizational culture to support Scrum practices	3.97	6	2.71	1
Breaking the waterfall mentality that everything needs to be sequentially completed	5.32	15	4.29	2
Difficulty in delivering "potentially shippable increments" at the end of each sprint	2.70	3	4.32	3
The need for developers to be a "master of all trades"	3.59	5	4.68	4
Poorly executed Scrum meetings (Daily stand-up meetings, Retrospectives, Sprint Review meeting, Sprint Planning Meeting)	2.45	2	5.14	5
Difficulty in creating empowered self-organizing teams and fostering the team mentality	5.11	13	5.14	6
Challenges with distributed teams	4.52	7	5.18	7

Difficulty in adopting Engineering practices (Test Driven Development, Pair Programming, Continuous Integration etc.)	2.79	4	5.25	8
Changing the leadership style from "command and control" to "leadership and collaboration"	5.20	14	5.29	9
Not understanding the values and principles behind Scrum	5.45	17	5.32	10
Determining sprint velocity	2.03	1	5.39	11
Lack of effective Scrum training	4.94	8	5.64	12
Lack of agile compliant performance evaluation	4.99	10	5.71	13
Ineffective Product Owner or Scrum Master	5.37	16	5.75	14
Attempting to scale Scrum at the start	5.79	21	5.75	15
Resistance to change from team members	5.45	18	5.79	16
Fear in developers caused by skill deficiencies	4.99	11	5.82	17
Managing business expectations for deadlines	4.99	9	5.86	18
Difficulty in grooming, estimating and managing the Product Backlog	5.07	12	5.96	19
Lack of top level executive support	5.62	19	6.00	20
Agreeing on the "Definition of Done"	5.75	20	6.00	21

Table 9: Scrum Transition Challenges (Importance) – Coaches V's Practitioners

The line-graph below in Fig 15 provides a visual interpretation of the alignment between frequency and importance.

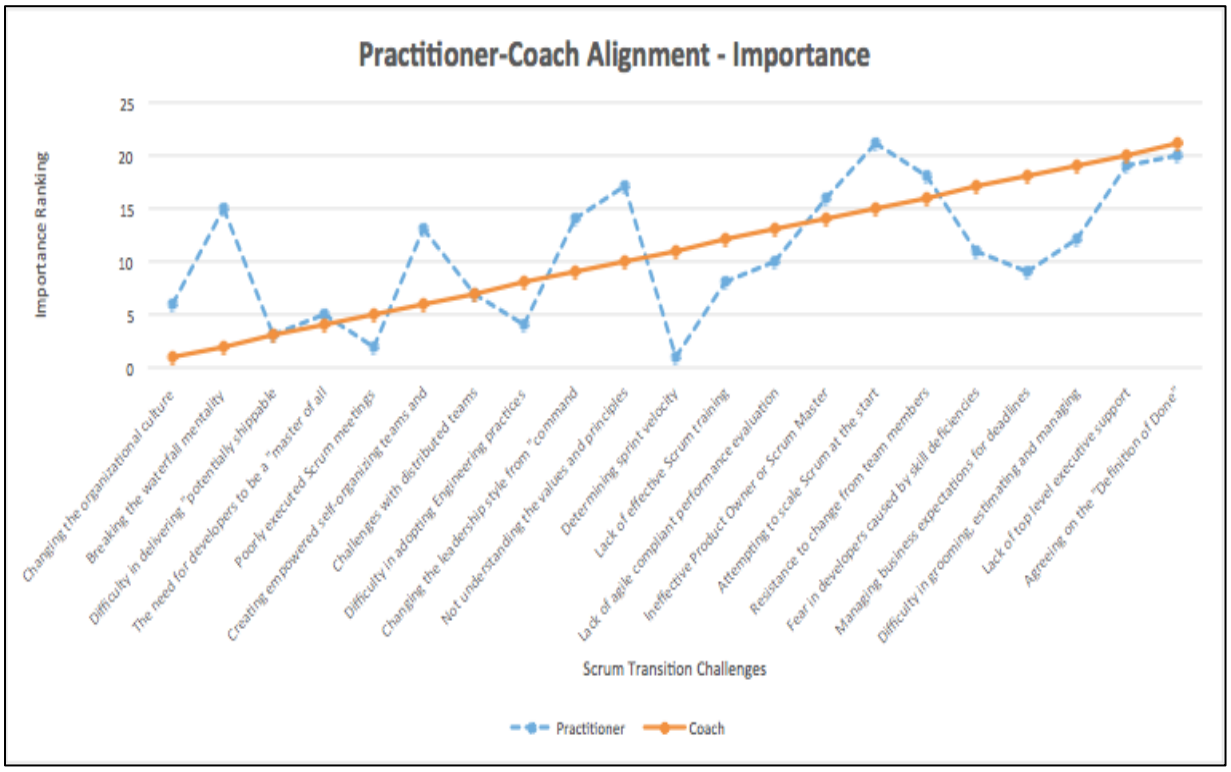

Fig 15: Practitioner V's Coach – Alignment In Importance

To calculate the correlation between frequency of occurrence of each challenge and importance of each challenge, the spearman's rank correlation coefficient was calculated.

Statistical Measurement	Value
Spearman's rank correlation coefficient	0.5584
P-value	0.0085

Table 10: Statistical Measurements For R.Q. 5.

5.11.3. Discussion

The Spearman's rank correlation coefficient has values extending from -1 to 0 to +1 representing perfect negative correlation with +1, no correlation with 0 and perfect positive correlation with +1. The Spearman's rank correlation coefficient of 0.5584 indicates a positive medium correlation. The p-value needs to be interpreted as the probability of seeing the observed correlation if no correlation exists. Since, the p-value is very low (0.0085), we can conclude that it is very improbable that the observed correlation comes from a random effect. In other words, the observed correlation reliably represents actual correlation.

So, to answer the research question, regarding the most frequent challenges, the views of Scrum coaches and Scrum practitioners have a positive medium correlation.

From the line graph, it is clear that there is a high disagreement between Scrum coaches and Scrum practitioners in their responses to the challenge "Determining Sprint Velocity". Scrum practitioners ranked sprint velocity as their most important, while Scrum coaches only gave it a rank of eleven. So, Scrum coaches should take note of this opinion put forward by Scrum practitioners and find solutions to the problem of "Determining Sprint Velocity" so as to facilitate smoother Scrum transitions.

In order to elaborate on this further; Scrum Practitioners and Scrum coaches share slightly different views about the most important challenges faced during a Scrum transition. According to the Scrum coaches, the top five most important challenges are:

- Changing the organizational culture to support Scrum practices
- Difficulty in delivering potentially shippable increments at the end of each sprint
- Breaking the waterfall mentality that everything needs to be sequentially executed

- Poorly executed Scrum meetings, which include the Daily stand- up meetings, Retrospectives, Sprint Review meetings and the Sprint Planning Meeting.
- The need for developers to be a master of all trades.

According to Scrum practitioners, the top five challenges are,

- Determining sprint velocity
- Poorly executed Scrum meetings, which include the Daily stand- up meetings, Retrospectives, Sprint Review meetings and the Sprint Planning Meeting.
- Difficulty in delivering potentially shippable increments at the end of each sprint
- Difficulty in adopting Engineering practices which include Test Driven (Development, Pair Programming, Continuous Integration etc.)
- The need for developers to be a master of all trades.

This chapter quantitatively analyzes the qualitative results from the previous chapter to arrive at the final results for this exploratory sequential mixed methods research design. After discussing and analyzing this data, the next chapter presents the conclusion to this thesis.

CHAPTER 6. CONCLUSION

6.1. Foreword

The objective of this chapter is to present a conclusion to this thesis by summarizing the results obtained in the previous chapters. Furthermore, this chapter also summarizes possible solutions to the challenges identified in the previous chapters and well as some threats to the validity of this study. Finally, this chapter also comprises a set of future perspectives for further research.

6.2. Summary Of Contributions

Given below are the key findings of this thesis:

1. Twenty one challenges that software teams face when they transition from a traditional waterfall software development model to Scrum were identified by means of qualitative interviews with Scrum coaches.

1. R.Q. 1: What are the most frequent challenges?

The first research question ordered these Scrum transition challenges in terms of their frequency of occurrence.

2. R.Q. 2: What are the most important challenges?

The second research question ordered these Scrum transition challenges in terms of their importance.

3. R.Q. 3: Is there a correlation between the frequency and importance of challenges?

The third research question led to the conclusion that the most frequently occurring Scrum transition challenges and the most important Scrum transition challenges have a moderate positive correlation.

4. R.Q. 4: Regarding frequency of challenges, do coaches agree with practitioners?

The fourth research question led to the conclusion that Scrum coaches and general Scrum practitioners are in high positive correlation regarding the most frequently occurring Scrum challenges.

5. R.Q. 5: Regarding importance of challenges, do coaches agree with practitioners?

The fifth research question led to the conclusion that Scrum coaches and general Scrum practitioners have a moderate positive correlation regarding the most important Scrum challenges.

6. Another insightful finding from this study was that the top challenge identified by Scrum practitioners, "Determining Sprint Velocity" was not given much importance by the Scrum coaches. This means that coaches should coaches should pay more attention to determining the team's velocity and helping the team stay on track with it.

6.3. Threats To Validity

There were a number of threats to validity that may have affected the results of the data:

1. Hypothesis guessing – When people take part in an experiment, they may try to figure out what the intended result of the experiment is. Then, they are likely to base their behavior on their guesses about the hypothesis, either positively or negatively, depending on their attitude towards the anticipated hypothesis [74].

2. Evaluation apprehension – Some people are afraid of being evaluated. A form of human tendency is to try to look better when being evaluated which is confounded to the outcome of the experiment [74].

3. Duplicate Responses - There was no guarantee that could have prevented duplicate responses since respondents could have intentionally answered the industry survey questionnaire more than once [74].

4. Unknown population size - Although, there were a total of 210 valid responses, the true size of the population represented by the industry survey questionnaire is unknown [74].

5. No monetary benefit - Since no incentive was offered, some respondents who may feel strongly for or strongly against will respond and outweigh those who do not care enough to do the survey [74].

6.4. Recommended Solutions

This section seeks to recommend a set of best practices that organizations undergoing a Scrum transition could seek to implement. An extensive list of best practices is outside the scope of this thesis, so this section focuses on the top five challenges identified in this study. The following five subsections are organized according to the top five challenges faced in this study.

6.4.1. Determining Sprint Velocity

Software teams measure their efficiency in terms of how much work they can complete over a certain period. This measure is commonly called sprint velocity and

expressed in terms of story points. Predicting velocity is a difficult task especially for teams that are new to each other and new to an agile framework [57]. There are several factors to be taken into account when calculating velocity such as number of developers, maximum amount of work that can be accomplished by one person during the iteration, number of iteration days, actual work output over a period of time and how much backlog value a team can deliver in one iteration. When planning for initial velocity the Scrum Master should not forget to estimate for coffee breaks, emails, design meetings, research, rework etc. [60]. He should remember that the initial estimate velocity is just a ballpark figure. As soon as the first sprint is completed, estimates derived from historical data and blind estimation should be replaced with a predictive range based on the velocity attained in the first sprint. As the project progresses, confidence in the velocity range will increase and previous estimates can be replaced with new ones as more data is gathered [58]

6.4.2. Potentially Shippable Increments At End Of Each Sprint

One of the hardest things for new Scrum teams is to produce potentially shippable software at the end of each sprint. Teams new to Scrum are not sure how to build a system incrementally. Creating working software requires teams to change their thought process, accept the reality of work and focus on an end-to-end scenario [51]. The pressure of having to plan for a release at the end of every two weeks is lessened if the team realizes that they do not have to worry about whether the software would actually ship. They just have to make sure that it is potentially shippable and demonstrable at the each sprint and meets the definition of done criteria [59].

6.4.3. Effective Scrum Meetings

Programmers often poorly speaking of agile because they feel that they lose a lot of time in poorly executed Scrum meetings. Either the meetings are too frequent or too long, or they are not adding expected business value. In order to have a successful Scrum project, effective Scrum meetings have to be held. In an effective daily scrum meeting, , members stand instead of sit when keeps the meeting short and to the point. The goals of the daily Scrum are communicate status of the project, identify any obstacles the team has come across, set plans for the day and help build unity in the team. Summaries should be short and each member should focus on only the following three items which are what was done yesterday, what will be done today and what issues may cause problems for progress. The meeting should also be held in the same place every day and in fifteen minutes or less [57].

6.4.4. Engineering Practices

Good engineering practices are essential to becoming a high-performing Scrum team [57]. Continuous Integration does not start and end at setting up the system. The real challenge is to keep it running smoothly over time. As the daily build gets larger, the key is to keep integration repeatable, fast and simple [58]. Pair programming is something a lot of developers are uncomfortable with. Some common tactics that helped the adoption of pair programming are implementing guidelines, having a champion, creating a positive atmosphere and creating a room dedicated for pair programming [60].

6.4.5. Changing The Organizational Structure To Support Scrum Practices

One of the largest barriers to organizational change is the culture, which is the shared set of mental assumptions that define 'how work is done' [46]. Most of the values that comprise organizational culture are completely subconscious, and are therefore hard to identify [37]. Scrum is all about change. Some people embrace change and some avoid it like the plague. What is often not understood is that change is inevitable, sudden market shifts or a competitor's newest release cannot be predicted before it happens. To implement a successful Scrum transition, change must be embraced [48]. When team members begin to understand the principles and learn the practices of Scrum, they begin to realize Scrum's benefits. As these benefits emerge, team members embrace the new reality. Chaos begins to fade and the team moves towards a new equilibrium. As people learn to work together, performance starts to improve often, exceeding levels prior to the change [46].

6.5. Future Work

The findings of this thesis are valuable to understanding the challenges faced during a transition from Waterfall to Scrum. However they pave the way for several opportunities for future work namely:

- Conduct more extensive surveys with more participants to further investigate these challenges.
- Test the findings of this thesis by comparing them to the results of the above case studies and surveys.

- Conduct similar studies to identify challenges that software teams face when they transition to other agile methodologies such as eXtreme Programming, Kanban, Lean, etc.
- Conduct a survey and a more detailed study to gain a better understanding of Scrum best practices.

BIBLIOGRAPHY

[1]. A. Cockburn and J. Highsmith, “Agile Software Development: The Business of Innovation." Computer 34.9 (2001): 120-27. Web.

[2]. A. Tashakkori, & C.Teddlie, (2003). Handbook of Mixed Methods in Social & Behavioral Research. Thousand Oaks: Sage.

[3]. Cohn, Mike. Agile Estimating and Planning. Upper Saddle River, NJ: Prentice Hall Professional Technical Reference, 2006. Print

[4]. Abran, J.W. Moore, Guide to the Software Engineering Body of Knowledge (SWEBOK®), IEEE Computer Society 2004 Guide, Angela Burgess, 2004.

[5]. Boehm, R. Turner, “Management Challenges to Implement Agile Processes in Traditional Development Organizations”, IEEE Software (22)5, 2005, pp. 30-39.

[6]. S. Bowen, F. Maurer, “Process Support and Knowledge Management for Virtual Teams Doing Agile Software Development”, Proceedings of the 26th Annual International Computer Software and Applications, Conference, IEEE Computer Society Press, 2002, pp. 1118-1120.

[7]. G. Corbitt, L.R. Gardiner, L.K. Wright, “A Comparison of Team Developmental Stages, Trust and Performance for Virtual versus Face-to-Face Teams”, Proceedings of the 37th Hawaii International Conference on System Sciences, 2004

[8]. T. Dybå, T. Dingsøyr, “Empirical Studies of Agile Software Development: a Systematic Review”, Journal of Information andSoftware Technology 50 (2008), 2008, pp. 833-859.

[9]. J. Erickson, K. Lyytinen, K. Siau, “Agile Modeling, Agile Software Development, and Extreme Programming: the State of Research”, Journal of Database Management 16 (4), 2005, pp. 88-100.

[10]. J. Creswell, (2003). Research design: Qualitative, quantitative, and mixed methods approaches. Thousand Oaks: Sage.

[11]. Kitchenham, S. Charters, Guidelines for Performing Systematic Literature Reviews in Software Engineering, Technical Report EBSE- 2007-01, School of Computer Science and Mathematics, Keele University, 2007.

[12]. K. Petersen, R. Feldt, S. Mujtaba, M. Mattsson, “Systematic Mapping Studies in Software Engineering”, 12th International Conference on Evaluation and Assessment in Software Engineering, June 2008, pp.71-80.

[13]. Hossain, M. Babar, H. young Paik, “Using Scrum in Global Software Development: a Systematic Literature Review”, Fourth IEEE International Conference on Global Software Engineering, 2009, pp. 175-184.

[14]. Hossain, M. Babar, H. young Paik, J. Verner, “Risk Identification and Mitigation Processes for Using Scrum in Global Software Development: a Conceptual

Framework", Asia-Pacific Software Engineering Conference, 2009, pp. 457-464.

[15]. L. Hvatum, "Agile Practices and Distributed Teams", Cutter IT J. (USA) 20(5), 2007, pp. 6-11.

[16]. Kotz, Samuel, Norman Lloyd. Johnson, and Campbell B. Read. Encyclopedia of Statistical Sciences. New York: Wiley, 1982. Print.

[17]. J. Rothman, "Agility in a Box [Project Scheduling Tool]", Soft. Dev. (USA) 12(3), 2004, pp. 25-7.

[18]. Roussev, R. Akella, "Agile Outsourcing Projects: Structure and Management", International Journal of e-Collaboration 2(4), 2006,pp. 37-52.

[19]. R. Sangwan, P. Laplante, "Test-driven Development in Large Projects", IT Professional 8(5), 2006, pp. 25-29.

[20]. T. Schummer, S. Lukosch, "Supporting the Social Practices of Distributed Pair Programming", Lecture Notes in Computer Science,vol. 5411 LNCS, 2008, pp. 83-98.

[21]. Nerur, S., Mahapatra, R. and Mangalaraj, G, "Challenges of Migrating to Agile Methodologies", Communications of the ACM, Vol. 48, No. 5, May 2005, pp. 72-78.

[22]. Gopal, T. Mukhopadhyay and M. S. Krishnan, "The Role of Software Processes and Communication in Offshore Software Development," Communications of the

ACM, Vol. 45, No. 4, 2002, pp. 193-200. doi:10.1145/505248.506008

[23]. E. Fossey, C. Harvey, F. McDermott and L. Davidson, "Understanding and Evaluating Qualitative Research," Australian & New Zealand Journal of Psychiatry, Vol. 36, No. 6, 2002, pp. 717-732.

[24]. N. B. Moe, T. Dingsoyr and T. Dyba, "Overcoming Barriers to Self-Management in Software Teams," Software, Vol. 26, No. 6, 2009, pp. 20-26.doi:10.1109/MS.2009.182

[25]. S Dorairaj, J Noble, P Malik Agile Processes in Software Engineering and Extreme Programming, 47-61

[26]. N. B. Moe, T. Dingsoyr and T. Dyba, "Overcoming Barriers to Self-Management in Software Teams," Software,Vol.26,No. 6, 2009, pp. 20-26.doi:10.1109/MS.2009.182

[27]. P. Laplante, C. Neill "The Demise of the Waterfall Model Is Imminent" and Other Urban Myths " – ACM Queue, Feb 2004

[28]. J. Grenning, "Planning Poker or How to avoid Analysis Paralysis while Release Planning," Hawthorn Woods: Renaissance Software Consulting, Vol. 3, 2002

[29]. J. Iivari and N. Iivari, "The relationship between organizational culture and the deployment of agile methods," Information and Software Technology, vol. 53, pp. 509-520, 2011.

[30]. J. P. Wan and R. T. Wang, "Empirical Research on Critical Success Factors of Agile Software Process Improvement," Journal of Software Engineering and Applications,

Vol. 3, No. 12, 2010, pp. 1131-1140. doi:10.4236/jsea.2010.312132

[31]. J. Shrinivasavadhani and V. Panicker. Remote mentoring a distributed agile team. In AGILE, pages 322–326. IEEE, 2008.

[32]. J. Srinivasan and K. Lundqvist, "Agile in India: Challenges and lessons learned," in 3rd India Software Engineering Conference, ISEC'10, Mysore, India, 2010, pp. 125-130.

[33]. K. Beck and C. Andres, Extreme Programming Explained: Embrace Change, 2nd ed. Boston, MA: Addison-Wesley Professional, 2004.

[34]. K. Beck, A. Cockburn, R. Jeffries, and J. Highsmith. (2001, July 2013). Agile manifesto. Available: http://www.agilemanifesto.org

[35]. K. Conboy, S. Coyle, X. Wang, and M. Pikkarainen, "People over process: Key challenges in agile development," IEEE Software, vol. 28, pp. 48-57, 2011.

[36]. Nagappan, Nachiappan, E. Michael Maximilien, Thirumalesh Bhat, and Laurie Williams. "Realizing Quality Improvement through Test Driven Development: Results and Experiences of Four Industrial Teams." Empirical Software Engineering 13.3 (2008): 289-302. Web.

[37]. K. Sureshchandra and J. Shrinivasavadhani, "Moving from waterfall to agile," in Agile 2008 Conference, AGILE '08, Toronto, ON, 2008, pp. 97-101.

[38]. L. Adkins. Coaching Agile Teams: A Companion for ScrumMasters, Agile Coaches,

and Project Managers in Transition. Addison-Wesley, 2010.

[39]. L. Anderson, G. B. Alleman, K. Beck, J. Blotner, W. Cunningham, M. Poppendieck, and R. Wirfs-Brock, "Agile management - an oxymoron?: who needs managers anyway?," presented at the Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications, Anaheim, CA, USA, 2003.

[40]. L. Radha alais Nagalakhmi, Dr. Mayank J. Trivedi, 2010.Utilization of online survey tools for academic research: A practical approach to survey monkey

[41]. M. A. Cusumano and R. W. Selby, “Microsoft Secrets: How the World’s Most Powerful Software Company Creates Technology, Shapes Markets and Manages People,” Free Press, New York, 1995.

[42]. M. Q. Patton. Qualitative evaluation and research methods. Sage Publications, Newbury Park, Calif., 2nd edition, 1990.

[43]. M. Senapathi and A. Srinivasan, "Understanding post-adoptive agile usage: An exploratory cross-case analysis," Journal of Systems and Software, vol. 85, pp. 1255-1268, 2012.

[44]. Mary Poppendieck, August 2004. “Unjust Desserts” www.stickyminds.com

[45]. N. B. Moe, A. Aurum, and T. Dybå, "Challenges of shared decision- making: A multiple case study of agile software development," Information and Software Technology, vol. 54, pp. 853-865, 2012.

[46]. N. Ganesh and S. Thangasamy, "Lessons learned in transforming from traditional to agile development," Journal of Computer Science, vol. 8, pp. 389-392, 2012.

[47]. Nis Ovesson The Challenges of Becoming agile- Implementing and conducting Scrum In Integrated product development , June 2012, PHD Thesis, Alaborg University

[48]. P. Deemer, G. Benefield, C. Larman and B. Vodde, “The SCRUM Primer,” 2013. http://www.brianidavidson.com/agile/docs/scrumprimer121.pdf

[49]. P. Tsirakidis, F. Köbler, and H. Krcmar, "Identification of success and failure factors of two agile software development teams in an open source organization," in 4th IEEE International Conference on Global Software Engineering, ICGSE 2009, Limerick, 2009, pp. 295-296.

[50]. R. Hoda, "Self-Organizing Agile Teams: A Grounded Theory," PHD thesis, Victoria University of Wellington, New Zealand, 2011.

[51]. R. Hoda, J. Noble, and S. Marshall, "Developing a grounded theory to explain the practices of self-organizing Agile teams," Empirical Software Engineering, vol. 17, pp. 609-639, 2011.

[52]. R. Hoda, J. Noble, and S. Marshall, "The impact of inadequate customer collaboration on self-organizing Agile teams," Information and Software Technology, vol. 53, pp. 521-534, 2011.

[55]. R. Hoda, J. Noble, and S. Marshall. Organizing self- organizing teams. In Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering

– Volume 1, ICSE ’10, pages 285–294, New Y ork, NY , USA, 2010. ACM.

[56]. Rich C. Lee, “The Success Factors of Running Scrum: A Qualitative Perspective,” Journal of Software Engineering and Applications, Vol. 5, No. 6, 2012, pp. 367-374. doi:10.4236/jsea.2012.56043

[57]. Roche, G. and Vaquesz-McCall, B. 2009. The amazing team race – a team based on the agile adoption. In Proceedings of the Agile 2009 Conference. IEEE, Los Alamitos, CA, 141- 146.

[58]. S. Adolph, W. Hall, and P. Kruchten, "Using grounded theory to study the experience of software development," Empirical Software Engineering, vol. 16, pp. 487-513, 2011.

[59]. S. C. Misra, V. Kumar, and U. Kumar, "Identifying some important success factors in adopting agile software development practices," Journal of Systems and Software, vol. 82, pp. 1869-1890, 2009.

[60]. Sutherland, Jeffrey Victor. Scrum: A Revolutionary Approach to Building Teams, Beating Deadlines, and Boosting Productivity. London: Random House Business, 2014. Print.

[61]. S. Nerur, R. Mahapatra, and G. Mangalaraj, "Challenges of migrating to agile methodologies," Communications of the ACM, vol. 48, pp. 72-78, 2005.

[62]. Schatz, B., and I. Abdelshafi. "Primavera Gets Agile: A Successful Transition to Agile Development." IEEE Software 22.3 (2005): 36-42. Web.

[63]. T. J. Gandomani, H. Zulzalil, A. A. A. Ghani, A. M. Sultan, and M. Z. Nafchi, "Obstacles to moving to agile software development; at a glance," Journal of Computer Science, vol. 9, pp. 620-625, 2013.

[64]. Vinekar, V., Slinkman, C. and Nerur, S. “Can Agile and Traditional Systems Development Approaches Co-exist? An Ambidextrous View,” Information Systems Management Journal, Vol. 23, No. 3, Summer 2006, pp. 31-42

[65]. Wald, A. Sequential Tests of Statistical Hypotheses. Ann. Math. Statist. 16 (1945), no. 2, 117--186. doi:10.1214/aoms/1177731118.

[66]. Miaoulis, George, and R. Dean Michener. An Introduction to Sampling. Dubuque, IA: Kendall/Hunt Pub., 1976. Print.

[67]. Flor, Nick V. "Globally Distributed Software Development and Pair Programming." Communications of the ACM 49.10 (2006): 57. Web.

[68]. Duvall, Paul M., Steve Matyas, and Andrew Glover. Continuous Integration: Improving Software Quality and Reducing Risk. Upper Saddle River, NJ: Addison-Wesley, 2007. Print.

[69]. Richardson, Jared R., and William Atwill. Gwaltney. Ship It!: A Practical Guide to Successful Software Projects. Raleigh, NC: Pragmatic helf, 2005. Print.

[70]. Löwe, Michael. "Refactoring Information Systems." ACM SIGSOFT Software Engineering Notes 36.4 (2011): 1. Web.

[71]. Cohn, Mike L. "Setting and Managing Expectations." Mike Cohns Blog Succeeding With Agile RSS. Mike Cohn, 03 Mar. 2012. Web. 06 Apr. 2015.

[72]. Varma, Tathagat. "Automated Software Testing." ACM SIGSOFT Software Engineering Notes 25.3 (2000): 65. Web.

[73]. Wohlin, Claes. Experimentation in Software Engineering. Heidelberg: Springer, 2012. Print.

[74]. Kautz, Karlheinz; Pedersen,, Casper Fabricius; and Monrad, Ole, "Cultures of Agility – Agile Software Development in Practice" (2009). *ACIS 2009 Proceedings.* Paper 87.

[75]. Wald, A. Sequential Tests of Statistical Hypotheses. Ann. Math. Statist. 16 (1945), no. 2, 117--186. doi:10.1214/aoms/1177731118.

APPENDICES

APPENDIX A. GLOSSARY AND ACRONYMS

A.1. Acceptance Criteria: It is defined as the product characteristics, specified by the Product Owner, that need to be satisfied before they are accepted by the user, customer, or other authorized entity. These are used as standards to measure and compare the characteristics of the final product with specified characteristics.

A.2. Acceptance Test: Acceptance tests are tests are run from a business and customer point of view. These tests check the requested and implemented feature and determine whether these features match the business and the customer requirements

A.3. Acceptance Test Driven Development: This is defined as a system or product development method in which the acceptance criteria are discussed extensively by the participants, through the use of examples and well-designed acceptance tests on the basis of the these criteria before development begins.

A.4. Agile: Agile is a group of iterative and incremental software development methods. It encourages flexibility and speed in responding to change. It requires collaboration between self-organized, cross-functional teams to generate requirements and solutions.

A.5. Burn-up Chart: A burn up chart is a chart showing the evolution of an increase in a measure against time. Burn-up charts are an optional implementation within Scrum to make progress transparent.

A.6. Code Refactoring: Code refactoring is a technique used in software development for restructuring/redesigning an existing body of code without changing its behavior.

The purpose of re-factoring is to improve non-functional attributes of the software, e.g., managing technical debt or making coding faster.

A.7. Continuous Delivery: Continuous delivery is the continuous process of delivering the product or each product feature to its users immediately after it is integrated and tested by the developer.

A.8. Continuous Deployment: Continuous deployment is defined as the process of delivering the product or each product feature to its users immediately after it is integrated and tested by the developer.

A.9. Continuous Improvement: Agile aims to continuously learn and improve during each project and apply lessons learned within the current project. Several tools, techniques, knowledge sets, and skills can continuously improve Agile projects - e.g. retrospective meetings, knowledge sharing etc.

A.10. Cost of Delay: Cost of delay is defined as any monetary loss incurred due to delay in work, process, or achieving production targets. This concept emphasizes that the time associated with project has a financial cost.

A.11. Cross Functional Team: A cross functional team is a project team that has expertise from different fields like designers, developers, and testers who have skills required to complete the work effectively and efficiently.

A.12. Confidence Interval: The confidence interval defines how frequently the observed interval contains the parameter.

A.13. The Confidence Level: When a population is repeatedly sampled, the average value of the attribute obtained by those samples is equal to the true population value. In a normal distribution, approximately 95% of the sample values are within two standard

deviations of the true population value. In other words, this means that if a 95% confidence level is selected, 95 out of 100 samples will have the true population value within the range of precision specified earlier.

A.14. Degree of Variability: The degree of variability in the attributes being measured, refers to the distribution of attributes in the population. The more heterogeneous a population, the larger the sample size required to obtain a given level of precision.

A.15. Daily Standup/Scrum: These are daily time-boxed events of 15 minutes, or less, for the Development Team to re-plan the next day of development work during a Sprint. Updates are reflected in the Sprint Backlog. The daily standup meeting, or Scrum meeting, is a daily team meeting in the Scrum Framework. The name comes from the practice of the attendees standing up. This encourages the members to keep the meeting short. It gives the team a regular opportunity to monitor progress along the sprint plan.

A.16. Definition of Done: A definition of done is a shared understanding of expectations that software must live up to in order to be releasable into production which is managed by the Development Team.

A.17. Development Team: The role within a Scrum Team accountable for managing, organizing and doing all development work required to create a releasable Increment of product every Sprint. A development team is formed with members from different areas of functional expertise. It has to be self-organized, and it must drive toward a single goal. This team is collectively responsible developing of an acceptable product.

A.18. Engineering standards: Engineering standards are a shared set of development and technology standards that a Development Team applies to create releasable Increments of software.

A.19. Epic: An epic is a large user story, typically one that is too big to fit in a single sprint. Epics need to be broken down into smaller user stories at some point before implementation as part of a sprint.

A.20. Estimation: An estimation is a rough calculation of the number, quantity, or size of product backlog items, portfolio backlog item, and sprint backlog task.

A.21. Forecasting: The selection of items from the Product Backlog that the development team deems feasible for implementation in a Sprint. Estimating or predicting future project status and progress based on knowledge and information available at the time of forecasting.

A.22. Functional Test: Functional Testing usually describes what the system does. Here functions are tested by feeding input and examining the output. It is type of 'Black Box Testing' where we do not consider the internal program structure and mostly compare the actual and expected outputs.

A.23. Increment: An increment is a piece of working software that adds to previously created Increments, where the sum of all Increments -as a whole - form a product.

A.24. Integration: Integration is the combination of various components of a product to form a coherent, larger-scope work product that can be validated to function correctly as a whole.

A.25. Internal Stakeholders: Internal stakeholders are the stakeholders who are internal to the organization, i.e., those who are involved in product development. For example, senior executives, managers and internal users.

A.26. Interviews: A formal or informal approach to obtain information from stakeholders by talking to them directly

A.27. Iterative product development: In iterative product development, the final product is developed over a few iterations and delivered to the customer.

A.28. Minimum Marketable Feature: The smallest or minimum set of functionality related to a feature that must be delivered for the customer to perceive value (for it to be marketable). Contrast with "minimum releasable features."

A.29. Minimum Viable Product (MVP): A product with just those minimal features that allow it to be deployed, and no more. Usually, MVP is the result of the first sprint.

A.30. Pair Programming: Pair programming is a programming technique in which two programmers working together on a single system. This practice has been around since the 1950s. Many studies have shown that pair programmers are more than twice as efficient; in code production and especially in freedom from bugs, than one single programmer on a single system.

A.31. Product Backlog: An ordered list of the work to be done in order to create, maintain and sustain a product. Managed by the Product Owner.

A.32. Product Backlog refinement: The activity in a Sprint through which the Product Owner and the Development Team add granularity to the Product Backlog.

A.33. Product Owner: The role in Scrum accountable for maximizing the value of a product, primarily by incrementally managing and expressing business and functional expectations for a product to the Development Team(s).

A.34. Product Roadmap: A product road map is a high-level plan that shows when in the future new products are expected to be developed or introduced by the organization/team. The requests to edit the road map (usually by adding new products)

come from the sales force or senior management in the company when the marketing strategy is made.

A.35. Product Vision: A statement describing the desired future state that would be achieved by developing and deploying a product. A good product vision is simple, easy to understand statement and provides a coherent direction to the people who are asked to realize it.

A.36. Release Planning: A term borrowed from Lean Manufacturing. The function of release planning is to synchronize projected range of potential delivery dates in the future with tasks to be done today.

A.37. Scrum: A framework to support teams in complex product development. Scrum consists of Scrum Teams and their associated roles, events, artifacts, and rules, as defined in the Scrum Guide.

A.38. Scrum Board: A physical board to visualize information for and by the Scrum Team, often used to manage Sprint Backlog. Scrum boards are an optional implementation within Scrum to make information visible.

A.39. Scrum Guide: The definition of Scrum, written and provided by Ken Schwaber and Jeff Sutherland, co-creators of Scrum. This definition consists of Scrum's roles, events, artifacts, and the rules that bind them together.

A.40. Scrum Master: The role within a Scrum Team accountable for guiding, coaching, teaching and assisting a Scrum Team and its environments in a proper understanding and use of Scrum.

A.41. Scrum Team: A self-organizing team consisting of a Product Owner, Development Team and Scrum Master.

A.42. Scrum Values: A set of fundamental values and qualities underpinning the Scrum framework; commitment, focus, openness, respect and courage.

A.43. Self-organization: The management principle that teams autonomously organize their work. Self-organization happens within boundaries and against given goals. Teams choose how best to accomplish their work, rather than being directed by others outside the team.

A.44. Sprint: Time-boxed event of 30 days, or less, that serves as a container for the other Scrum events and activities. Sprints are done consecutively, without intermediate gaps.

A.45. Sprint Backlog: an overview of the development work to realize a Sprint's goal, typically a forecast of functionality and the work needed to deliver that functionality. Managed by the Development Team.

A.46. Sprint Goal: a short expression of the purpose of a Sprint, often a business problem that is addressed. Functionality might be adjusted during the Sprint in order to achieve the Sprint Goal.

A.47. Sprint Planning: time-boxed event of 1 day, or less, to start a Sprint. It serves for the Scrum Team to inspect the work from the Product Backlog that's most valuable to be done next and design that work into Sprint backlog.

A.48. Sprint Retrospective: time-boxed event of 3 hours, or less, to end a Sprint. It serves for the Scrum Team to inspect the past Sprint and plan for improvements to be enacted during the next Sprint.

A.49. Sprint Review: time-boxed event of 4 hours, or less, to conclude the development work of a Sprint. It serves for the Scrum Team and the stakeholders to inspect the

Increment of product resulting from the Sprint, assess the impact of the work performed on overall progress and update the Product backlog in order to maximize the value of the next period.

A.50. Stakeholder: a person external to the Scrum Team with a specific interest in and knowledge of a product that is required for incremental discovery. Represented by the Product Owner and actively engaged with the Scrum Team at Sprint Review.

A.51. Story Point: The abstract measure of effort to implement a story is called a story point. Typically determined by engaging in planning poker.

A.52. Sustainable Pace: The appropriately aggressive pace at which a team works so that it produces a good flow of business value over an extended period of time without getting burned out.

A.53. Task Estimation: The team breaks down the selected Product Backlog items into tasks and then the tasks are estimated by the team members according to their complexity, risk involved, potential time required, and so on using team exercises.

A.54. Technical Debt: A status category for technical debt that represents the debt that is known to the development team and has been made visible for future consideration. Contrast with "happened-upon technical debt" and "targeted technical debt." See also "technical debt."

APPENDIX B: INTERVIEW TRANSCRIPTS

B.1. Scrum Coach A

According to Scrum Coach A, the 10 most important issues are:

1. Structural changes at organizations:

Most traditional software companies are organized around functional disciplines or roles. There are teams of developers, testers and analysts. Scrum does not necessarily require that people be given new managers, but it requires that thought be put into how people are managed.

"The success of a Scrum implementation depends less on the manager and more on the team. Without the structural changes to support this new organization, managers often find themselves with less to do. At the very least, it's complicated."

2. Problem solving in the daily Scrum meeting:

The team should understand that the daily Scrum meeting is not for finding solutions to those problems. Daily Scrums should essentially be time boxed to 15 minutes and be limited to answering the three questions. What have you done yesterday? What will you do today? What issues do you have? The Scrum methodology lays out a simple set of rules. Teams that do not follow these simple rules set themselves up for failure.

3. Requires a cultural shift

Going from top down management to a self-empowered team approach is hard. There is a lot of structure and discipline in Scrum, but it is not imposed from the top in

a command and control manner. This is the responsibility of the self-empowered team. This is a significant shift from how a lot of companies operate today.

4. Human resource policies

One of the most significant contributions that senior management can make toward a successful transition is to work with the human resources and payroll processing departments to align them with this new way of working. What is being referred to here are job descriptions, the review process, compensation and bonuses.

"There are some new behaviors that Scrum wants to encourage. But there is nothing more frustrating than when the policies that govern the organization do not align with these new behaviors. It is not fair to ask people to work against their best interests."

5. Changing the management style

One approach that organizations follow to redefine the role of team managers is to convert them to Scrum Masters for their team. This has a poor success record. When the manager plays the role of Scrum Master, it becomes highly unlikely that the team will ever begin to self organize.

"The previous habits of order giver and order follower are difficult to break. What will most likely happen instead is that the currently existing command and control values and patterns will be transplanted into the heart of the Scrum practices. As a result, the benefits that follow a self-organizing Team which are ownership, focus, drive, quality, improved morale and better productivity will not likely be realized."

6. Managing business expectations for deadlines

Managing the expectations of customers, salespeople, marketers, managers and developers is one of the key roles of the Product Owner. Business people who are used to waterfall projects are used to the idea of having a plan that gives dates for when all the features will be delivered. Scrum does not provide set dates for product delivery. Some rough estimates may be provided, but they too could change depending on how the project goes. This may create challenging situations.

7. Ensuring technical excellence

Scrum is a lightweight project management framework. Although, the founders of Scrum initially refrained from prescribing any engineering practices, Scrum teams have experienced success by pairing the framework with practices such as pair programming, sustainable pace to prevent burnout, coding standards, endless refactoring, acceptance tests, unit tests and continuous integration. Initiating these practices and convincing people to stick with them is a challenging process.

8. Need for the developer to be a master of all trades

Boundaries between developer roles are less clear in Scrum and it is important that developers are competent in a broad range of skills and not just in any one.

"To be a successful Scrum core team member, you need to be a coder, a tester, an architect, a customer, a quality engineer and a lot of other things."

The need for a multi-faceted skill set causes numerous problems. First, almost all project managers find it difficult to find developers that display all the skills necessary for agile either externally or within their respective organization. Training is also more difficult and expensive. The fact that agile encourages blended roles, is dependent on voluntary contributions and emphasizes teamwork as opposed to

individual performance, means that team members may become a 'jack of all trades' but lack the opportunity to hone a smaller number of key skills.

9. Creating and maintaining a concise backlog

The sheer size of new product backlogs can be overwhelming. There is so much to keep track of. When the backlog is raw and not prioritized, it can be hard to know where to begin.

"The customers and stakeholders will look at the stories in a much different way than the team. They are not interested in how many points a story has. What they are interested in is finding the stories that relate to them and making sure that those get done. This may put the stakeholders in conflict with one another, but this reflects reality and the product owner will needs to make some tough decisions in such situations."

10. Delivering potentially shippable increments at the end of each sprint

One of the hardest things for new teams to do is to produce potentially shippable software at the end of each sprint. Teams new to Scrum, are just not sure how to build a system incrementally. They tend to jump in and build an entire package, a database for example before they have a way to validate it. In the waterfall software development lifecycle, entire software components are built all at once, integrating them near the end of the project. The problem with this is that each component on its own does not have any value. These components only have value when they are integrated. This is the problem that Scrum seeks to solve by starting with a skeleton of the entire project and then building on that.

"The challenge is to fight the urge to build a complete component of a system, when the rest of the system does not work. Focusing on end to end scenarios gives the team something to show the customer while keeping the code malleable".

B.2. Scrum Coach B

According to Scrum Coach B, the most important issues are:

1. Motivating the team

If people have to be motivated, they need to be vested in the project and they will only be vested if they are doing what they believe is right, not what they are ordered to do.

"Management has to pay attention to what the team says. If the team says something does not work or needs to be changed, and the management just ignores that, than the team members will simply check out and let the management deal with the project."

2. Scrum Training

Sometimes, when official Scrum training is provided, it is usually only given to the developers or the technical team. If the business part of the team is not included at this point, then team division sets in before the first sprint begins. When business teams are invited, the training content is usually focused on the changes the technical members of the team should adopt. Often times, it does not address how business stakeholders fit in or how long-term planning should be completed.

This introduces a split in two ways. Firstly, business stakeholders are skeptical whether Scrum can provide everything they need. Secondly, the development teams are left unaware of the new challenges the business stakeholders face.

3. Engineering Practices

Engineering Practices are essential to becoming a high performing Scrum Team. There are five key engineering practices in Scrum, which are test-driven development, refactoring, continuous integration, pair programming and automated acceptance tests. All these seem to very challenging initially. But, the team will recover the time later through efficiency, improved quality and code stability.

4. Keeping People Engaged with Pair Programming

Pair programming is one of those things that people either love or hate. What no one can argue is that when done well, pairing produces high quality software in a relatively short amount of time.

"The problem with pair programming tends to be keeping people engaged when they are not the ones with their hands on the keyboard. Having to work with another person on what is usually a solitary practice can be distracting or tiring. Whatever the cause, people tend to check out. When that happens, immediate remedial action should be taken."

5. Weak Product Owner

The product owner is a key role in Scrum. But many teams and organizations struggle to fully understand and effectively apply this role.

"This role is very important. Sometimes we get a great person, but often something is still wrong. Sometimes the person does not have enough decision-making

authority, sometimes the person is not trained to be a Product Owner; sometimes the Product Owner is not interested in attending Sprint goals demonstration etc. At other times, the Product Owner may be focused on project deadlines, leading to a continuous process of scope negotiation just to meet such deadlines. That causes technical debt to increase continuously thereby making it more difficult to add features as the project evolves."

6. Potentially Shippable Increment at the end of each sprint

The number one key to a successful Scrum project is to deliver a potentially shippable increment at the end of each sprint in which the code is tested and the technical debt is low.

"Managing defects in any software project is a challenge. People have years of muscle memory that tells them that they should fix defects at the end, after development is completed. Therefore, when they transition to Scrum, there is a natural feeling that defects should be fixed at the end of a sprint, or have a bug-fixing sprint. In short developer's brains have to be retrained to have a product release every 2 weeks or so."

7. Distributed Teams

There are several challenging areas that need to be addressed for distributed team management. The first one is communication, which is one of the core issues among distributed teams. Different time zones and conflicting working hours impact communication and collaboration. The second one is that cultural and language differences impact communication and collaboration. An effective tool chain is needed for requirements repo's, deployment setup, bug tracking, and project management tools.

The third one is that scheduling differences at the team level for various activities becomes more challenging with increasing levels of distribution.

8. Running a daily productive stand up meeting

The daily stand up meeting or the daily Scrum, often does not get the respect it deserves. Done correctly, daily stand up meetings keep everyone on the same page for the daily deliverables and moving as one towards the sprint goal. Done poorly, the meeting becomes a mere status meeting or a finger pointing session where team members feel the urge to defend their past actions. People quickly find that they are spending far too much time talking and not enough time doing.

9. Sprint Retrospective

The retrospective is the opportunity for team members to learn how to improve, work more efficiently and deliver at a higher velocity and superior quality.

"As things get tough, retrospectives are one of the first Scrum elements to go by the wayside. As schedule pressures mount, teams feel that they do not have time to do retrospectives. Once teams start skipping retrospectives, the downward spiral begins. Retrospectives are a key part of the team's inspect and adapt cycle."

10. Empowering the self organized team

The traditional role of a manager in the corporate world is based on a model known as "command and control". Scrum is based on a different approach, which is the self-organizing team. This difference is evident from the first steps the team takes. In Scrum, the team decides on how much work to commit to in a sprint.

"Teams are so conditioned to follow orders that they will often not begin to self-organize until there are no orders available to follow. This requires a leap of faith for

the manager who is called a Product Owner or Scrum Master within the Scrum framework. The manager needs to change his style of interaction and constantly signal to the team members that they are now responsible for themselves."

B.3. Scrum Coach C

According to Scrum coach C, the 10 most important challenges are:

1. Corporate Culture Roadblocks

A strong corporate culture not accustomed to Scrum values can be a hard environment for implementing Scrum. Adopting Scrum processes can be a challenge if the philosophy itself is not fully embraced. In some cases, the company's incentive structure may reward traditional benchmarks while inadvertently discouraging the success of Scrum.

"That's a tough one to manage because philosophy and the big picture tend to pale in comparison to the paycheck."

2. Lack of support from management

The principles of Scrum are derived from those of lean manufacturing. In order to run a lean manufacturing facility, businesses must have a management system in place that is fully in support of the transition. If a software company's management team is hands-off or it fails to proactively help teams overcome roadblocks, the business will not be able to fully leverage the benefits of Scrum.

3. Team Members are Resistant to Change

Change is inherently difficult and uncomfortable, especially for experienced team members who have a long history of waterfall successes to look back on.

"The average Jack sitting at a workstation in the engineering development team can throw a monkey wrench into a Scrum transformation when he maintains a stubborn "this is how we have always done it" attitude. Even if the transition is lucky enough to have top management support, it is hard to achieve success if team members are not on board."

4. Team members having generalized skill sets

A very skilled and an efficient project team is required to implement Scrum effectively and successfully. All the roles, that is, the Scrum Master, the Product Owner and the development team members need to be aware of the Scrum principles, and adhere to them as effectively as possible.

"Part of a successful Scrum implementation relies on the development team members being generalists. They should be cross trained so as to be able to identify and execute points of convergence for the skill sets involved in the project. At the same time, they should have the advanced skills and acumen in a specific skill set that adds high degrees of quality. The team members should also have a strong sense of dedication and commitment to the project and to the Scrum principles."

5. Scrum Values

Any framework worth using is built on a set of values. These values guide people, provide clarity in times of ambiguity and most importantly help people understand why they do things in a certain way. The values behind the Scrum framework are focus, respect, commitment, courage and openness. These values provide a framework for teams to interact on an open level. Several challenges arise when the team is not aware of Scrum values.

6. Middle Management

A transition is often a time for a substantial restructuring of the organization. When people are in the status quo they are comfortable. Often, people find excuses to stay in the status quo because it is the world that they are familiar with.

"One of the biggest challenges that I have faced in my experience as a Scrum coach is not really related to Scrum as such, but more to the consequences that Scrum creates in the organization by exposing the real problems. It often happens that some people especially among managers and senior technical experts are afraid of losing their position. They start to get alarmed by the serious challenge to the status quo that the transition to Scrum brings about and do their best to slow down the change."

8. Distributed teams

Scrum works well for co-located teams, but distributed teams present a different set of challenges.

"Senior leadership often face stiff resistance in breaking the distributed team culture as managers often do not trust the remote teams completely. In addition, the distributed portion of the team is often under the control of a third party vendor with a completely different management chain. In short, the biggest challenge with distributed teams is that they often do not behave as one single team."

9. Determining Team Velocity

Velocity is the average number of story points that the team completes per sprint. Predicting velocity is a difficult task especially for newly formed teams and for teams new to a Scrum framework. There are different variables that should be taken into consideration when estimating velocity such as the newness of a team and its

composition, the political environment, the project size and complexity and the product backlog estimated in story points.

10. Micromanaging

" My experience has shown that when teams themselves decide on how much to commit to, and when this commitment is realistic and achievable, the team's focus, motivation and drive is significantly higher and they produce better results. One of the biggest challenges in successfully making the transition to self-organization is that the team will not begin to self-organize until everyone outside the team stops micromanaging them."

B.4. Scrum Coach D

According to Scrum Coach D, the 10 most important issues are:

1. Unrealistic Expectations

Scrum is usually introduced in an organization when a lot of things are not going right and it is perceived as a solution to these problems. Usually, a transition is initiated when somebody has heard about a successful Scrum adoption and wants to replicate that success. But the key is to set realistic expectations.

Unrealistic expectations are a common issue that shows up as a desire for overnight success and instant exponential productivity improvements. While these benefits will eventually come, people who demand dramatic results right away hinder progress.

"The building of a product should be envisioned less like building a cathedral and more like growing a tree. Long term schedules and detailed product perimeters will

be in constant discussion throughout the product's life. Products may also greatly change between the initial idea and the final delivery"

2. Resistance to change

Transitioning to Scrum involves a lot of change. There are some practices that need to be changed to adopt agile ones. The trouble is that, in general, people usually do not like change. They prefer the known problem to the unknown solution. The Scrum coach has to introduce change, but he also has to find the right rate for this change to take place. If the change is implemented too rapidly, the team will not be able to follow along and integrate new practices. This may also cause a burnout. If the transition is implemented in a very slow manner, the top level management is likely to raise concerns.

3. Micro-management

Scrum Boards, burn-down charts and other dashboard tools give detailed visibility to managers, but they should not use that to try to too closely control team behavior. That information is there to help teams interact with the outside world and help managers to report to the business. But neither managers nor coaches should try to use that information to micro manage the team's work.

"It is important to trust the team's commitment. Trying to force more stories into the sprint to get more story points done in an iteration is a bad practice. It can create undesirable practices such as longer workdays, less code grooming, less automated tests, less sharing of knowledge inside the team, a tendency to avoid pair programming etc."

4. Make Scrum rituals meaningful

User Stories, Test Driven Development, Retrospectives etc. are all different types of Scrum rituals. They have value if everybody understands and agrees on why they are being done. So, they should only be followed if the team understands why and agrees that it is the best way to do things.

Rituals must be understood, challenged and adapted to every team. Sometimes the best way will be to completely drop it, or do something else. Only the goal of the ritual is important not the ritual itself.

"I have often seen programmers poorly speaking of agile because for instance they never lost as much time in meetings than since we started that agile thing. If team members have this general opinion, they are likely to be right. Either the meetings are too frequent or too long, or they are not adding the expected value."

5. One person being both the Product Owner and Scrum Master

Having one person play the roles of both scrum master and product owner is a bad idea, to say the least. In that case, the person who is responsible for guarding the team and its process is also responsible for the direction and profitability of the product. This has an explicit potential for conflict of interest and the outcome depends on which of those hats is the more dominant. For example, when a scrum master assumes the role of product owner, the lack of someone pushing business priorities can result in an excessive spend on technology or a team pace that gets very comfortable. In contrast, a product owner assumes the role of Scrum Master can go too far the other way and may use the opportunity to take direct control of the development team, sacrificing the technological needs of the project and pushing for an aggressive and unsustainable pace.

"In both cases, being both product owner and scrum master is also likely to be a big draw on one person's time, leading to possible sacrifices in either their leading of the team or in having up to date information on the product."

6. Difficulty in creating empowered self organized teams and fostering team mentality

Teamwork is very vital in Scrum. The Scrum philosophy is based around self-organized teams. But, this may be a big change from the way people are used to working. Experienced programmers may not like to explain what they are doing to newer programmers and see it as a waste of time. It is important to build trust and share resources between team members. Building an actual self-empowered, self-organized team from a group of programmers is a big challenge for a scrum coach.

7. Lack of Vision

To be efficient, the team must have a shared vision for the final product. If the team does not see the larger picture it will impact the way they implement individual user stories. Sharing the Product Owner's vision with the team will help team commitment and is likely to lead to user stories with better consistency. Sharing the product vision should also help in choosing the right stories.

8. Breaking the waterfall mentality that everything needs to be sequentially completed

The idea in an agile approach is that software design emerges from code as it gets written. Previously, many development projects were built using a traditional waterfall approach. This meant that the entire system was defined, designed, developed, tested and then released in that precise order. In its purest form, a system built using waterfall techniques is not useful until the time when nearly all of its features are complete and ready for testing.

"To put it very simply: if a project has X features, a team will attempt to build all X and only then test all X at the same time without any formal quality analysis and automated testing during the construction of the features. This is a recipe for wildly missed delivery dates at best and a major disaster at worst."

9. Distributed teams

It is a nightmare for the Scrum master if the team not collocated. It may not always happen that the entire team sits in the same office. In such cases, the Scrum Master's job becomes even more difficult. It's not easy to build a team culture and ensure uniform practices when some members of the team are elsewhere. It's not ideal to try solving a team member's problem via Skype or WebEx. If the time zones are different, then it is difficult to schedule meetings.

10. Fear in developers caused by skill deficiencies

Scrum procedures such as stand up meetings require direct and constant communication and collaboration among team members. The participation of an on-site customer and the use of storyboards and whiteboards make developer shortcomings very visible to the rest of the team. For example, storyboards track the status of user stories and make a developer's lack of progress very obvious. Whiteboards can also highlight the deficiency of technical and communication skills of any one developer since they need to present their ideas in front of their peers on a regular basis. In addition, continuous integration and automated testing means that developers cannot hide poor, low quality code.

B.5. Scrum Coach E

According to Scrum Coach E, the 10 most important issues were:

1. Management or stakeholders managing the team

Scrum requires teams to be self-managing and self-organizing. Thus, neither the management nor the stakeholders should try to manage the team or assign work to them. The management style should be changed from command and control to leadership and collaboration. On the other hand, the team too should not wait for either the project manager or the team lead to delegate tasks to the team.

2. Scrum Master as a contributor

The Scrum Master's role is multifaceted. He should work to build and maintain a high performing team. This appears to be simple on the surface. He manages the daily stand-ups, collects status updates from the team and coordinates the various meetings and tasks. But actually, there is nothing simple about the Scrum Master's job. He is not the team manager or leader, but it is his job to make that the team is running at maximum effectiveness. He needs to keep the team focused and on track. He also needs to look at the big picture and find out where the team can make improvements.

3. Imposed Deadlines and Resources

Scrum teams know best what to complete in a particular Sprint, so neither the stakeholder nor the management should try to impose deadlines or prescribe resources as this would not only demotivate the team, but also reduce their productivity. This would also affect the quality of the software produced.

4. Retrospectives without improvement

Inspection and adaption lie at the very heart of Scrum. Retrospectives focus on inspecting and adapting the most valuable asset in a software organization, the team itself. Performance can be neither improved nor maintained without practice. Conducting a meeting alone is not enough to make it successful. Attention must be paid to ensuring teams plan improvements. If a plan to improve is not part of the outcome, it is not actually a sprint retrospective.

"When done well, retrospectives are often the most beneficial ceremony a team practices. When done poorly, retrospectives can be wasteful and a pain to attend. A retrospective whose action items are not acted on quickly becomes a meeting that people will not have respect for."

5. Lacking Test Automation

Due to the iterative nature of Scrum, multiple rounds of testing for a project are often needed.

"Having an automated testing framework, which takes care of both system and integration tests, adds a lot of power to a team. It not only acts as a safety net against regressions caused by new development, but also more importantly frees up a lot of valuable developer and tester time. This allows them to focus on the things they do best."

Test automation also supports continued refactoring required by iterative software development. Allowing a developer to quickly run tests to confirm that refactoring has not modified the functionality of the application may reduce the

workload. This can also increase the confidence that cleanup efforts have not introduced new defects.

6. Choosing the right Scrum Master

The selection of a new team's Scrum Master can impact the success or failure of the team's Scrum adoption. Choose the wrong person, and the team could face an uphill struggle in trying to become self organized while under the thumb of a command and control type of manager. Choose the right person - matching the skills of the new Scrum Master with the initial needs of the team and the team will have an incredible head start in adopting Scrum.

7. Compromising On Quality

Delivering a high quality, high business value, and potentially shippable product is the very basic foundation of Scrum. But, many teams tend to give up on quality due to limited resources or due to the pressure to release new features. This is one of the most common Scrum pitfalls.

8. Engineering Practices (Refactoring & Test Automation)

Due to the iterative nature of Scrum, multiple rounds of testing for a project are often needed.

"Having an automated testing framework, which takes care of both system and integration tests adds a lot of power to a team. It not only acts as a safety net against regressions caused by new development, but also more importantly frees up a lot of valuable developer and tester time. This allows them to focus on the things they do best."

Test automation also supports continued refactoring required by iterative software development. Allowing a developer to quickly run tests to confirm that refactoring has not modified the functionality of the application may reduce the workload. This can also increase the confidence that cleanup efforts have not introduced new defects.

9. Distributed Teams

In today's global economy adapting Scrum to a distributed team presents a different kind of challenge. In an ideal Scrum team, developers sit only a few feet away from the product owner and Scrum Master. The biggest challenge that distributed Scrum teams face is a breakdown in communication. Establishing strong leadership between onsite and offshore teams is also a problem. Distributed teams cannot often rely on a single scrum master due to the different time zones that the teams may be operating in.

10. Requires role changes

Scrum defines three roles: The Product Owner, The Scrum Master, and the team member. In a scrum implementation, jobs roles and job titles change. A Product Owner and a Product Manager are different. A Scrum Master is not the same thing as a Project Manager.

"The organization needs to have a plan for retraining its existing employees and getting them comfortable with their new responsibilities. People need to know what they are supposed to do, and how they can be successful."

B.6. Scrum Coach F

According to Scrum Coach F, the 10 most important issues are:

1. Mindset change

Unlike traditional software development methodologies that expect team members to follow instructions and hit deadlines, Scrum is a framework that requires each team member to be his own boss. This is an integral part of the team's decision making process. Scrum supposes that the team knows how to get the work done and more importantly how to best organize and manage the work. A lot depends on how well the team can self organize and self manage itself and this is a huge shift in mindset.

2. Keeping defects in check

The number one key to a successful Scrum project is to deliver a potentially shippable increment at the end of each sprint, where the code is tested and the technical debt is low.

"Managing defects in any software project is a challenge. People have years of learnt brain muscle memory that tells them that they should fix defects at the end after completing development. Hence, when they transition to Scrum, there is a natural tendency to fix defects at the end of a sprint or have a bug fixing sprint. Developer's brains have to be retrained."

3. Giving Estimates to Management

Estimating work that is creative and unpredictable is really hard. Choosing a way to do it is equally difficult. But, teams are often asked to give estimates for software projects up front and early. In Scrum, user stories are measured in story points.

Because people are not inherently good at estimating, most teams struggle with sizing stories.

Given below is an example that the coach described from his personal experience

"Initially, the team was asked to estimate their story points. Management then plotted the completed story points with the team's estimation. The team was then told that the differences were too wide and that the management was unhappy with the number of completed story points. They were told to increase the story points. The problem is that management used story points as a metric in this case. The team members were fearful of a punishment at the time of their bonus or appraisal."

4. Engineering Practices

Scrum processes tend to immediately speed up the software development process. But if there is no focus on working on Engineering practices such as Test Driven Development, Acceptance Test Driven Development and Pair Programming, then the team will soon end up deep in technical debt. During the course of a project, if the development team cuts corners due to a variety of reasons, then this will result in a large backlog of technical inefficiencies resulting in large technical debt.

5. The problems highlighted by Scrum need to be solved

One of the significant advantages of Scrum is that it reveals problems at quite an early stage in the development process. Knowing is only half the battle, though. The more important task is to make an effort to remove the problems that have come up. If teams just ignore the challenges that get exposed, this would result in being the cause for a Scrum failure.

"I have noticed that some teams make no effort to deal with these issues. They are either ignored or hidden until the end of the project. It is actually this dilly dallying that leads to failures to meet commitments or delays in delivery."

6. Underpowered Product Owner (PO)

An underpowered PO lacks decision-making power. There may be several causes for this such as the PO not having enough management attention, sponsorship comes from the wrong level or the wrong person, management not fully trusting the PO, PO finds it difficult to delegate decision making authority etc. As a consequence, the product owner struggles to do an effective job.

"A product owner of a new product development project I worked with, for instance, had to consult his boss for every major decision. Not surprisingly, this caused delays and eroded the team's confidence in the product owner. Ensure that the product owner is fully empowered and receives support and trust from the right person."

7. The daily stand up meeting

Time-boxing the Daily Standup Meeting to 15 minutes is an everyday challenge for most Scrum Masters. Daily Standup Meeting is conducted only to get answers to the following three questions from Scrum Team, but, often the meeting goes into discussion of small things in detail.

8. Attempting to Scale at the Start

Organizations that are adopting Agile often want to scale it right away. It is difficult to get a few teams off to a good start without complicating matters and without additional communication problems. Launching at scale can be done, but it requires a lot of coaching. It is best to start small with a pilot group.

9. Poorly executed Scrum Meetings

For a team which is new to Scrum, meetings such as the daily standup meeting, sprint planning meeting and the sprint retrospective can be a cause for concern. These meetings can quickly turn messy if they are not managed effectively. They could also turn into mentally and physically exhausting arguments without the moderation of a Scrum Master. This is one of the main reasons why Scrum falters in many teams. The team's enthusiasm for the process could wear out due to incorrectly executed Scrum meetings.

10. Development team skillsets

Part of the successful Scrum implementation relies on the development team members being generalists. They should be cross trained so as to be able to identify and execute points of convergence for the skill sets involved in the project. At the same time, they should have the advanced skills and acumen in a specific skill set that adds a high degree of quality. The team members should also have a strong sense of dedication and commitment to the project and to the Scrum principles.