1-1-2016

Implementing Agile Development at Scale: An Industry Case Study

Nikita Kataria
University of Denver

Implementing Agile Development at Scale: An Industry Case Study

Abstract

Agile software development methodologies are extremely popular. Their dynamic restructuring of the development process has been seen as the silver bullet for increasing the productivity of software development. A significant number of studies have analyzed the impact of implementing agile techniques. However these are mostly evaluated only in smaller team settings. There is very little reporting done on how agile development methods can be implemented at the team level and scaled up at the program/portfolio level in large software organizations.

We present the results of an empirical study conducted at Pearson Education. The study focuses on the penetration of agile development in the organization, agile development practices followed at the team and program level and the perception of agile development by the people in diverse roles.

The study shows that about 90% of the respondents use agile development. Of those working in agile development 13% work at the program level and 87% work at the team level. Similar to the practices at the team level, there are standard practices followed at the program level with varying rigor. Most view agile development favorably due to the benefits of agile development. Top benefits reported are improved communication between team members, quick releases and the increased flexibility to changes. Our analysis also indicates that among the population using the non-agile methods, 83% would like to switch to agile methods, while 11% of the agile users would like to switch to non-agile methods. Agile practices are followed more rigorously in larger teams. Respondents who only have experience working with agile methods practice agile techniques more rigorously and perceive it more positively. Respondents with training in agile methods are significantly more inclined to adhere to the process and have an overwhelmingly positive opinion about it. However, challenging conventional wisdom is the finding that experience does not impact the rigor or perception of agile methods. Dependencies among projects seem to have negative impact on the success at the program level due to the challenges in coordination. There is an increased need to focus on testing at the project level; however the rest of the aspects like estimation, prioritization, productivity and time tracking, reviews and continuous integration are working well at the project level. There can be an increased focus on some of the less rigorously used practices at the program level. As training seems to have a significant positive impact on the overall experience of agile development, it would help to increase the focus on training at an organizational level.

In conclusion the data indicates that there is a way of successfully scaling up agile methods from the team/project level to the program level by following a disciplined approach. Teams and programs have dependencies, so better synchronization and coordination can be achieved if the agile methods are implemented across all the teams and programs. Training resources, defining and rigorously practicing agile techniques at the program and project level and reducing dependencies are key factors in the success of scaling agile methodologies.

Document Type

Thesis

Degree Name

M.S.

Department

Computer Science

Implementing Agile Development At Scale : An Industry Case Study

---

A Thesis

Presented To

The Faculty of the Daniel Felix Ritchie School of Engineering and Computer Science

University Of Denver

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

---

by

Nikita Kataria

June 2016

Advisor: Dr. Matthew Rutherford

Author: Nikita Kataria
Title: Implementing Agile Development At Scale: An Industry Case Study
Advisor: Dr. Matthew Rutherford
Degree Date: June 2016

Abstract

Agile software development methodologies are extremely popular. Their dynamic restructuring of the development process has been seen as the silver bullet for increasing the productivity of software development. A significant number of studies have analyzed the impact of implementing agile techniques. However these are mostly evaluated only in smaller team settings. There is very little reporting done on how agile development methods can be implemented at the team level and scaled up at the program/portfolio level in large software organizations.

We present the results of an empirical study conducted at Pearson Education. The study focuses on the penetration of agile development in the organization, agile development practices followed at the team and program level and the perception of agile development by the people in diverse roles.

The study shows that about 90% of the respondents use agile development. Of those working in agile development 13% work at the program level and 87% work at the team level. Similar to the practices at the team level, there are standard practices followed at the program level with varying rigor. Most view agile development favorably due to the benefits of agile development. Top benefits reported are improved communication between team members, quick releases and the increased flexibility to changes. Our analysis also indicates that among the population using

the non-agile methods, 83% would like to switch to agile methods, while 11% of the agile users would like to switch to non-agile methods. Agile practices are followed more rigorously in larger teams. Respondents who only have experience working with agile methods practice agile techniques more rigorously and perceive it more positively. Respondents with training in agile methods are significantly more inclined to adhere to the process and have an overwhelmingly positive opinion about it. However, challenging conventional wisdom is the finding that experience does not impact the rigor or perception of agile methods. Dependencies among projects seem to have negative impact on the success at the program level due to the challenges in coordination. There is an increased need to focus on testing at the project level; however the rest of the aspects like estimation, prioritization, productivity and time tracking, reviews and continuous integration are working well at the project level. There can be an increased focus on some of the less rigorously used practices at the program level. As training seems to have a significant positive impact on the overall experience of agile development, it would help to increase the focus on training at an organizational level.

In conclusion the data indicates that there is a way of successfully scaling up agile methods from the team/project level to the program level by following a disciplined approach. Teams and programs have dependencies, so better synchronization and coordination can be achieved if the agile methods are implemented across all the teams and programs. Training resources, defining and rigorously practicing agile techniques at the program and project level and reducing dependencies are key factors in the success of scaling agile methodologies.

Acknowledgments

I wish to thank my academic advisors, Dr. Matthew Rutherford, Dr. Susanne Sherba and Dr. Cathy Durso, for their strong interest in my research and constant guidance. Dr. Matthew Rutherford's insightful suggestions and recommendations provided direction for my work. Dr. Susanne Sherba's enthusiasm about software engineering combined with Dr. Catherine Durso's deep understanding of statistics and data analysis improved the quality of my work.

I thank my industry mentors Craig Rudman and Marianne Molberg for providing me with this great opportunity to work with the resources at Pearson and perform research with real industry data.

I wish to thank my family, my in-laws for being caring and appreciative, my Dad and Mom for their constant encouragement throughout the process and strengthening my belief in myself. Thanks to my motivation, my little sister for encouraging me to hold the vision for success in life, constantly achieve higher goals and trust the process. Lastly, I would like to acknowledge my loving husband. Without his unflagging support, strong conviction and constant reassurance, none of this would have ever been possible.

Table of Contents

List of Figures

Chapter 1

Introduction

The issues of improving productivity in software development, reducing waste and delivering faster, cheaper and more efficient solutions are constantly being dealt with in numerous ways. In this process of constant improvement, the invention of agile software development was a tremendous breakthrough. However, this process is often applied only to smaller teams. Large companies with massive teams find it hard to transform to agile development. Thus, there has been a need to scale agile techniques for larger teams and organizations. Today there are several forms in which these methods are being implemented in larger organizations.

However there are very few studies performed to analyze how the scaled implementation of agile techniques impacts the productivity of the organization, what are the benefits and challenges of these software development processes and the perception of these methods in large organizations. Also there is a need to analyze how agile development is perceived by the people using it.

The analysis of the full impact of implementing agile development methods in a larger organization can be of immense value to the entire software development

industry. Hence this study seeks to systematically survey an organization, Pearson Education, perform an empirical study and present the findings of the study to draw conclusions. We report the benefits and limitations of agile development as well as suggest potential improvements to maximize the efficiency and productivity. We can gain great insights through interaction with the organization employees about practices followed at different levels, their perceptions of the development processes and the impact on productivity in general.

We conduct a survey across Pearson Education's Higher Education Division sent out to employees in development, testing, management and related roles. We interact with employees directly involved in the production of software. Our questions aim to understand respondent demographics, agile methodology usage, rigor of agile practices and respondents perceptions of why agile development works well or poorly for them. From these responses, we hope to gain insight on how agile methods are scaled at Pearson Education.

Historically, Pearson was a publishing company. Keeping up with the changing technology, it entered the E-Learning industry segment and today it is one of the world's leading E-learning companies. They produce software solutions for the education sector. The company has a global presence with development centers all across the world. Pearson has largely adapted their processes to agile development methodology. They implement agile methods at scale using the Product Creation Framework [1] which is based on Scaled Agile Framework(SAFe) model [3].

Even though the organization had a positive experience from the implementation of agile development, the full impact of transforming to an agile enterprise, the benefits and shortcomings of this adoption and the perception of their employees towards it are yet to be evaluated systematically. Thus, the impact analysis of

implementing agile techniques at scale in a large software development organization provides valuable findings in a world with little research in this area.

1.1 Problem Statement

In this thesis we seek to analyze the penetration of agile methods across Pearson Education and how agile practices are implemented at the program and project level. We want to understand the perception of the organization towards the implementation of agile methods and analyze the impact of various factors on the rigor of practicing agile techniques.

1.2 Method of Approach

To address our research objective we conduct a survey asking questions about demographic information, practices followed, and the perception of these practices. This data is analyzed using inferential and exploratory analysis methods. We identify trends in the data to understand the impact of the methodology and provide business insights. We analyze:

- demographic information to understand the background of the respondents
- data about the agile methods and techniques to understand the penetration of agile methods and which agile methods are practiced more than others.
- impact of various factors like experience, non-agile work experience, training and team size on the perception of agile development and the rigor with which the practices are followed

1.3 Research Contributions

Through our analysis we find that the organization overall has a positive experience with agile methods. Among the few projects which use a non-agile approach, most would like to transform to agile methods. We perform analysis to see if these results can be generalised across the organization and do not overrepresent employees who feel strongly about agile development. Results of this analysis indicate that the survey data is generalizable to the organization. Results show that about 90% of the organization is using agile methods. Scrum and Kanban are the only two methods practiced and Scrum is by far more popular than Kanban. Team sizes vary however the average team size is 11. Teams are heavily dependent on other teams in the same program and better dependency management can improve productivity. The rigor of practices does vary, with larger teams practicing more rigorously. However experience does not impact the rigor, as experienced people also strongly believe in agile methods due to its positive impact on productivity. There is organization-wide positive perception about agile development with project level implementation being more positive than program level. Training has a positive impact on perception and makes agile development work better. Hence there is an increased need for training on agile techniques across the organization. Exposure to other methodologies of development (non-agile) have significant impact on the rigor and perception. Respondents who have worked only in agile development have a more positive experience and rigorous approach with agile practices. Experience has no significant impact on the perception of agile development methodology.

1.4 Thesis Outline

The 'Literature Review' chapter summarises the state of current research in the field of software development and more importantly focuses on agile software development research. The 'Background' chapter explains different software development methodologies, fundamental concepts of agile development, artifacts maintained, processes and the terminology used in agile development across the industry and specific Pearson nomenclature. In the 'Findings and Results' chapter we present the findings of our exploratory and inferential analysis of the data and finally present the conclusions and discuss future work in the 'Conclusions' chapter.

Chapter 2

Literature Review

Agile software development methodologies [4] have become popular in mainstream software developers since the late 1990s. There are several methods like Scrum [5], Crystal [6], Extreme Programming [7] and others used to implement agile development. Our research aims to systematically review the usage and perception of agile practices at scale. For effective research we need to understand the current research in agile development and associated issues. Therefore we perform a survey of the published work in this area.

[8], [9], [10], [11] and [12] present a comparison of analysis of traditional and modern methodologies. These in general show that there are different methods suitable for different development environments. However modern development methodologies are more dynamic in nature. Hence they are well suited to cater to the scenarios of constantly changing/evolving requirements. Gaurav and Prateek [13] analyse the impact of agile development methodology on software development based on quality. This paper analyzes values and principles of ten agile practices becoming popular in software development. Agile processes can also have some challenges thus, not

necessarily proving to be beneficial. [14] presents the benefits and challenges of agile development processes. The study [15] performs analysis on agile software development methods from the view of supply chain management, concluding the overall positive impact of agile software development. [16] analyzes metrics collection methods for measuring productivity, estimation and quality for an agile development organization using Extreme Programming (XP) method of agile development. In [17] William investigates some of the XP [7] practices with an IBM project group. It is found that the product developed using these methods has improved quality in comparison to other methods. Teams using XP development witnessed surges in productivity, positive team spirit and satisfied customers. Melnik and Maurer [18] analyze the development of a web based system. There were nine members on the team using Extreme Programming techniques. They also observed significant gains in productivity. Our work is closely related to the work by Melnik and Maurer who analyze the perception of students towards agile development. They collected data over three years. The students were positive towards using Extreme Programming. However, even though students indicate productivity increase and improved quality using this method of agile development, they could not compare it against other development methods and this could skew the results in favor of the method. In [19] Carver discusses using students as subjects in empirical studies. They conclude that case studies done with students as subjects are not sufficient experimenting methods in the industry. Hence from industry perspective there is little evidence on the applicability and positive impact of agile practices. Sharp and Robinson in [20] present a study of the overall human interaction and cultural impact of Extreme Programming practices in a small company. Others papers [21] address individual practice of pair-programming and [22] test-driven development.

In [23] Tsun Chow and Dac-Buu Cao report on the critical success factors for agile development. This research uses data from a survey to explore the critical success factors of agile development using quantitative methods. Data was collected from 109 projects from various organizations using agile development. Statistical analysis was performed and conclusions were that there are very few critical success factors. Of 48 total factors considered only 10 factors are statistically significant. Multiple regression analysis proves that only correct delivery strategy, rigorous practising of agile techniques and high-caliber team are critical to project success. Three other factors that can be critical are good project management, an agile-friendly team environment and intense involvement of customer. This research provides a short list of factors management can focus on to adopt agile development methods in their development projects. We have used a similar data collection approach. However this research aims to provide us with the condensed list of the critical factors to be considered in order to implement agile development successfully, while we intend to analyze which practices are followed at the team and program level, how widely agile techniques are used and what the perception of the people using it towards it is.

The paper [24] is an analysis of the impact of the adoption of Scrum on customer satisfaction. Data is collected performing a survey in 19 development projects with 156 software developers. The findings show that there is no evidence of achieving customer satisfaction and increasing the possibility of success in software projects by this adoption. Each dependent variable is compared to provide a thorough analysis. The statistical Mann-Whitney hypothesis test popularly called a U-test reports no significant differences between the groups. This research provides conceptual knowledge and an understanding of certain myths promoted by agile development advocates. It also uses an empirical method for evaluation, performing exploratory

as well as inferential analysis methods to confirm the impact of Scrum on Customer Satisfaction. As we are evaluating the impact of agile techniques in an organization using empirical methods, this research is similar to our work and provides a direction to proceed. However this research is addressing only the impact of Scrum on customer satisfaction, while we intend to analyze which practices are followed at the team and program level in Scrum and other agile methods, how they impact the productivity in the organization and how these methods are perceived by the people using them.

[25] reports on studies done in the agile development area. They select 36 of 1996 research studies based on good rigour, relevance and credibility. 33 of these studies are primary studies and the remaining are secondary studies. This paper can be used as a map of findings according to topic, which can be used to find further relevant studies and compare the development scenarios. The major finding is that there needs to be more good quality research in agile development for concrete conclusive recommendation of agile method adoption.

In [26] Moniruzzaman and Hossain find the major improvements by agile software development in meeting the changing business environments. A comparative study of agile development methodology and traditional development methodology is done. This paper confirms that the iterative incremental model of agile methodology is more effective than the traditional approach. This provides background knowledge about the potential benefits and improvements of agile methodology implementation.

Some small organizations use models to guide management and deployment to improve the software process improvement (SPI). However there are issues associated with existing models. Hence in [27], the researchers propose a new process, with appropriate strategies based on the organization size to incorporate improvements with

techniques of Scrum. They also apply the process in two small companies. Initial results suggest that they are suitable for small organizations. This paper suggests a lightweight model using the Scrum methods by introducing some modifications in the base process specialized for the company size.

In [28] the researchers track the changing perception of agile development at Microsoft using a survey. They intend to provide conclusive opinion about if agile development is the "silver bullet" in software development. The data is collected from five surveys from 2006 to 2012 with a total of 1969 respondents. The results show that even though there is immense market pressure, agile development adoption at Microsoft is very slow. There has not been a strong growth trend in any practice. The results show that both agile and non-agile development users agree on the benefits and challenges of agile development techniques. Non-agile users are more strongly agree with the problem areas than benefits. Scaling agile practices is the biggest challenge limiting its adoption. This study in an organization over a six year period aggregates data to find growth trends in agile adoption and practices to confirm if it is actually a silver bullet in software development.

In [29] Laanti reports that scaling agile methods is the challenge faced by organizations even though they wish to adopt agile development methods. Deploying agile development methods at the team level is insufficient as they have dependencies with other teams and synchronization is even more difficult to achieve if these teams sharing dependencies deploy and operate on different schedules. They propose a framework for scaling agile techniques to the program level. There is no measured evidence as the programs have not yet completed, however qualitative data shows the superiority of this new framework. Anecdotal evidence from employees operating at the program level is positive. Over 60 programs of varying sizes are implement-

ing this model. This paper presents one approach for deploying agile techniques in scaled software development environments. This work is similar to ours in the sense of implementation of agile methods at the program level; however while we attempt to analyze the impact of the established practices to implement agile development at scale this paper presents an altogether new set of practices to scale agile development up to the program level.

In [30] Pichler, Rumetshofer and Wahler present the challenges faced while working on a software development project for a period of three years. They focus on the requirements engineering process. The project uses agile development techniques for requirements elicitation for development while the client uses traditional software development processes. Recommendations of this study are demonstrating objectives and applicability of agile development techniques to the client and highlighting the difference between prototypes and final product to the client. Thus, this paper presents ways of working in asynchronous environments using the recommendations provided by the researchers to deal with the challenge of co-ordinating between agile development teams and traditional customers.

In [31] Moe, Dingsøyr and Dyba provide a better understanding of the nature of self-managing agile teams and the challenges that arise with such self organizing teams. Researchers did field work for nine months in a software development company. This company adopted Scrum by focusing on the cultural impact and analyzed how teamwork is perceived by the team members. The conclusion is that self-managing teams needs a change of mindset from management along with development. Even though this takes time it increases the trust in the team. The scope of their research is to analyze the self organizing nature of teams while we explore all the other aspects of agile methodologies to a limited extent.

In [32] Bagel and Nagappan present the penetration, usage and success of agile methodologies in a large software development organization Microsoft. They conduct a web based survey to collect the data and find that one-third of the respondents use agile techniques. Overall there is a positive perception of agile development. Top benefits reported are better communication, flexibility and quicker releases. The Scrum method of agile development is most popular at Microsoft. Developers face challenges in scaling agile techniques to larger projects, co-ordinating between agile and non-agile teams and having too many meetings. Respondents mean work experience is 9.2 years in software development. Of 14 different agile practices, 60 percent of respondents use over 12. This Microsoft research study analyzes the implementation of agile development in a large scale organization is similar to our study. However the scope of their investigation is limited to project level implementation while we explore the implementation at the program level as well some analysis about the implementation of non-agile methods.

Several organizations wish to adopt agile development for the advantages potential quicker return on investment, customer satisfaction and improved quality. To provide a systematic way to adopt these methods Sidky, Arthur and Bohner [33] present the Agile Adoption Framework which is an innovative approach to implementing agile methods. The framework has two components: an agile development measurement index and a four-stage process to guide adoption of agile methodology in an organization. There are five defined agile development levels used to identify the extent of agile techniques that can be implemented based on the project in consideration. The four-stage process determines if the organization can adopt agile development and with which set of agile practices. To evaluate this framework, various members of the agile development community are presented with this frame-

work and their responses are mostly positive. This research presents a model for the adoption of agile development and helps understand how the transition between traditional to agile methods can be done. As we study the responses from the groups using agile development as well non-agile development groups it is helpful to understand ways of this transition. However this is different from our subject of research that examines the current state of implementation to provide business insights to management.

In [34] Kovitz reports on skills for different styles of requirements engineering. There are 4 main focuses: advanced prioritization, requirements engineer guiding the customer to find the problems early before major design decisions, importance of negotiation for features that are not realistically accommodable in the given conditions and creating a requirements document to determine components and subcomponents needed to be build to calculate development time and resources for the project. This paper focuses on the success of development by emphasizing certain factors during requirements gathering while we study all the techniques applied at the project and program levels in order to ensure success in delivery.

In [35] Qumer and Henderson-Sellers develop an analytical framework '4-DAT' and apply it to six agile methods. For comparison they also apply it to two traditional development methods. Results show that it can be determined whether agile methods are applicable for that project based on the degree of agility found. The agile methods used are Scrum, Feature Driven development, Extreme Programming, Dynamic development, Adaptive development and Crystal Programming. These methods are evaluated from four perspectives at the process level and practice level. While this research study provides us with a model for evaluating which methods can be best applied to the process and which degree of agility can be found in this

method, our work focuses on a systematic evaluation of the method already applied in an agile enterprise. Qumer and Henderson's work does provide a good foundational understanding of the construction of a formal model.

In a summary of a panel to discuss the scaling agile techniques, [36] identifies the top challenges in scaling agile methods: reconciling agile methods with traditional practices, generating guidelines for non-sweet spot agile projects, augmenting agile practices for large projects, resolving issues with integration in agile projects, scaling agile techniques across several applications in an enterprise, handling non-collocated agile development projects and integration testing for bigger systems. The suggested ways to resolve these issues are shorter sprints, improving communications in large projects, architectural planning before starting the sprints, intense collaboration with onsite customers, packaging components and conservative expectations for change from large projects. This paper provides insight regarding some issues and potential solution guidelines in the process of scaling agile methods. While these generally discuss the problems foreseen and experienced in scaling, we look at the issues of a specific organization which is implementing agile techniques at scale.

It is a general observation by researchers that quality, productivity and staffing needs determine the major costs in software development. In [37], Erdogmus proposes a cost effectiveness indicator combining the cost drivers using an economic criterion. If the cost effectiveness indicator increases for a project, a lower unit value is required to break even and project profitability increases. The break even point is an aggregate economic indicator for software development as the multi-criteria comparisons on productivity, quality, productivity and staffing metrics are combined in single criterion of cost effectiveness. Availability of base measures, ability to accurately capture them and dependency on the output measures limits this indicators

applicability and portability. This research derives an indicator for cost effectiveness for software development as costs are a significant parameter to be considered in optimising the process. We get significant understanding about the cost effectiveness calculation parameters for software development from this study as implementing agile development heavily focuses on reducing the project cost by decreasing time to market, increasing efficiency and quality of product.

In [38] Qumer and Henderson-Sellers propose a new framework to support the adoption, evaluation and improvement of agile development methods in practice. To face the challenges typically found in quickly adopting agile methods, the researchers provide a number of approaches to assist in this transition. The Agile Software Solution Framework gives a context for exploring agile methods. It contains an agile toolkit for quantifying part of the agile process by linking the business aspect of software development to ensure that the agile process and business value are well aligned. They describe how to apply these theories in practice using the agile adoption and improvement model in two companies and performing case studies. While our research focuses on the evaluation of the current state of agile implementation in this organization and differs significantly from their work, it provides us with an understanding of frameworks for systematically transitioning to agile development.

The challenges of migrating to agile development have been investigated in the paper [39] and the challenges that management faces are presented in [40].

Chapter 3

Background

3.1 Software Development methodologies

There are various different traditional and modern software development methodologies. Waterfall development, prototyping, spiral development,iterative and incremental development, rapid application development, agile development are the most popular software development methodologies [41]. Waterfall development and agile development are the only two development models seen at Pearson Education.

The waterfall model [42] is one of the most widely used model in the software development industry. In this model, the project is divided in phases including requirements gathering, design, implementation, testing and deployment. There is some overlap and iteration between phases with emphasis on planning, time scheduling, budgets and implementation of an entire system [41]. There is elaborate written documentation, formal reviews and a user acceptance process. The issues faced with this development model are that product is delivered a couple years after the requirements gathering and is outdated for the current market requirements. Thus,

business software is late, over budget and not able to fulfill the dynamically changing requirements.

Agile development is a software development methodology essentially comprised of iterations. The customer and developers agree on a list of tasks for each iteration. Thus, changing requirements can be accommodated. Solutions are developed incrementally by the development teams in close collaboration with the customer over sprints. There are frequent releases. Continuous feedback successively refines and finally delivers a complete software system. Agile software development is a lighter and more people-centric approach in comparison with traditional approaches.

3.2 Agile Manifesto and types of Agile methods

3.2.1 Agile Manifesto:

A manifesto was designed in March 2001 by 17 experts in software development processes and associated issues of software development to deal with the issues of traditional development. The Agile Manifesto principles are mentioned below [43]:

- Customer satisfaction through continuous delivery
- Welcome changing requirements in development for customers competitive advantage
- Deliver working software frequently
- Collaboration between development and business throughout the project
- Motivated individuals in an inclusive, supportive environment

- Effective communication within a development team - face-to-face conversations
- Working software used as the measure of progress
- Sustainable development - The product owners and developers should be able to maintain a constant pace throughout the project
- Attention to technical excellence and good design throughout the project
- Simplicity - the art of maximizing the amount of work not done
- Self-organizing teams
- The team reflects improvising the process further at regular intervals

Agile Manifesto principles are summarized in [43] as:
Individuals and interactions over processes and tools
Working software over comprehensive documentations
Customer collaboration over contract negotiations
Responding to change over following a plan

3.2.2 Agile methods:

Agile development is simple and delivers software in quicker time frames by delivering software in short cycles, getting feedback and responding to that feedback [44]. According to [44] an agile method has the following properties:
Incremental: Frequent release of software
Cooperative: Collaboration between developers and customer
Straightforward: Easy to learn and simple

Adaptive: Able to accommodate changing and new requirements at various development stages

Industry research shows that agile development has a positive impact on various software development aspects like project visibility, productivity and software quality. [45]. Agile software development methods have minimal documentation and use prototyping and iterative development. According to [46], agile methods can accommodate changing requirements and support continuously delivery with close interaction between customer and development. In [47], Miller proposes the characteristics of "modularity, iterations, parsimony in development process, adaptive, incremental, convergent, people oriented and collaborative" in the agile development process.

Some agile methods are Scrum, Extreme Programming, Kanban and the Rational Unified Process. As Scrum and Kanban methods are used by our respondents, we discuss these methods in the following sections:

3.2.3 Scrum:

The main purpose of Scrum is providing a way to accomplish dynamic requirements gathering, iterative cycles for implementation and thorough testing in smaller chunks. The development process allows responding to the changing requirements. Thus, the development process can deliver a market-relevant product.

SCRUM Phases:

The Scrum development method has phases of planning and design, development and closure [48].

Pregame Phase: The pregame phase is mainly divided into planning and architecture high-level design. The planning phase includes the overall product definition and a product backlog list of the main overall current requirements is created. Prioritization and estimation of effort for the implementation of the product backlog is done. There is estimation of the delivery date and functionality of the release. The formation of the team, tools to be used, various other resources and funding from management are done in this phase.

Product architecture/high-level design is done based on the product backlog. The product backlog is reviewed and modifications are made to refine the system architecture. This phase also identifies the possible issues during implementation and redesigns or redefines backlog based on that.

Development Phase: In this phase there are development releases. The development work in Scrum is done in cycles. It is iterative. There are fixed length sprints usually of two weeks; however, these can be anywhere between one to four weeks. Each sprint includes review of the previous sprint, prioritization and estimation, implementation, testing and delivery. The development team, customer and product management participate in the review. During the review the focus is on finding what went well and what did not in the last sprint. Estimations and actual time spent on tasks are compared to determine the accuracy of estimation and productivity of team. During this development phase, management tracks the development time along with the progress in functionality and the quality of work. Multiple sprints occur before the final product delivery.

Closure: The closure phase occurs after all the requirements are implemented and there are no new product increments thereafter. The product is ready for release and involves integration, system testing and documentation and final deployment.

The Scrum Teams Roles and Responsibilities

According to [48], Scrum teams have the capability to complete their product backlog. These teams work on the product in every sprint and incrementally deliver the product. Scrum is designed for flexibility, creativity and productivity [48]. The Scrum team is made up of the product owner, Scrum master and the development team.

The product owner [49] increases the product value by collaborating with the development teams. The product manager manages the user stories in the product backlog and also their priority. The product owner ensures that the development team clearly understands the product backlog [50]. The product owner is a single person; any changes in the product backlog have to done by running them by the product owner. The product owner has the final decision-making authority.

The development team [49], [50] consists of developers and testers. This team delivers releasable increments of software components at the end of each sprint. The development team is self-organized and does the actual implementation of the product backlog items. The development team also does the estimations for the user stories, creates the sprint backlog and reviews the product backlog list.

The Scrum master's [50] primary responsibility is to remove obstacles on a daily basis for the agile team. He/she is also responsible for checking that the project adheres to the practices of Scrum. The Scrum master acts as a nexus between the product owner, the development team and management. The Scrum master facilitates the smooth functioning of the entire team and sprints by better management of product backlog and barrier removal.

Scrum Terminology

The following table presents a list of common Scrum terms [51]:

Activity	Description
Daily Scrum	Short daily meeting held to check the status of every team member's tasks and remove barriers
Done	In a sprint review a task is reported as 'Done' if everyone agrees mutually that the task is completed according to the guidelines and standards that the team adheres to.
Increment	A shippable product with partial functionality to be delivered to the product owner stakeholders at the end of the sprint.
Sprint	Sprint is an iteration in the agile development methodology. The duration of the sprint is about two weeks. It produces an increment of the product.
Product Backlog	List of requirements with assigned priority and allocated time for completion. These are requirements expressed as user stories that can be usually implemented over a sprint.
Sprint Backlog	A list of tasks to be completed in a sprint. Each task has an estimated time.
Sprint Planning Meeting	This meeting is held before the sprint for planning and estimations of the next sprint. The product owner presents the priority of the product backlog and the team does the estimation for the items in the backlog.

Sprint Retrospective Meeting	The Scrum master facilitates this meeting which is held at the end of the sprint to make decisions about what should be removed and changed to increase the productivity in the next sprint.
Sprint Review Meeting	This meeting is held between the development team, the product owner and stakeholders at the end of each sprint. There are discussions about the completed sprint and the next sprint.
Stakeholder	A stakeholder is anyone affected by the project.
Velocity	At the end of each iteration, the team adds up effort estimates associated with user stories that were completed during that iteration. This total is called velocity.
Burndown Chart	This is a graphical representation of work remaining on the vertical axis versus time along the horizontal.

Table 3.1: Scrum terminology.

3.2.4 Kanban

Kanban is a framework used to implement agile development methods.

Kanban Principles

As explained in [52], Kanban is very simple. The key Kanban principles are:

- Visualize the workflow

- Keep improving flow or Kaizen
- Limit work in process

A Kanban team[52] essentially functions by focusing on the task which is actively in progress. After this work item is completed, the next work item with the highest priority is 'plucked' from the product backlog. The product owner can re-prioritize work in the backlog as any changes apart from the current work item do not impact the team. As long as the highest priority work items are on top of the backlog, the development team delivers maximum value to the business. There are no fixed-length iterations; these are not required because the next work items are plucked from the backlog once the in-progress work items are completed.

Kanban teams uses Kanban boards and cards to represent the work and workflow. There is a whiteboard with sticky notes. This can be done virtually using software as well. This helps in visualizing the work and helps the team watch how the work item is moving across the board. This is called observing the work-flow. A limit of how many work items can be in progress at any moment is set. When there are barriers preventing work item completion, they are displayed on the board and team members collaboratively resolve these issues and finish the work item. There can be a regular deployment cadence or continuous delivery. The cycle of planning, estimating, development, testing and release is done through every work item when the work item is the highest priority item. Items ready to be delivered are released as per the deployment cadence.

Key Features of Kanban:

- Flexibility in planning: The team plucks the work item with the highest priority from the product backlog after completing the current work items. Thus, the product owner can change the backlog without affecting the team hence enabling accommodation of changes in requirements.

- Minimizing cycle time: Cycle time is the time taken for a work item to flow across the board, i.e. from the time it is started on until the time it ships. As there are limited work items in progress and the entire teams ensures the work items are moving smoothly through the process, the cycle time is minimized. A control chart used in Kanban shows the cycle time for each each work item and rolling average for the team.

- Efficiency through focus: As there are limited work items in progress, the team can focus better on these work items. Multitasking often hampers efficiency due to reduced focus. If there are many work items in progress at a given time, there is more context switching. This hinders their path to completion.

- Moving towards continuous delivery: The quality of code can be maintained by building code incrementally and validating it throughout the project life cycle. These tested code fragments can be released continuously to customers (weekly/ daily/ hourly etc). Kanban supports continuous delivery as it focuses on just-in-time delivery of value to customers and optimizing the flow of work.

3.2.5 Scrum vs Kanban

Kanban and Scrum are both different frameworks for implementing agile development methods. Even though they are similar, they are different approaches. Some teams at Pearson Education combine the idea of Kanban and Scrum into 'Scrumban'. For instance fixed length sprints from Scrum are combined with the focus limiting the work in progress from Kanban to effectively customize it to the teams requirements.

3.3 Scaling agile methods

As we see in our review, agile methods are designed for smaller teams. Thus, large scale implementations of agile methods need some modifications. There are frameworks used in large agile development projects for scaling agile practices. Pearson Education uses the Product Creation Framework which is based on the Scaled Agile Framework (SAFe) [3] for implementing agile practices at the program and portfolio level.

3.3.1 Product Creation Framework:

The Product Creation Framework (PCF) is based on agile development principles. PCF principles help the teams focus on delivering the highest value to the customer. PCF blends the agile development practices with Pearson Education's core principles.

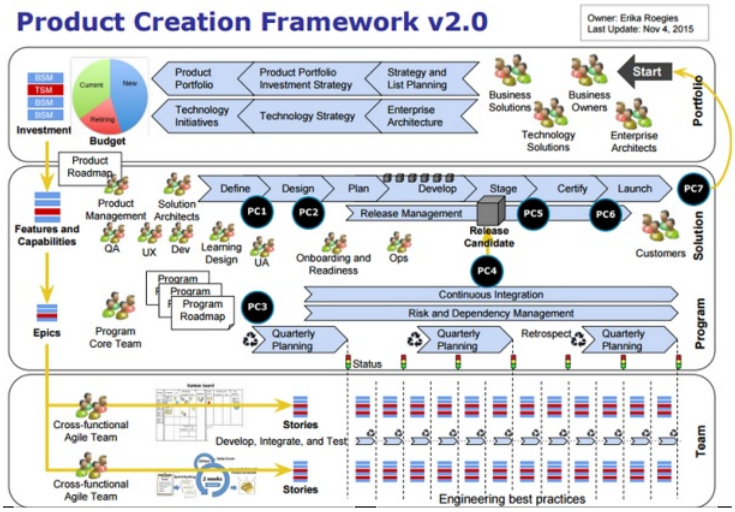

Figure 3.1: Product Creation Framework from [1]

3.3.2 Product Creation Framework values

Pearson Education's Product Creation Framework document [1] lists the following values:

Focused on the Customer : Deliver features and functionality early and throughout the process that provide value to the customer.

Continuous improvement : Implement small, incremental changes and streamlined work flows to improve quality and efficiency of products and services.

Optimize for the whole (design thinking) : Organize self-sufficient teams that are complete, multi-disciplined, and co-located who can complete delivery end to end.

Transparency and visibility : Define a visible process to the stakeholders.

Build quality in and eliminate waste : Ensure quality is considered

early and throughout the process and identify opportunities that cause waste.

3.3.3 PCF Milestones:

Milestones in PCF [1] are depicted by black circles in the figure below

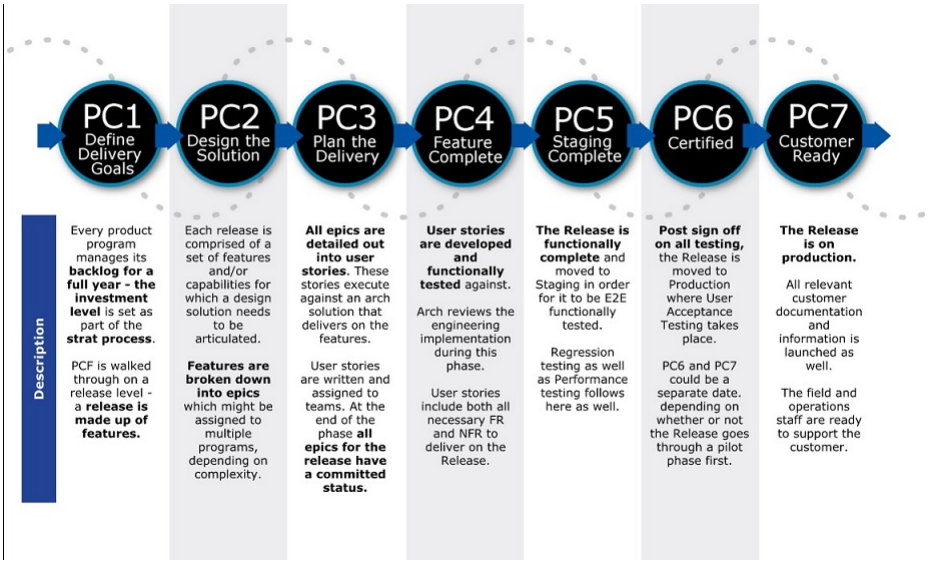

Figure 3.2: PCF milestones from [1]

Pearson Education's Product Creation Framework document [1] lists the following milestones:

Release Planning

Milestone 1 - Defining the delivery goals

Milestone 2 - Designing the optimum solution

Quarterly Planning

Milestone 3 - Detail planning of the delivery

Milestone 4 - Releasable complete features

Milestone 5 - Certifiable release components

Milestone 6 - Releasable components are ready for the customer

Milestone 7 - Customer ready solution delivered to the customer

3.3.4 The structural hierarchy in scaling agile methods:

In order to scale agile methods to large teams there is a structure of teams, programs and portfolios[3]. There is division of tasks at each level. At the portfolio level there are investment themes, at the programs level there are features and epics, and at the team level there are user stories. Following are brief descriptions of the terminology used in PCF for implementing agile methods at scale:

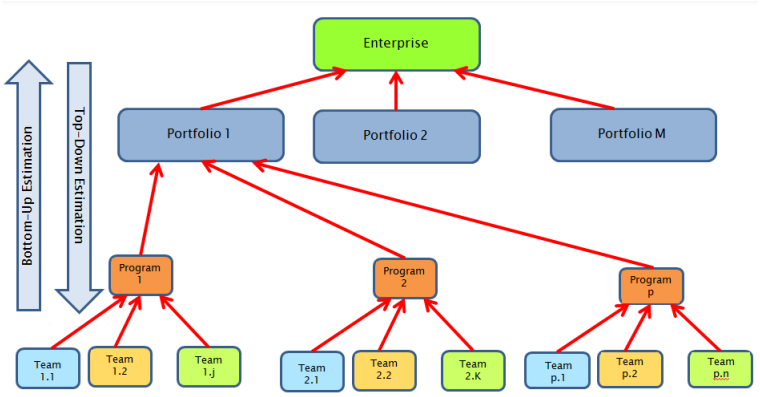

Figure 3.3: Structural hierarchy for agile implementation in large organizations from [2]

Teams: The teams[3] are comprised of developers, testers, scrum masters and product owners. Teams consist of approximately five to nine people. The team backlog consists of user stories.

Programs: Multiple teams form a program [3]. Programs have approximately five to fifteen teams. Programs could be integrating components from different teams to form an entire product or they could be developing major functionality to be used across different products. Features and epics are defined at this level.

Portfolio: A portfolio [3] is composed of multiple programs and strategic decisions are made by the people working at this level. Investment themes are defined at this level. These are then decomposed into features and further into epics at the program level.

User Stories: User stories [3] are the smallest units of work. They deliver a particular value to the customer. User stories are a few sentences written in simple language by the product owner. Later the team collectively writes more detailed requirements.

Epics: Epics [53] are significantly larger units of work. These are development components which are further decomposed into user stories. User stories can be completed in biweekly sprints. Epics are usually delivered over a set of sprints. As a team learns more about an epic through development and customer feedback, user stories are added to the teams backlog. An epic burndown chart helps visualize epics. This keeps stakeholders informed about how the team is progressing and facilitates open conversation about the evolution of the product and completion.

Features and Capabilities[53]: The themes are decomposed into features and capabilities which are deliverable functionalities. These features are delivered in the quarterly releases and they provide specific features of the products and services. They are further decomposed into epics.

Investment Themes [53]: These are strategic decisions defined at the portfolio level. Products are developed based on the budget, market requirements and several other factors by the stakeholders. These are the semi-annual decisions determining the work flow for the organization. Investment themes could be a functional goal like remodelling a product or non-functional goal such as migrating from Windows

to Linux based servers. The figure below [2] depicts the structural hierarchy of scaled agile development organizations.

Chapter 4

Research Approach

We gather data through an anonymous survey about the demographic , methods, practices, perception, benefits and challenges of the development methodology used. We compile and analyze this data to completely understand implementation of agile development at Pearson Education and also discuss potential improvements in the process.

4.1 Survey Population

The survey was sent out to 2065 employees of Pearson Education working in the Higher Education division. The employees are chosen from the entire division after carefully considering the roles and functions in which agile implementation is relevant. The survey was sent to functions like development engineering, testing, management etc. where the agile techniques can be applied. Among the 2065 em-

ployees that this survey was sent out to, about 205 responded to the survey. The respondents are from diverse roles.

The goal of this survey is to understand development methodologies used by software engineering teams at Pearson Education, the state of agile development implementation and the extent and diversity of agile development practices at Pearson. The respondents are asked questions regarding demographics, technical practices in the project and their perception of the methods. This survey is anonymous and voluntary and takes approximately 10 minutes to complete. Most of the answer are multiple choice with options for free form responses.

4.2 Survey Questionnaire

The questionnaire has several questions divided into four sections. The first section is the demographic section. In this section the respondents are asked about their

- Work area
- Role
- Work location
- Experience
- Pearson experience
- Development methodology used by the respondents team

The second section of the questionnaire is different depending on the response to the last question about development methodology. We ask respondents using non-agile methods regarding:

- Methodology followed by the team/program
- Has the project always been using non-agile development methodology
- Have they used agile methods in past, if yes why using non-agile development methodology now
- Advantages
- Disadvantages
- Would they like to switch to agile development

We ask respondents using agile methods regarding:

- Which agile methodology is used by their project/program
- How long project has been using agile development methodology
- Did they work in non-agile development in the past
- Training in agile techniques
- Level at which they work (project / program)

We ask respondents questions based on the level at which they work. For the program level respondents we ask regarding:

- Type of program

- Size of Program
- Do all their teams use agile methodology
- How rigorously the program practices were followed
- If they have program demos
- If they have regular program release
- If they have program core team meetings and if yes how often
- Who had the program content authority

For the respondents working at the project level we ask regarding:

- Size
- Locations
- Collocation
- Agile ceremonies used and how rigorously
- Engineering practices used and how rigorously
- Modifications for scaling if any the teams are independent

The following questions regarding the perception of agile development are asked to all respondents using agile development regardless of their level:

- Perception of agile development based on multiple factors (architecture, collaboration etc.)

- How well is agile development working for them at different levels (team level, individual level, group level etc.)
- Benefits of agile development (listed as well as an option for free form responses)
- Challenges of agile development (listed as well as an option for free form responses)
- Main benefit
- Main challenge
- If they would like to switch to non-agile methods
- Overall suggestions for improvements if any

4.3 Pretesting

In order to pretest the survey we conduct a pilot survey. This survey was sent out to about 8 respondents from various backgrounds.

- 2 Professors working in the software engineering area of computer science
- 1 Professor working on data analysis and visualization
- 1 Agile coach
- 1 Software developer
- 1 Program manager
- 3 Students from computer science with previous work experience

The pretest results are utilized to improve the questionnaire by testing the questions on the basis of relevance to the objective of research, understandability of questions, applicability parameters and if the responses can be used for evaluation in order to derive conclusions from the data.

4.4 Data Analysis methods

Our data analysis primarily involved two major steps:

Data Preparation - Cleaning and organizing/formatting the data. Data preparation, transferring the data into readable formats, checking the data for accuracy by observing the values for the various responses and plotting scatter plots and checking for unreasonable outliers, cleaning the data using automated scripts written in R. For various fields like the Role, the respondents gave a wide variety of responses apart from the options given. Thus, closely examining these responses and fitting them into predefined categories as far as possible and creating new categories where required.

After the data is cleaned, data sets are created for non-agile methodology respondents, agile methodology respondents, team level agile respondents, program level agile respondents, team practices, perception etc.

Descriptive Statistics : We use descriptive statistics to describe the features of the data. We create graphical displays of various response categories to present the findings using the R programming language and spreadsheets. Thus, provide summaries about the sample and the measures. Together with graphical analysis, we provide the visualisation of quantitative analysis of data describing the findings from the data.

Inferential Statistics : We use Fisher's Exact test, Kolmogorov Smirnov test and linear regression in order to perform inferential statistics on the data sets divided into groups based on various parameters like experience, team size, employee level etc.

The results of the exploratory and inferential analysis are used to draw conclusions from the data which are described in the next chapter.

Chapter 5

Findings and Results

In this section, we report on the findings from the survey. We report on the respondent's demographics, the extent of agile adoption and the perceptions of agile software development techniques.

5.1 Demographics

The survey was sent to a population of 2065 employees. We received 205 responses. This section presents demographic information about the respondent population.

5.1.1 Work area

The respondents work in different work areas. These areas include:

- Development engineering - This area includes development, unit testing, build and integration. It functions as the crux of the organization for product development.

- Project management office - This team functions as the nexus between business and engineering. It drives the effective flow of work through the organization.
- Product management - Product management makes strategic decisions about what products should be built based on its knowledge of the market requirements and works with marketing to let them know what to communicate.
- Quality engineering - SQE looks for mistakes or defects in the products being developed to avoid bugs and issues after deployment.
- UX and Design - The user experience and design team ensures that the product has an intuitive, simple and appealing design that works well on several devices.
- Other - We categorized research, technical operations management, assessment and learning design in this category.

Overall 86 respondents work in development engineering, 46 work in the project management office, 23 respondents work in product management and 34 in quality. There are 7 respondents working in user experience and design and 8 working in various other work areas.

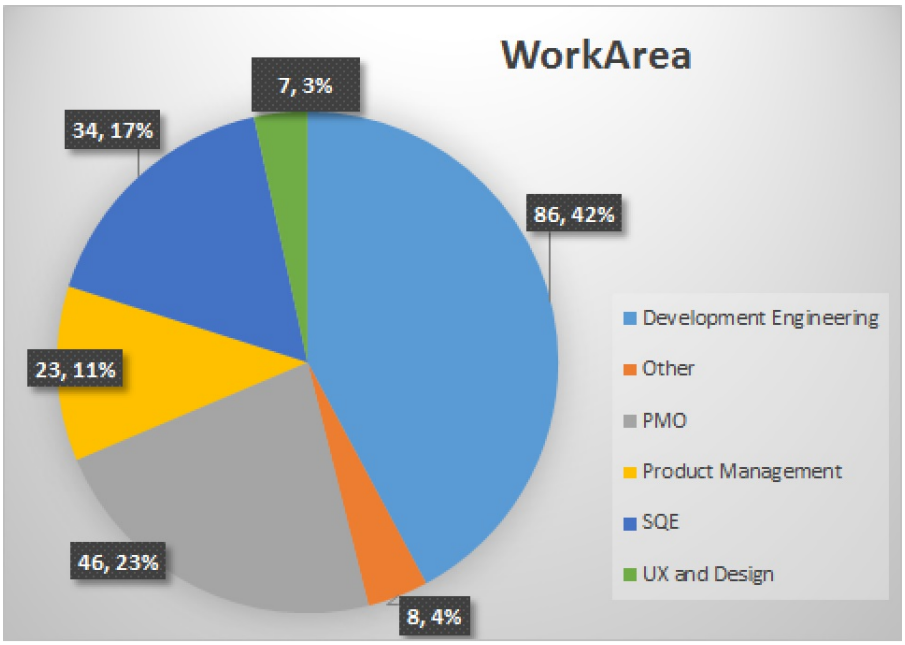

Figure 5.1: Work area

5.1.2 Location

The respondents work in different locations across the world. A vast majority (141) of the respondents work in the North America - United States while there are 60 respondents from Asia, 3 from Europe and 1 from Australia.

- Across United States there are respondents from Centennial CO, Boston MA, Hoboken NJ, Glenview IL, Chandler AZ, San Francisco CA, Field US, Tempe AZ, Piscataway NJ, New York NY and Phoenix AZ
- Across Asia respondents are from Colombo SriLanka, Bengaluru India, Hyderabad India and Chennai India
- From Europe respondents are from London UK

- From Australia respondents are from Hobart
- There are also 8 respondents who work remotely

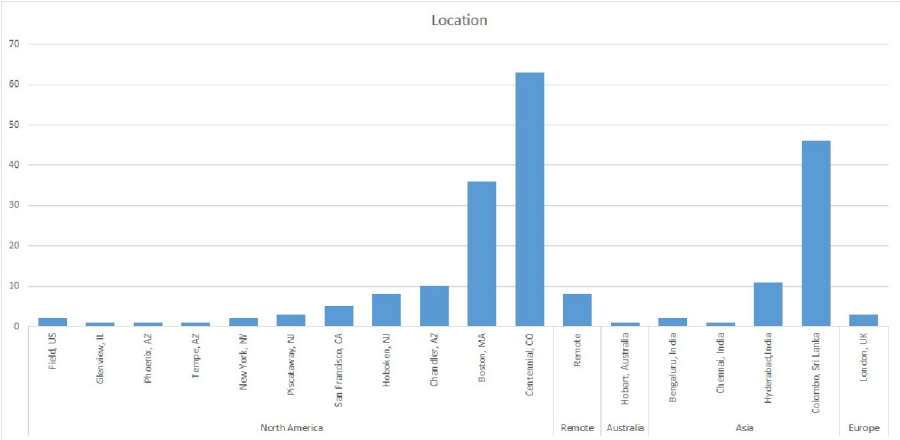

Figure 5.2: Respondent locations

5.1.3 Role

There are 13 different roles among the respondent population:
Developer, tester, data architect, software architect, scrum master, product owner, agile coach, manager, functional manager, manager of manager, business analyst, technical writer and others.

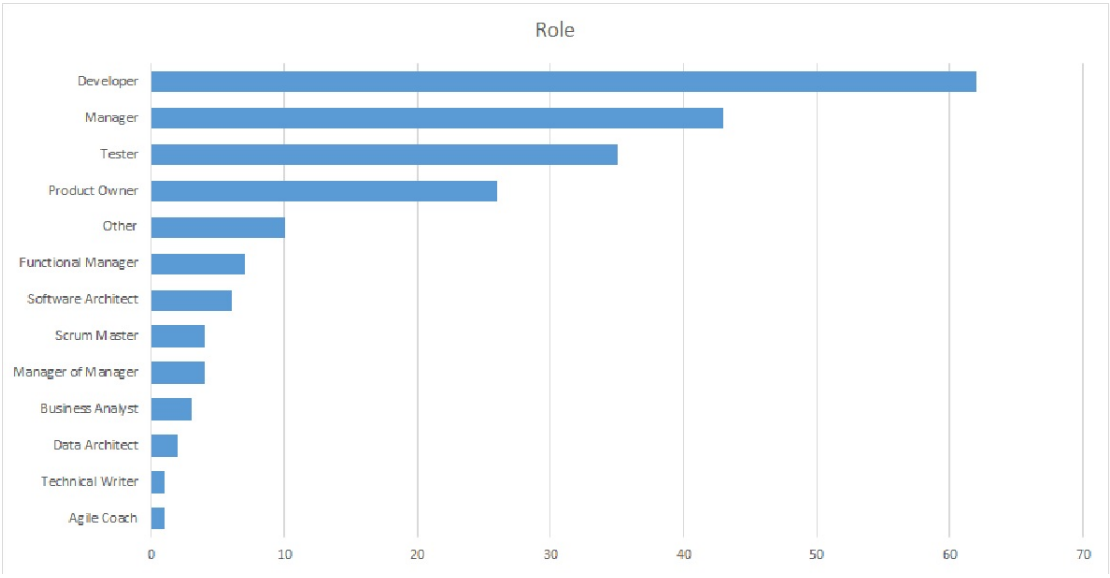

Figure 5.3: Count of respondents in different roles

5.1.4 Experience

The average experience of the respondents in the software industry is 12.8 years with a standard deviation of 7.6, with a minimum of 0.5 years and a maximum of 35 years. The distribution of experience can be seen in Figure 5.4 and 5.5.

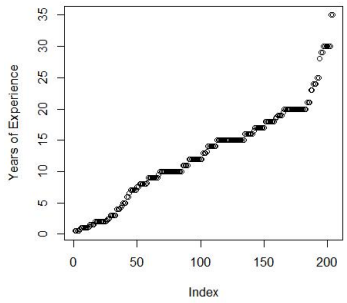

Figure 5.4: Experience of respondents

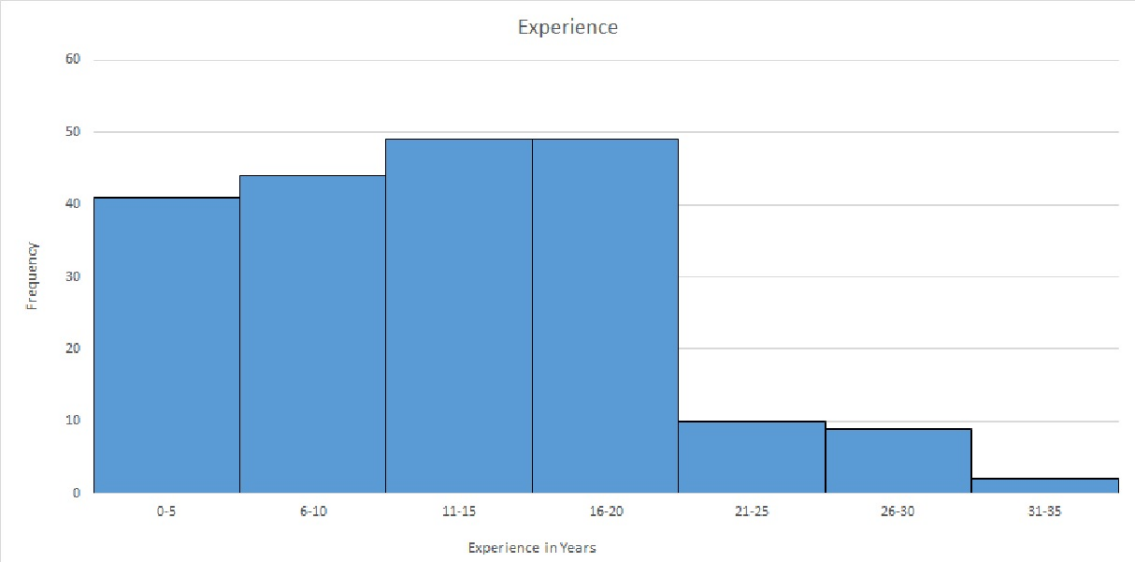

Figure 5.5: Experience of respondents

The plot in Figure 5.6 shows the correlation between experience and work area. The box plot shows the median experience for respondents working in different work areas, the higher and lower quartiles and the outliers.

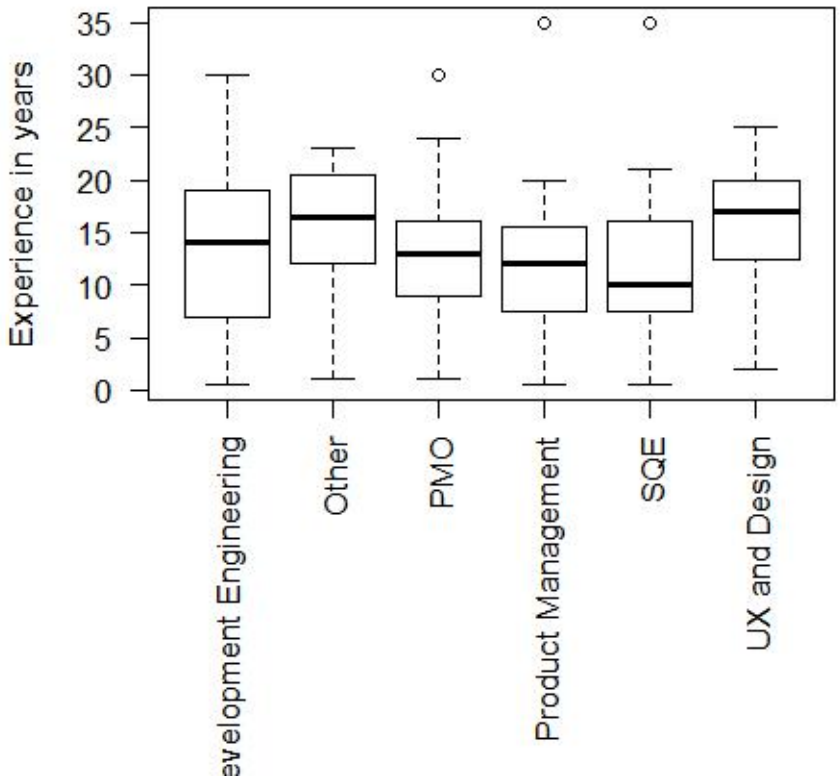

Figure 5.6: Correlation of work area and experience

The plot in Figure 5.7 shows the correlation between experience and roles. The box plot also shows the median experience for respondents in a specific role, the higher and lower quartiles and the outliers.

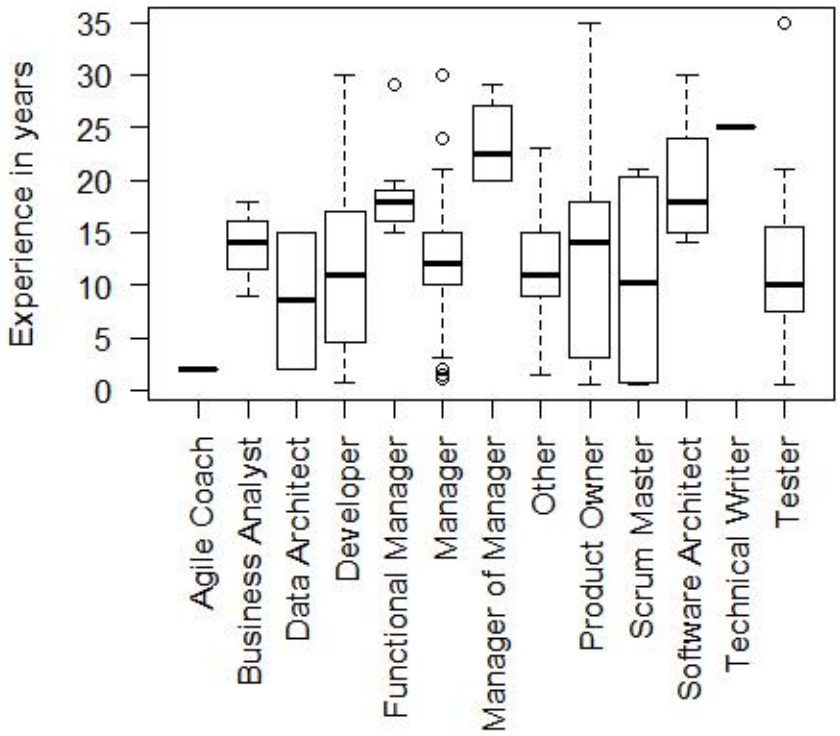

Figure 5.7: Correlation of roles and experience

Discussion: A vast majority of the respondents are developers working in the development and engineering area. The survey respondents are mostly from Centennial Colorado. Boston, Massachusetts and Colombo Sri Lanka are the other highly represented locations. The average experience of the respondents is 12.8 years. Thus we can see that there is wide diversity in the survey respondents based on their work areas, roles, location and experience. Based on the experience it can be said that the respondent population is fairly mature.

5.2 Extent of agile adoption

We analyze the extent of adoption of agile development by asking respondents if they use agile development methodologies or non-agile development methodologies on their projects. The graph in figure 5.8 shows the extent of adoption of agile methods.

Discussion: 89% of the respondent population uses agile development while 11 % of the respondents use non-agile development methodologies on their current projects at Pearson Education.

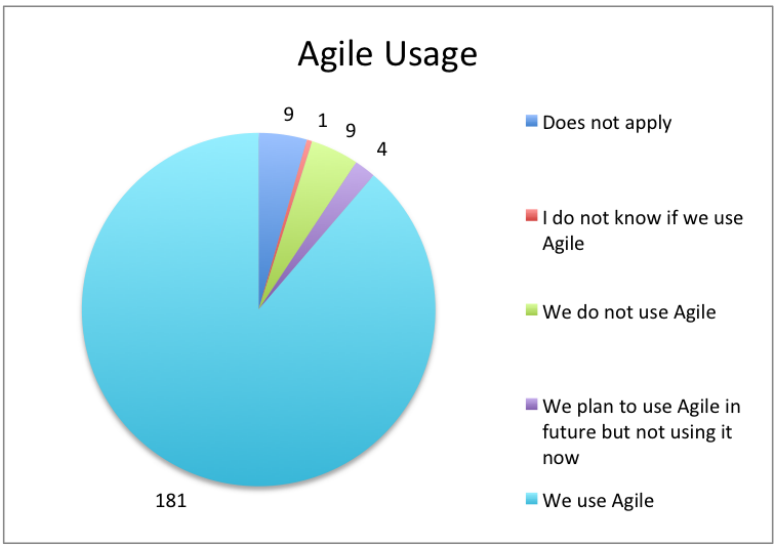

Figure 5.8: Usage of agile

5.3 Non-agile development

5.3.1 Benefits of non-agile development

We ask respondents about the benefits of using non-agile development methodologies. The survey allows the respondents to choose from the following benefits:

- Better adherence to requirements
- Better communication with management
- Cost effectiveness
- Documentation useful for on-boarding new team members
- Easily scalable to large teams
- Each stage has an expected results so easy to coordinate due to model rigidity
- Higher quality testing
- Increased productivity
- Increased quality of deliverables
- Stable architecture
- Scheduled process - one stage at one time during development
- Simple, easy to use software development model
- Structured design
- Reduction in defects

From the graphs in Figure 5.9 it can be seen that higher quality testing, scheduled process - one stage at one time during development and structured design emerged as the top three benefits of following the non-agile methodology.

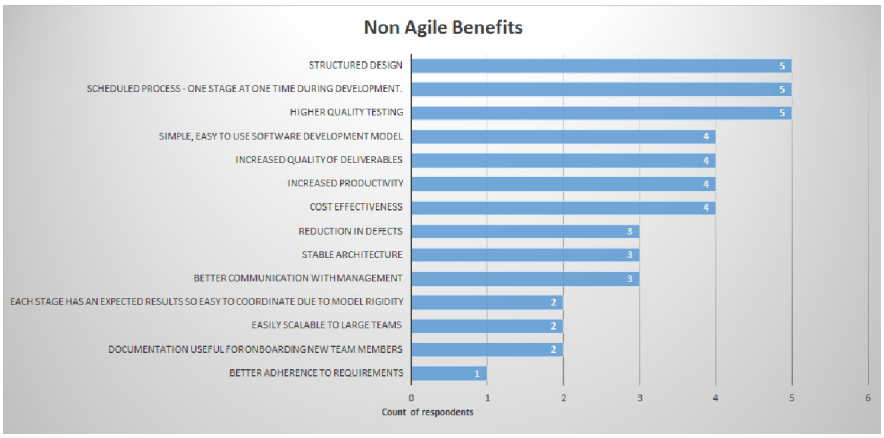

Figure 5.9: Benefits of non-agile methods

5.3.2 Challenges of non-agile development

We ask respondents about the challenges of following non-agile methodology. The survey allows the respondents to choose from the following challenges:

- Changing requirements cannot be accommodated in the same version of the software
- Estimation of time and budget for each stage is very difficult
- Design issues found during testing expensive and difficult to correct
- High risk in the entire life cycle of the development
- Low utilization of resources
- No option of changing (partitioning) the project into multiple stages
- No prototype before the end of the life cycle
- Requirements emerge after the requirements gathering phase

- Problems detected at a stage are not solved completely in the same stage
- Testing occurs in the last stage of the development
- Rigid process
- Time to market too long

From the graph in Figure 5.10 it can be seen that estimation of time and budget for each stage, requirements emerging after the requirements gathering phase are the top challenges and accommodating changing requirements in the same version of the software.

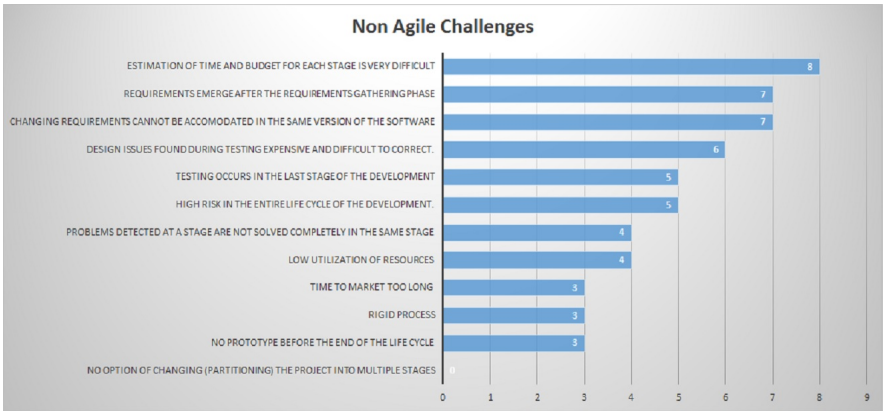

Figure 5.10: Challenges of non-agile methods

Discussion: The main benefit of non-agile development methods is the scheduled process, structured design and the quality of testing. Accommodating changing requirements and estimation are the top challenges. It is also observed that the total count of benefits (50) reported is lower than the total count of challenges (58) reported. This suggests that the users of non-agile development face more challenges than benefits in this methodology.

5.4 Agile Development

5.4.1 Agile Development Methodologies

The agile methods Scrum and Kanban are practiced among the survey respondents. From 5.11 it can be seen that 79 % of agile practitioners use the Scrum method while about 14 % use the Kanban method. Some respondents use a combination of Scrum and Kanban. At the program level some respondents have some teams use Scrum while others use Kanban. One respondent mentioned using scrum at the team level and waterfall at the program level.

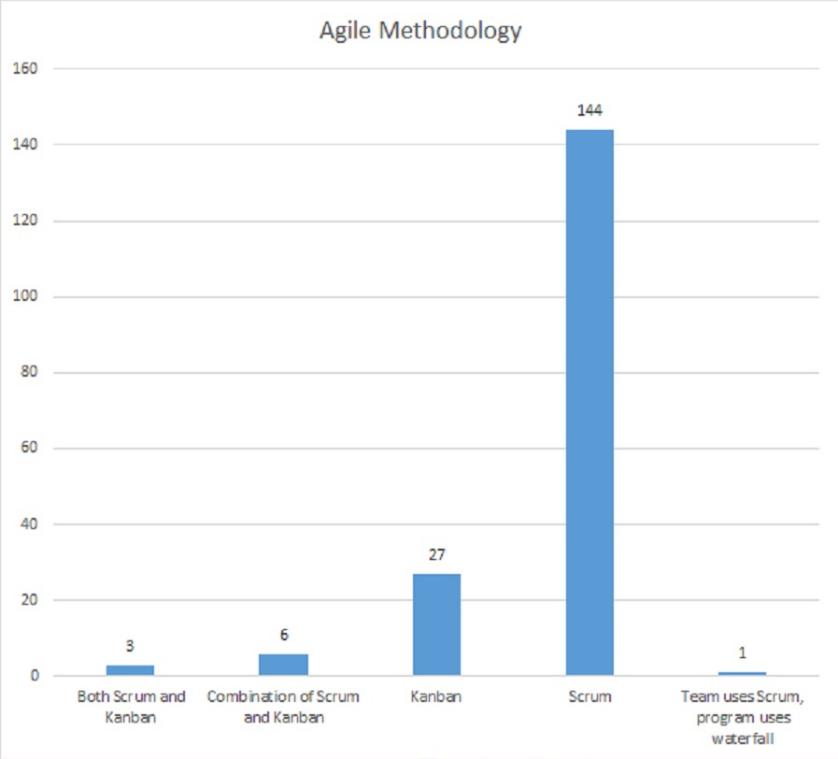

Figure 5.11: Use of agile methods

Discussion:A vast majority of the organization uses the Scrum method while the rest use Kanban or a combination of Scrum and Kanban. At the program level there are a few programs in which some teams use the Scrum method while others use the Kanban method.

5.4.2 Past non-agile projects experience

To understand if the respondents have any background to compare agile development against other methods we ask respondents working on agile programs and projects if they have worked with non-agile development methodologies in the past. We find that over 160 of the 184 respondents who work on agile methods have past non-agile development experience. This indicates good exposure to other development methodologies.

Past non-agile development experience	160
Only agile development experience	24
Total	184

Table 5.1: Agile methodology users with non-agile development experience

5.4.3 Maturity of agile projects

For analyzing the maturity of agile development projects, we record the time for which the projects our respondents work on have been using agile techniques. On an average projects have been using agile techniques for 2.49 years with a standard deviation of 1.84 years. The plot in Figure 5.12 shows the distribution for the maturity of projects using agile development techniques.

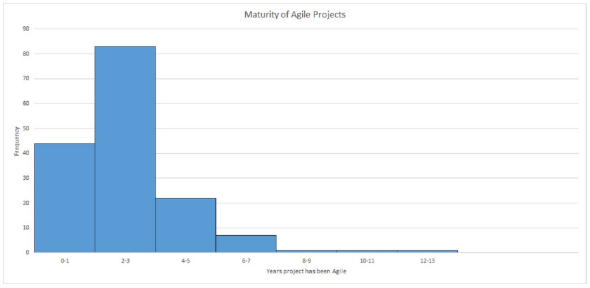

Figure 5.12: Maturity of agile projects

5.4.4 Willingness to switch methodologies

We ask respondents working on agile development methodology if they would like to switch to a non-agile methodology and those working on non-agile projects if they would like to switch to agile development methodology. The graph in Figure 5.13 shows the plot of the responses. We found that 11% of agile development users would like to switch to non-agile methods while 78% of non-agile development respondents would like to switch to agile methods.

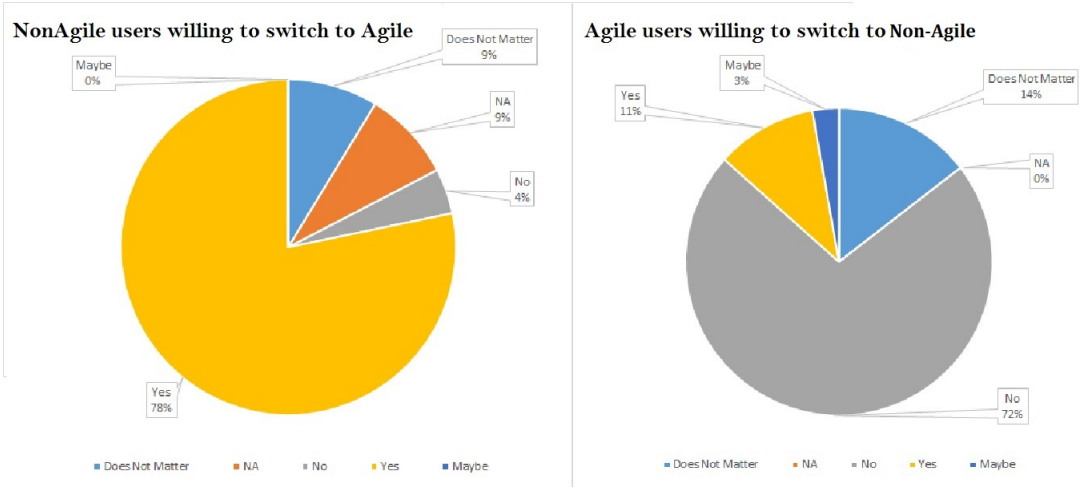

Figure 5.13: Willingness to switch methodologies

Discussion: It can be seen that most of the respondents have past non-agile work experience. The average maturity of agile projects is 2.49 years. Most of the agile development users would like to continue with agile development techniques; however, most of the non-agile methodology users would like to switch to agile development.

5.4.5 Training

To understand the training needs of the respondent population we evaluate the trainings done by our respondents in agile techniques. It is seen that 160 of 181 respondents working on agile projects have done some training in agile techniques. About 9% of the respondents have no training in agile techniques.

The respondents mentioned various different trainings out of which most relevant are:

Continuous Delivery, Kanban DevOps, Agile Testing, Agile Product Management, Scrum Practices, SAFe Training , Scaled Agile Framework Program Consultant Certification(SPC), Scaled Agilist Certification(SA), Certified Scrum Master(CSM), Certified Scrum Product Owner(CSPO) and Agile Certified Practitioner(ACP)

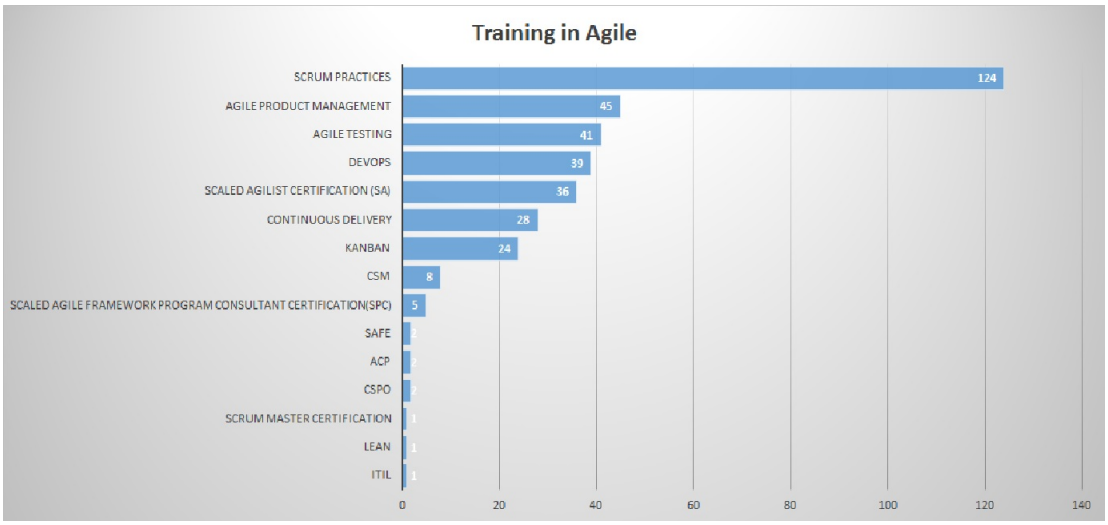

Figure 5.14: Count of respondents for agile trainings

Discussion: Most of the respondents have some training in agile methods however there are a few respondents with no training in agile methods. Scrum practices training is done by a majority of the respondents.

5.5 Agile implementation at the team level

5.5.1 Project Independence

Dependency management is usually a major challenge for large scale companies. Thus, we evaluate if the projects at Pearson Education face the same challenge. The respondents are asked if the project team is able to independently complete the user stories with its own resources and if not, how many other projects it is interdependent on. The responses suggest that 115 of the respondent's projects have dependencies. These projects are not able to complete their product backlog independently. There are 27 respondents who mention that their projects have no dependencies on other

projects. For those who say there are dependencies we also ask them about how many projects their project depends on; however, the respondents mention that the number changes with every iteration. The mean dependency count is 2.5. The pie chart in Figure 5.15 depicts the responses about project dependencies.

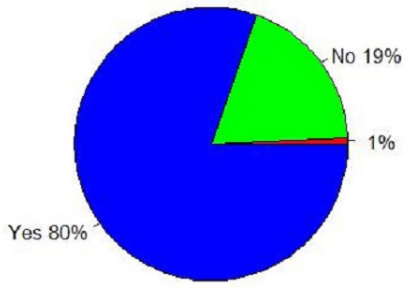

Figure 5.15: Project dependence

5.5.2 Team Members

We ask respondents working in agile development teams about their team size. Implementing agile techniques requires close coordination and increased communication among team members. Thus, team size is an important factor in determining the success of implementing agile techniques successfully. The recommended team size is between five and nine [3]. In our data we observe that the mean team size at Pearson Education is 11.6 with a standard deviation of 7.74.

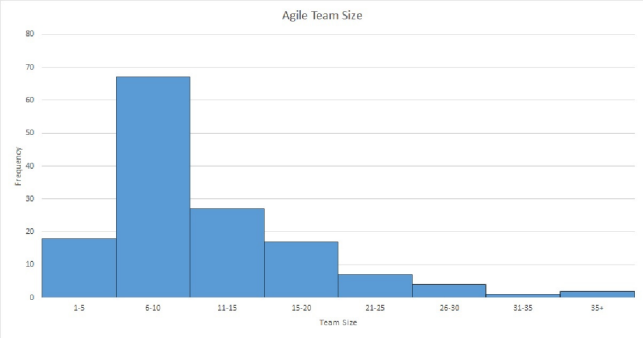

Figure 5.16: Team Size

5.5.3 Practices

At the project and program level the following practices are followed:

Team Ceremonies:

- Burndown charts
- Daily stand up
- Customer interaction
- Definition of done
- Fixed-length sprints
- Product backlog
- Quarterly planning
- Retrospective
- Release planning
- Sprint planning

- User stories
- Velocity

The graph in Figure 5.17 shows that except for maintaining burndown charts, all the other practices are followed rigorously by over 70 % of the respondents. Burndown charts are used by about 50 % of the respondents.
The details of the responses can be seen in the graph below.

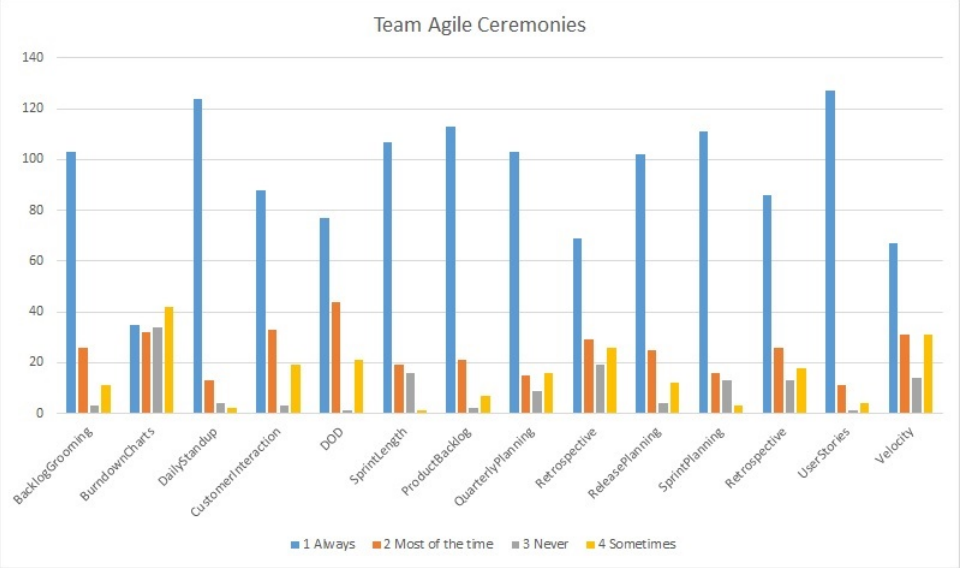

Figure 5.17: Team agile ceremonies

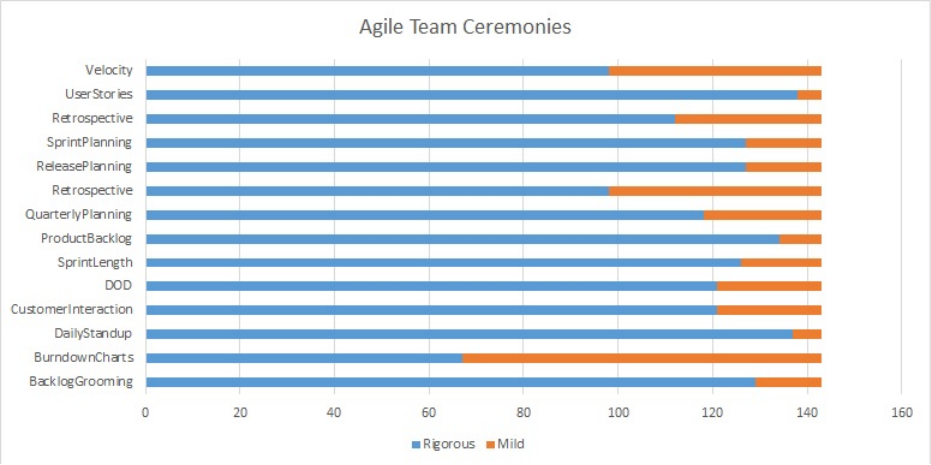

Figure 5.18: Team agile ceremonies

Engineering practices

Below are the engineering practices followed at the team level:

- Continuous integration of code (EPcontintegration)
- Collective code ownership (EPOwnership)
- Pair programming (EPPP)
- Small regular releases (EPSmallRelease)
- Team coding standards - code reviews (EPCodingStd)
- Test-driven development - writing unit tests before coding(EPTDD)

The graphs in Figure 5.19 and 5.20 show that all practices except pair programming, test driven development and small releases are practiced rigorously by over 70 % of the respondent population. The details of the responses can be seen in Figure 5.19 and 5.20 graphs.

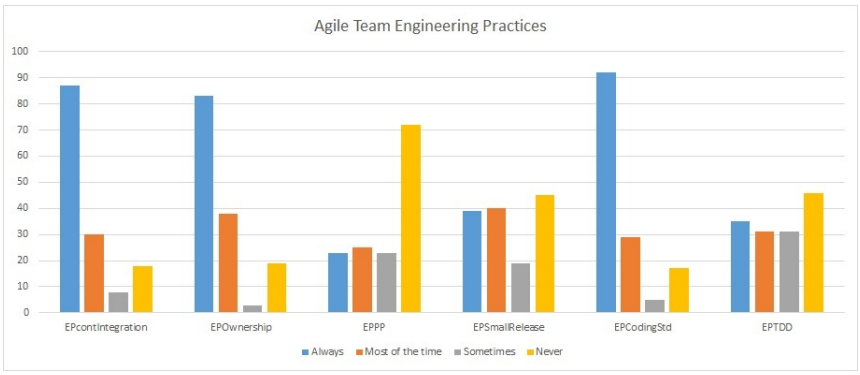

Figure 5.19: Team engineering practices

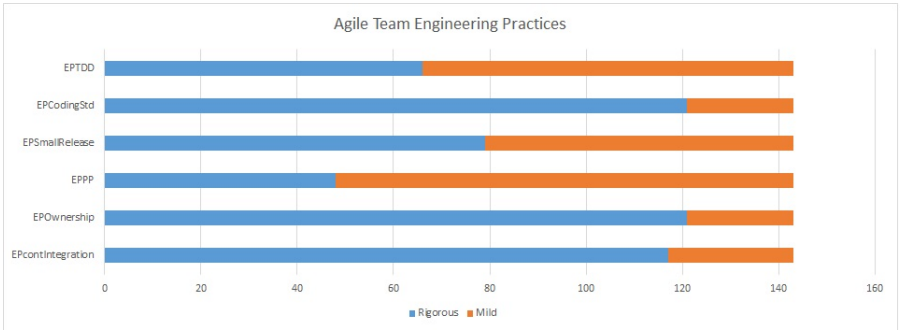

Figure 5.20: Team engineering practices

Discussion: Most of the projects are dependent on other projects. The average team size is 11.6. Among the several team ceremonies at the project level, burndown charts are least rigorously used and among the engineering practices pair programming and test driven development are practiced with the least amount of rigor. As the Extreme Programming method of agile development is not used at Pearson Education, the low rigor on pair programming is acceptable; however, it can be seen that there is less focus on testing than would be ideally expected. However, the overall rigor of the agile team practices is high.

5.6 Agile implementation at the program level

Below are the agile practices followed at the program level in order to scale agile techniques:

- Program retrospectives
- Programs pulling the backlog from portfolio investment items
- Scrum of scrums
- Integration testing before release
- Quarterly release retrospective evaluates how well the investment item value proposition was met
- Program dependency tracking
- Roadmap planning

From the graphs in Figure 5.21 and 5.22 it can be seen that program dependency tracking, roadmap planning and integration testing before release are rigorously followed by over 20 out of 23 respondents working at the program level.

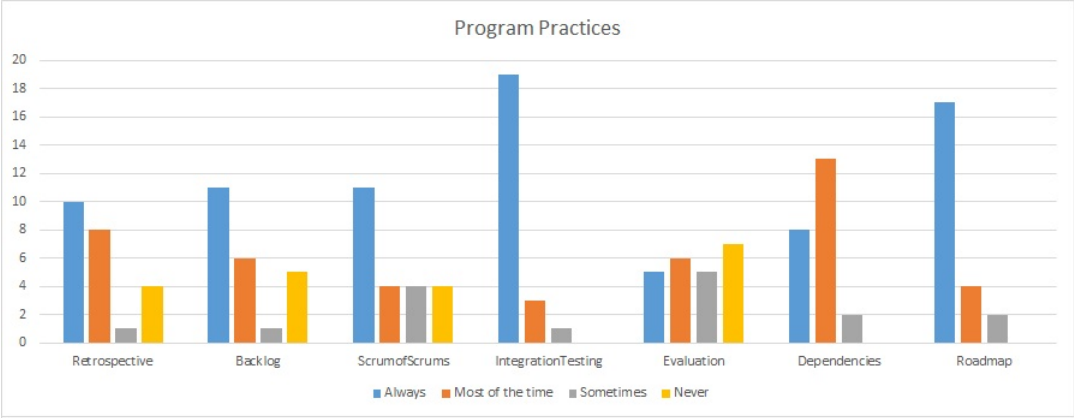

Figure 5.21: Program practices

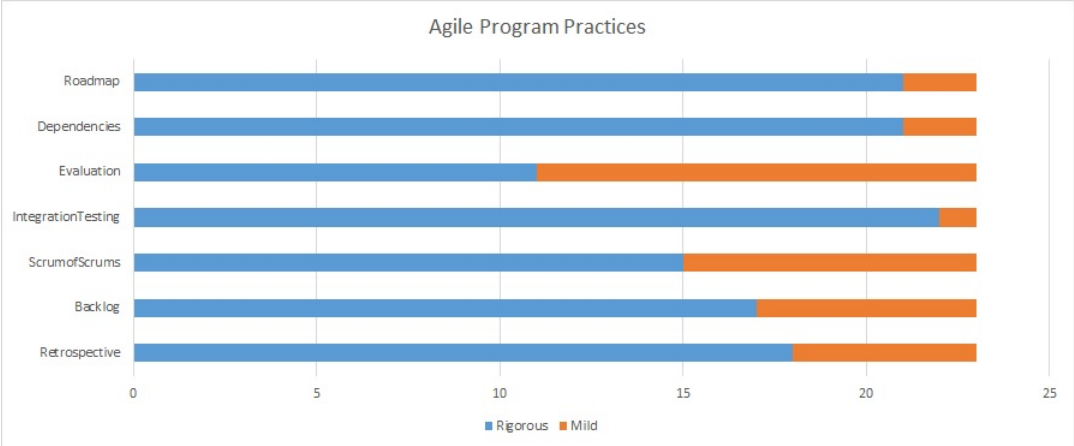

Figure 5.22: Program practices

5.6.1 Program Demos

It is observed that 20 of 23 programs perform regular demonstrations for the stakeholders.

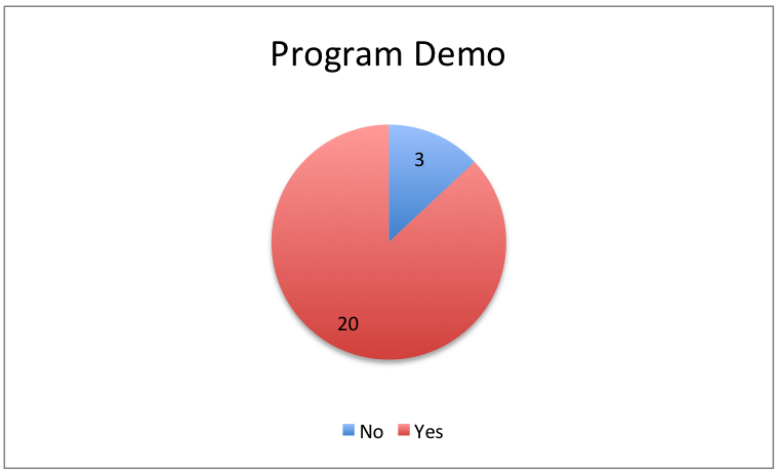

Figure 5.23: Program demos

5.6.2 Program Release

The programs release on the BackToSchool(annual), continuous or quarterly schedule. We asked the respondents about which release cadence they follow. The pie chart in Figure 5.24 depicts their responses.

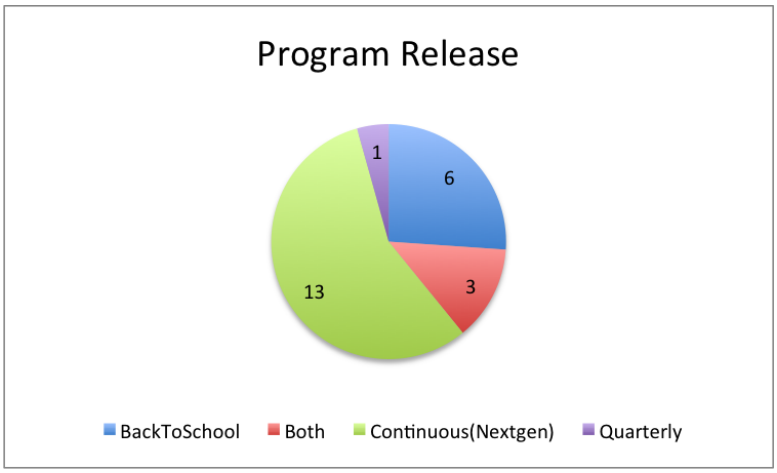

Figure 5.24: Program Release

5.6.3 Program core team meeting

The program core team meetings are held to synchronize the efforts of the entire program team. We ask respondents working at the program level how often their teams meet and their responses are displayed in the pie chart in Figure 5.25. It suggests that 70 % of the respondents have weekly program core team meetings.

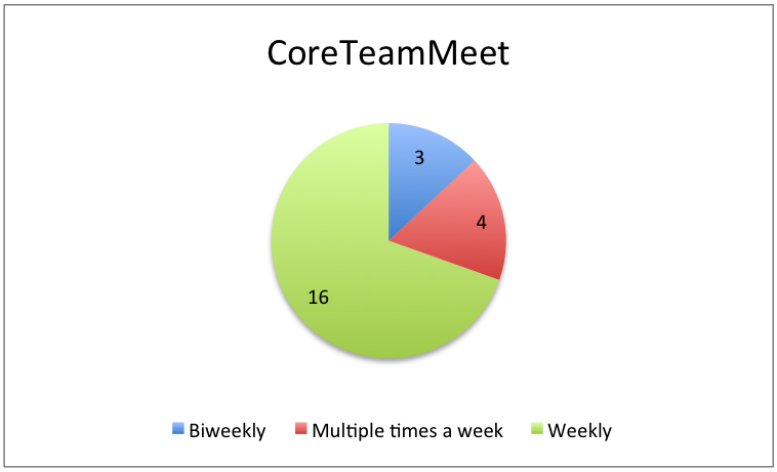

Figure 5.25: Program Core Team Meeting

5.6.4 Content authority

To understand how the investment theme is decomposed at multiple stages to ultimately form user stories, we ask respondents at the program level in agile development to provide a hierarchy and we got the decomposition structure of **Investment Theme - Feature - Epic - User Story**. Users stories are maintained at the team level; features and epics are maintained at the program level. The content authority refers to the way these features and epics are prioritised in the program backlog. It is handled differently in different programs. Content authority is shared between the

program manager, engineering team, program core team and product owner. The pie chart in Figure 5.26 shows the content authority at the program level.

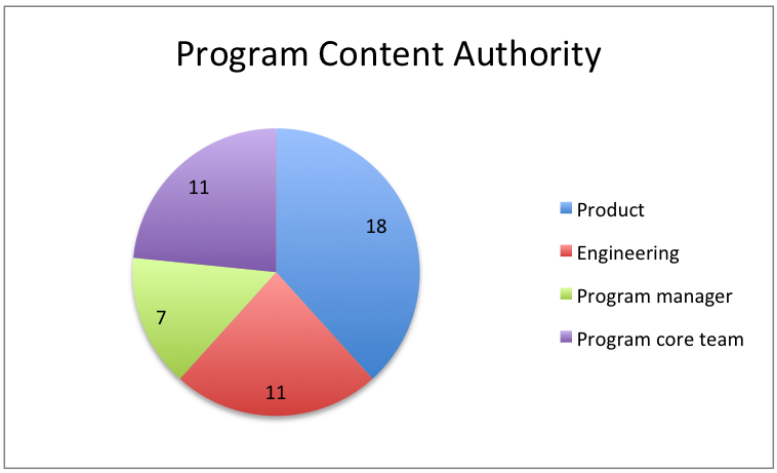

Figure 5.26: Defining and prioritizing the backlog

Discussion: All the program practices except evaluation of programs are practised rigorously. 85% of programs have regular demos. Among all the release methods, continuous release is practiced by most of the programs and the program core teams have meetings on a weekly basis. The program content authority is evenly distributed between the product owner, engineering, management and core team.

5.7 Perception of agile development

To understand the perception of our respondents towards agile techniques we ask them how much they agree with the statements below.

- Architecture: The product architecture is mature and stable. New functionality can be added easily without significant redesign. Hence when changes are

introduced, the team is confident the next release of the product will meet the demands without significant architectural rework.

- Teams Self Organizing: The teams are self organizing. Teams make decisions about work agreement and frequently discuss, criticize and experiment with work flow. Team is organized without undue influence from others.

- Frequent Integration: As the team is aware that it is risky to take too much change all at once, risk is managed by frequently integrating and releasing small set of features.

- Trust Respect: The work environment makes everybody feel trusted and respected. Team members have disagreements and constructively engage with one another to resolve differences.

- Collaborative: Teams are collaborative and there is collective ownership of the product throughout the life cycle. When problems surface, they are solved as a team.

- High Energy Work Environment: Agile development provides for a high energy work environment.

From the graph below in Figure 5.27 it can be seen that the respondents agree with the statements on collaboration, self organizing teams, trust respect and high energy work environment more than the statements on frequent integration and architecture. However overall there is positive perception of agile development based on these parameters. The observations show that the perception on how agile techniques impact the cultural aspects of the team is very positive; however, impact on the technical aspects of a stable architecture and release is less positive.

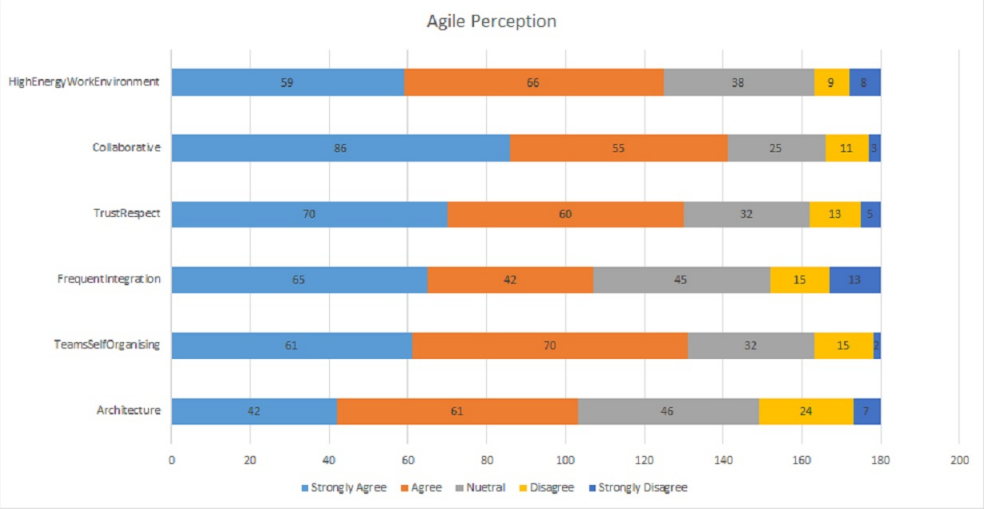

Figure 5.27: Agile development perception

We ask respondents if agile techniques are working well for them at different levels (personal, team, group-program and while interacting with the management). Our data as seen in Figure 5.28 suggests that agile development works well at the personal and team level; however it does not work as well in larger groups and when interacting with management.

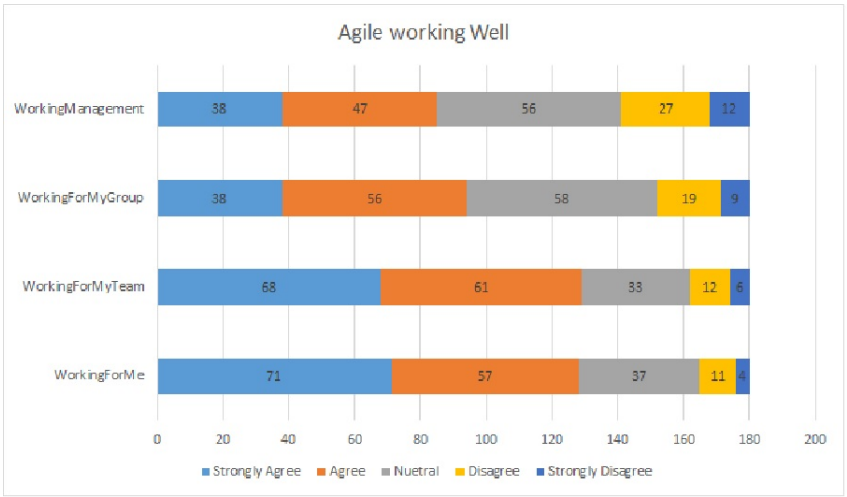

Figure 5.28: Agile development perception

Discussion: The organization overall has a positive perception of agile development. The perception about how agile techniques work for the team culture is more positive than how it works for the technical aspects of architectural stability and frequent integration. Agile techniques seem to work very well at the individual and team level but not as well at the upper levels.

5.8 Benefits of agile development

We list the commonly identified benefits of agile development. Below are the listed benefits:

- Better customer focus
- Better morale
- Cost effectiveness
- Correctness of code
- Flexibility of design
- Improved communication and coordination
- Increased quality
- Improved focus - better prioritization
- Increased productivity
- More reasonable process
- Reduction in defects

- Satisfaction of team
- Quick releases
- Quicker response to changes
- Testing first

The graph in Figure 5.29 displays the benefits as experienced by the respondents of our survey working in agile development teams and programs. Based on the responses received, improved communication and coordination and improved focus - better prioritization are the top 2 benefits of following agile methods.

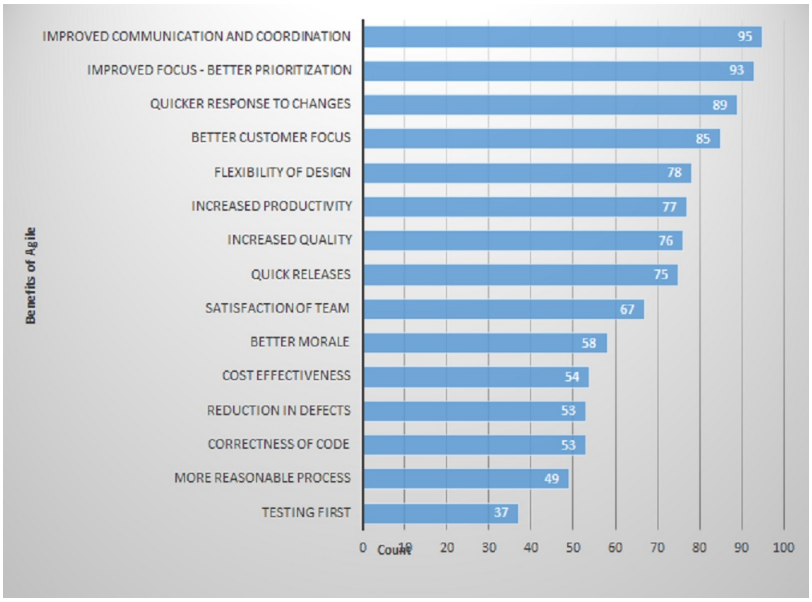

Figure 5.29: Benefits of agile methods

5.9 Challenges of agile development

Below is the list of common non-agile development challenges provided to our respondents:

- Coordination with other teams
- Excessive meetings
- Reduced focus on the architecture and design
- Demanding culture
- Low management buy-in
- Unfamiliar with practices
- Difficult to increase team size
- Short sighted development
- Requirements revision management
- Dev/Test integration
- Hard to manage time
- Lack of schedule

The graph in Figure 5.30 displays the challenges as perceived by the respondents working in agile teams and programs. Coordination with other teams and excessive meetings emerged to be the top challenges of following agile methodology.

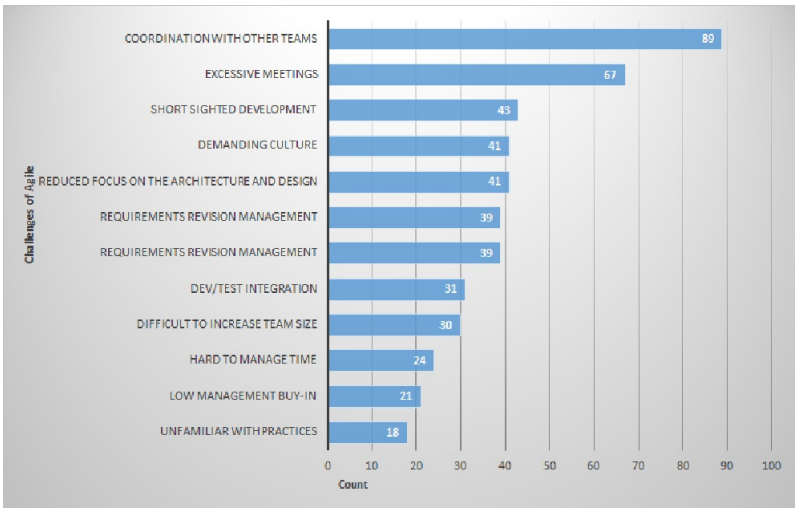

Figure 5.30: Challenges of agile methods

Discussion: The main benefits of agile development are improved communication, quicker response to change and better prioritisation while co-ordination with other teams is the biggest challenge. Quality testing is conventionally considered to be a major benefit of agile development, however at Pearson Education it is listed as a benefit by the least number of respondents signifying that there could be improvement by increasing the rigor in testing practices. It is also observed that the total count of benefits (1039) is significantly higher than the total count of challenges (442) reported thus, confirming that the users of agile development see more benefits than the challenges with this methodology.

5.10 Comparison of the two waves of responses

The survey was open for two weeks. An email was sent to announce the survey. After a week there was a reminder sent. Thus, the responses could be grouped in two waves. The respondents in the first wave are immediate respondents while the second

wave has the responses received after the reminder email was sent. The first wave has 100 respondents while the second wave has 104 responses. It is possible that the respondents who responded immediately (in the first wave) feel strongly about agile development. It is important to understand if the results from the survey sample are generalizable to the entire population or have an undue bias. Hence we compare the responses in these two waves.

From our exploratory and inferential analysis, we observe that there are no significant differences between the two waves based on the parameters of team/program practices, benefits, challenges and the perception of agile development. Thus, we can conclude that these results are generalizable to the population of the organization.

Inferential Analysis For the inferential analysis we use Fisher's Exact Test. This test is useful for categorical data that result from classifying objects in two different ways; it is used to examine if there are significant differences in the data. The p-value from the test is computed as if the margins of the table are fixed. This leads under a null hypothesis of independence to a hyper geometric distribution of the numbers in the cells of the table. If the p-value is above 0.05 it means that we can accept the null hypothesis that there are no significant differences in the two populations being compared. If p-value is below 0.05 we reject the null hypothesis of independence and conclude that there are significant differences in the populations being compared.

Our criteria of classification is immediate respondents - wave1 and hesitant respondents - wave2. The Fisher's Exact Test was applied to the programming practices, engineering practices, team ceremonies and perception of agile development. Following are the **pvalues** for every practice:

Program practices

- Retrospective - 0.6373
- Backlog - 0.795
- Scrum of Scrums - 1
- Integration Testing -1
- Evaluation - 0.4202
- Dependency Tracking - 0.04792
- Roadmap - 0.1873

The p-value for the program dependency tracking is the only one below 0.05. Thus, wave 1 is not significantly different from wave 2.

Team engineering practices

- EPcontIntegration - 0.2671
- EPOwnership - 0.1617
- EPPP - 0.6246
- EPSmallRelease - 0.5486
- EPCodingStd - 0.7723
- EPTDD - 0.9734

The p-values signify that wave 1 is not significantly different from wave 2.

Team ceremonies

- Backlog Grooming - 0.3965
- Burndown Charts - 0.7543
- Daily Standup - 0.3942
- Customer Interaction - 0.2434
- Definition Of Done - 0.001074
- Sprint Length - 0.798
- Product Backlog - 0.05449
- Quarterly Planning - 0.8228
- Retrospective - 0.8916
- Release Planning - 0.7039
- Sprint Planning - 0.1872
- User Stories - 0.2458
- Velocity- 0.1796

The p-value for the Definition of Done practice is the only one below 0.05. Thus, wave 1 is not significantly different from the wave 2.

Agile Working Well

- Working For Me - 0.9981
- Working For My Team - 0.725
- Working For My Group - 0.5673

- Working With Management - 0.9199

The p-value signifies that the wave 1 and wave 2 data about how well agile works for people is not significantly different. We accept the null hypothesis that both the samples are drawn from the same continuous distribution.

Perception of agile

- Architecture - 0.5463
- Teams Self Organizing - 0.9141
- Frequent Integration - 0.772
- Trust Respect - 0.1269
- Collaborative - 0.6828
- High Energy Work Environment - 0.5652

The p-value signify that the wave 1 data is not significantly differ from the wave 2 data except for the definition of done practice. Thus, generally speaking the we can accept the null hypothesis of Fisher's Exact test.

Experience

We perform the two-sample Kolmogorov-Smirnov test on the experience data from the two waves to see if there are significant differences and the p-value is **0.005437**. Thus, there is a significant difference in the experience of respondents in the two waves

Benefits and Challenges

We perform the two-sample Fisher's Exact Test on the benefits and challenges data from the two waves to see if there are significant differences. P-values are as

below:

Description	P-Value
Benefits	1
Challenges	0.119

Table 5.2: Fisher's Exact test results for benefits and challenges of the two waves

Based on the p-values we can accept the null hypothesis that both the samples are drawn from the same continuous distribution. There are no significant differences.

Graphical Analysis: The experience of the respondents in the two waves differs significantly based on distribution in the histograms in Figure 5.31. The first wave has an average experience of 11.37 years while in the second wave the average experience is 14.20 years.

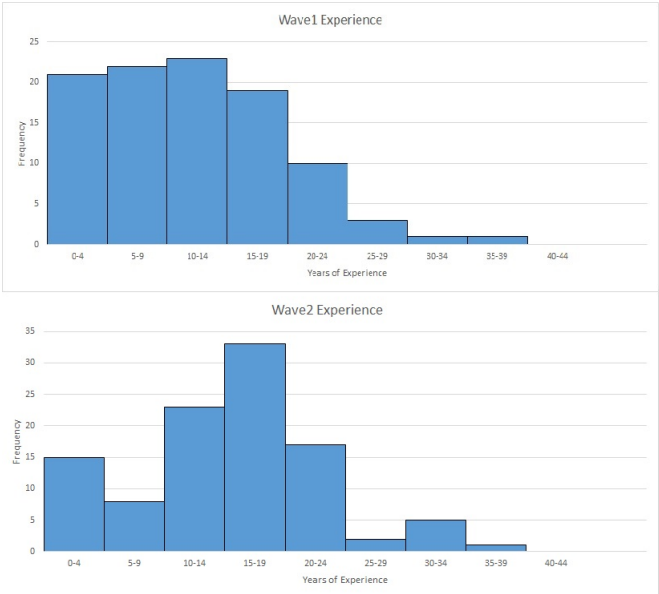

Figure 5.31: Comparison of the experience of respondents in the two waves

The graph in Figure 5.32 compares the team ceremonies in the two waves and as we can see that in the visual presentation the differences do not look stark.

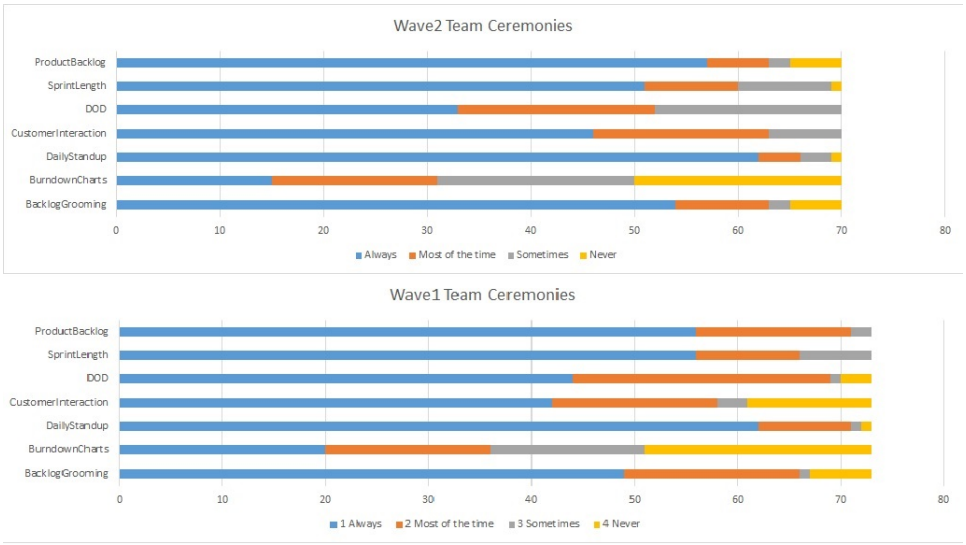

Figure 5.32: Comparison of team ceremonies in the two waves

The graph in figure 5.33 compares the team practices in the two waves and as we can see that in the bar plot presented side by side that there are no significant differences.

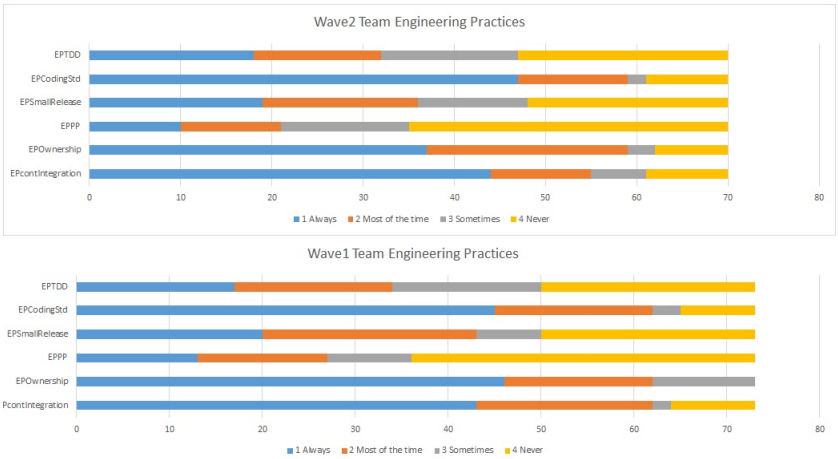

Figure 5.33: Comparison of team practices in the two waves

The graph in Figure 5.34 compares the program practices in the two waves and as we can see in the bar plot presented side by side there is a significant difference in the rigor of dependency tracking however the other practices have a similar rigor.

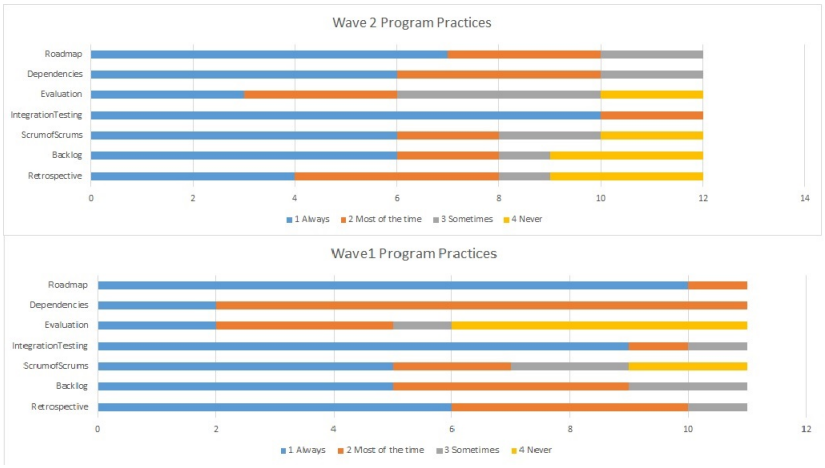

Figure 5.34: Comparison of program practices in the two waves

There is almost an equal count of responses in wave 1 and wave 2. Side by side comparison of the benefits in wave 1 and wave 2 is shown in the graph in Figure 5.35. The respondents in the wave 2 have generally reported more challenges that those in wave 1. The respondents in the second wave are more experienced than the first wave. This may indicate that higher experience leads to decreased tolerance towards the challenges of agile development as they tend to spend more time in meetings and co-ordinating due to the roles they work in than on the actual task reducing their productivity. Consistent with our previous observation the total count of benefits is higher on both the waves than the total count of challenges. Thus, confirming the positive opinion of agile development across the two waves of responses.

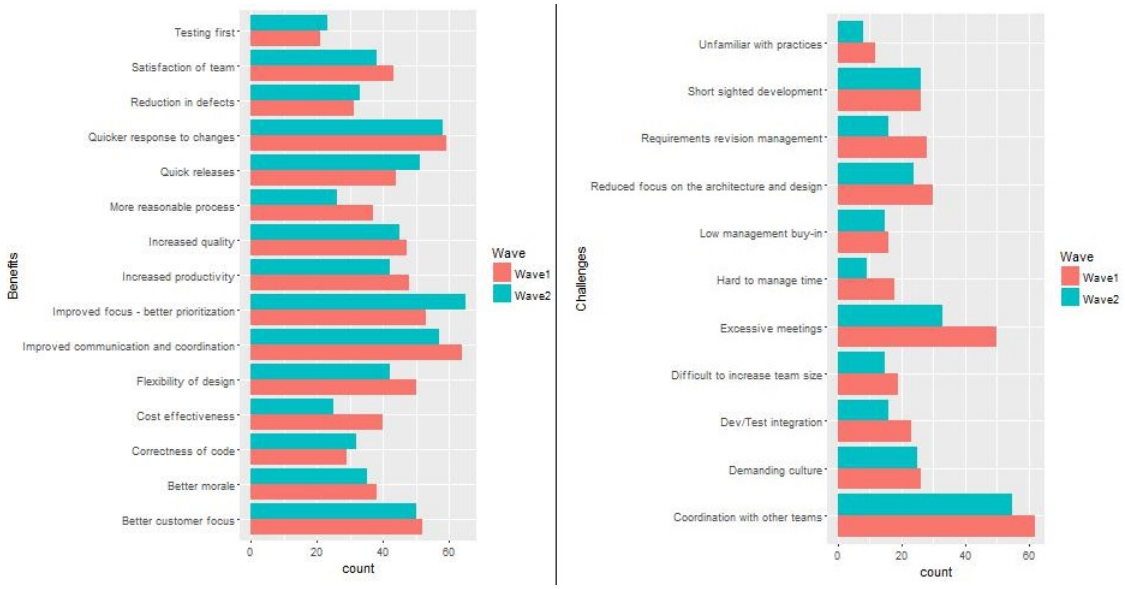

Figure 5.35: Comparison of the challenges in the two waves

The graph in Figure 5.36 compares the perception of agile in the two waves and differences do not look stark in the visual presentation.

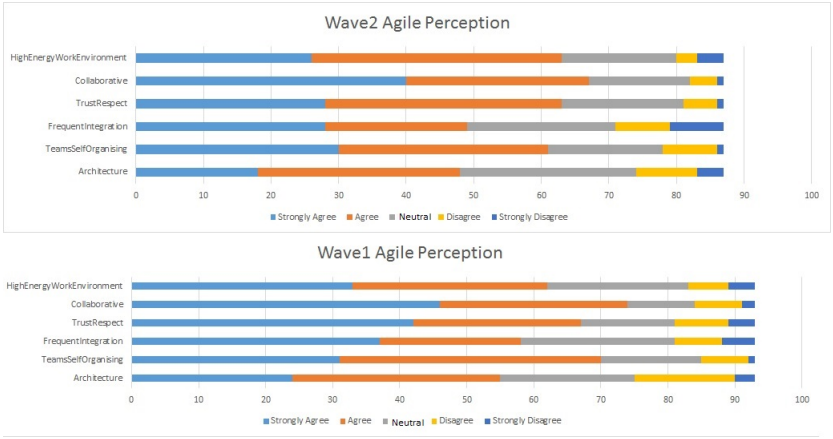

Figure 5.36: Comparison of the perception of agile in the two waves

The graph in Figure 5.37 compares how well agile is working in the two waves there are no significant differences.

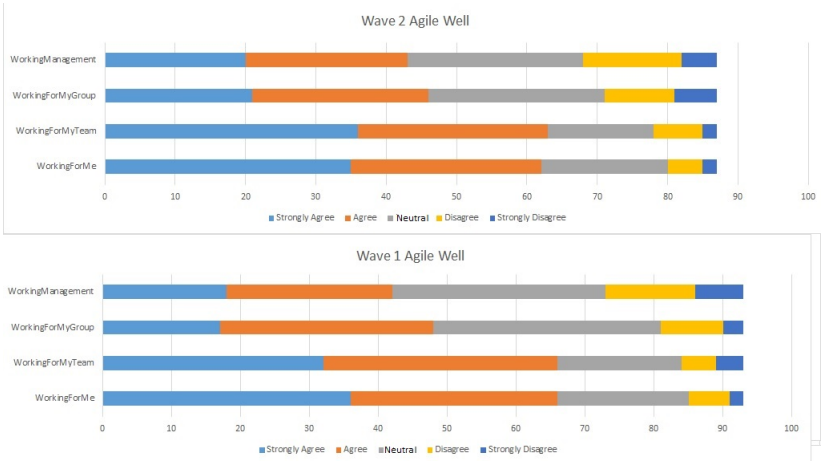

Figure 5.37: Comparison of the how well agile works in the two waves

Discussion: We see that the comparisons of rigor, perception, benefits and challenges do not show significant differences between the two waves even though there is significant difference in the experience of the respondents in the two waves. Thus, we can generalize these results and do not think that the results are skewed by opinions of respondents who feel strongly about agile development methodology.

5.11 Comparison of rigor based on experience

In order to compare the rigor of the practices based on experience we add up the scores for all practices for every respondent and called it the 'Rigor Score' for that respondent. We then plot this against the experience of that respondent. The plot in Figure 5.38 shows the distribution of rigor score vs experience

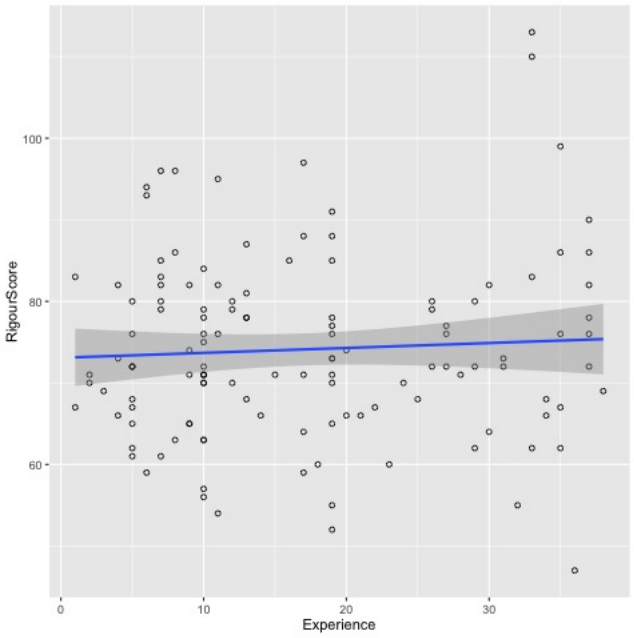

Figure 5.38: Comparison of the rigor based on experience

We then facet the plot by roles in Figure 5.39 to see if there is more rigorous practice based on the role of the individual.

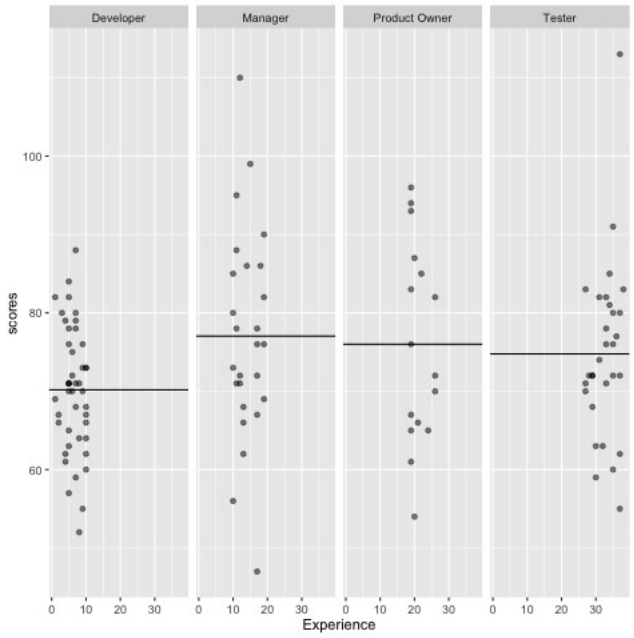

Figure 5.39: Comparison of the rigor facetted by role

We then facet the plot by past non-agile development experience(in figure 5.40) and find that respondents with only agile development experience practise agile development more rigorously.

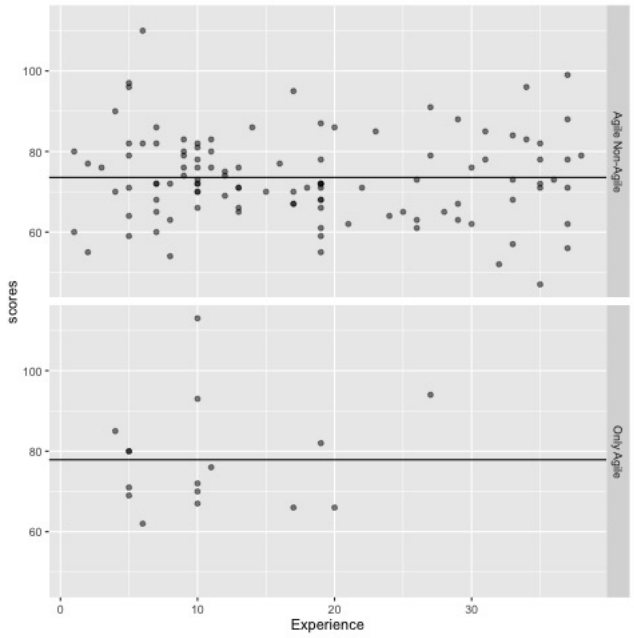

Figure 5.40: Comparison of the rigor facetted by non-agile work experience

Discussion: There is no significant impact of experience on the rigor of the agile development practices. Among all the roles, managers adhere to agile development the most. Respondents with only agile development experience practice agile development more rigorously than the respondents with past non-agile development experience.

5.12 Comparison of rigor based on team size

In order to analyze the impact of the team size on the rigor of agile practices we plot the rigor score of the respondent against their team size. The plot in Figure 5.41 shows the distribution of rigor score vs team size.

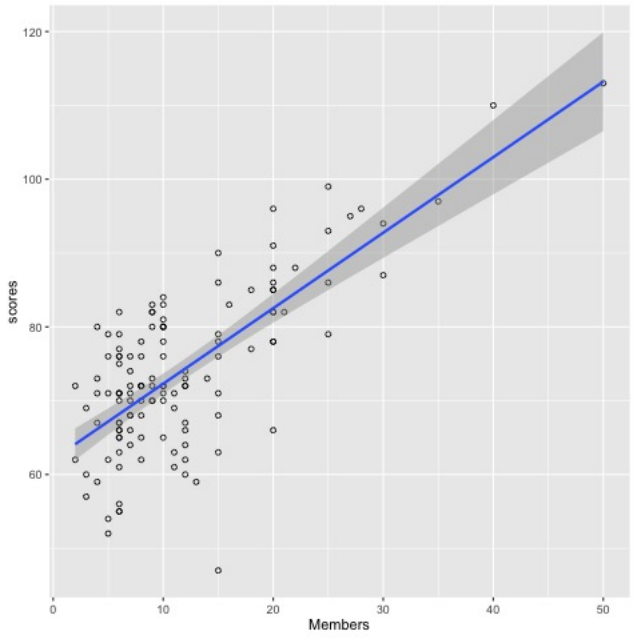

Figure 5.41: Comparison of rigor vs team size

It can be observed that the rigor scores are high for larger teams. The ideal team size is considered to be between five to nine [3]. However as seen in the plot this organization is able to practice agile techniques rigorously with larger teams as well.

We then facet the plot by roles in Figure 5.42 and find that among all the roles, managers practise agile development techniques with the highest rigor.

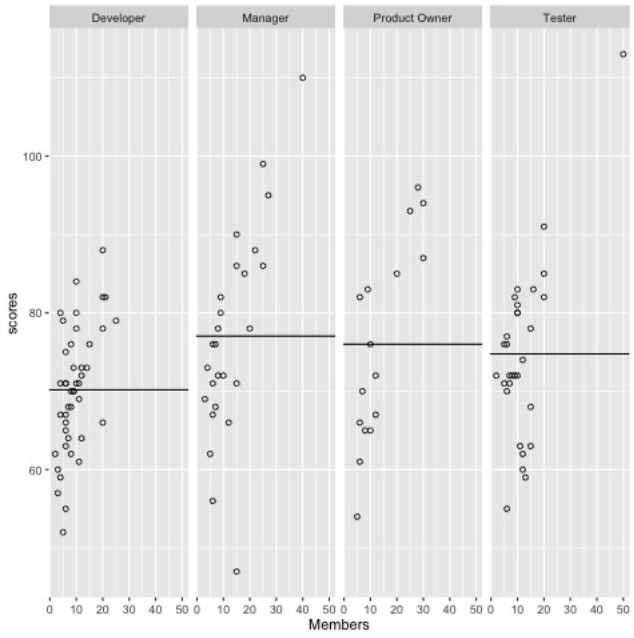

Figure 5.42: Comparison of the rigor facetted by role

We then facet the plot by past non-agile development experience in Figure 5.43 and find that respondents with only agile development experience practise agile development more rigorously.

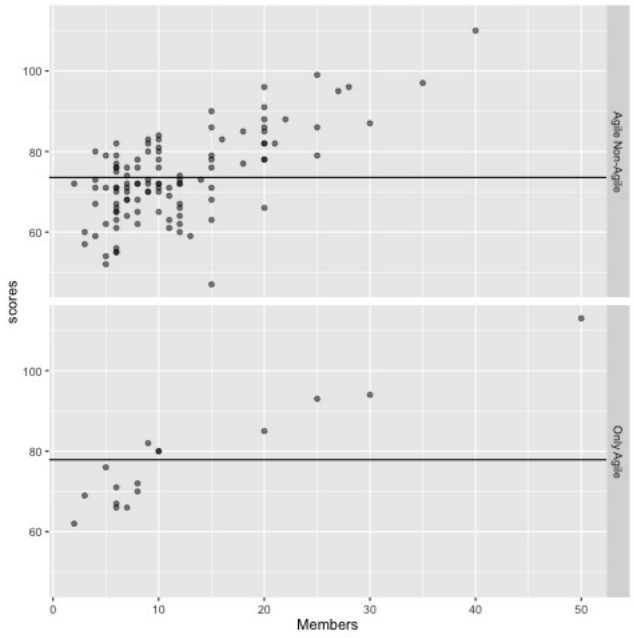

Figure 5.43: Comparison of the rigor facetted by non-agile work experience

Discussion: Larger teams tend to practise agile techniques more rigorously. It can be seen that respondents with only agile development experience practise agile development more rigorously than those with non-agile development experience. Among all the roles, managers practise agile development with maximum rigor.

5.13 Comparison of perception by experience

We divide the respondents in 4 experience groups.

- Group 1 - Experience less than 5 years
- Group 2 - Experience between 5 and 10 years
- Group 3 - Experience between 10 and 15 years

- Group 4 - Experience over 15 years.

The graph below shows no significant trends in the perception of agile development based on experience. This indicates that experience does not influence the perception of agile development.

We perform further analysis on the data as shown in the graph in Figure 5.44 and also perform inferential analysis using linear regression to see if there is a statistically significant impact on each of the perceptions (architecture, teams self-organizing, frequent integration, trust respect, collaborative, high energy work environment) based on the experience. The results of the linear regression t-values are converted to approximate p-values. The null hypothesis for the linear regression is that the experience has a zero co-efficient/no linear effect on the perception and this can be confirmed from the p-values below:

- Architecture - P-value - 0.67
- Collaborative - P-value - 0.05
- TeamsSelfOrganizing - P-value - 0.701
- FrequentIntegration - P-value - 0.318
- TrustRespect - P-value - 0.77
- HighEnergyWorkEnvironment - P-value - 0.22

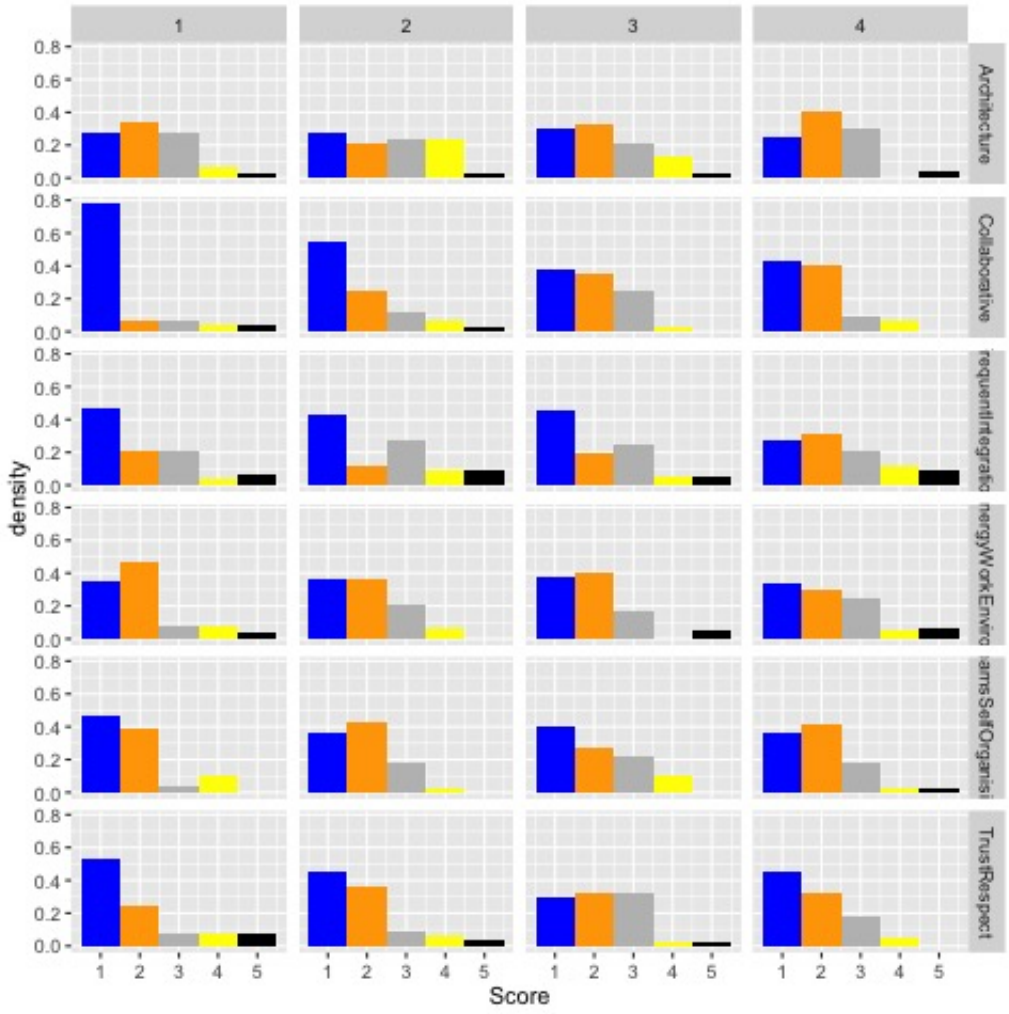

Figure 5.44: Comparison of the perception of agile development by experience

Discussion: Except for the perception of collaboration (which has a slightly negative impact of experience on being highly favourable) there is no significant impact of experience on the perception of agile development.

5.14 Grouping responses based on the respondents (program/project) level

The respondents of the survey using agile methods work at the program or project level. We divide the agile users into two groups based on which level they work at. We have 23 respondents working at the program level while 143 work at the project level. We analyze their responses on the perception of agile development and benefits and challenges of agile development. From the bar graphs in figure 5.44 we can see that the ratio of respondents who agree with the positive effects of agile development is higher in the project level respondents than the program level respondents.

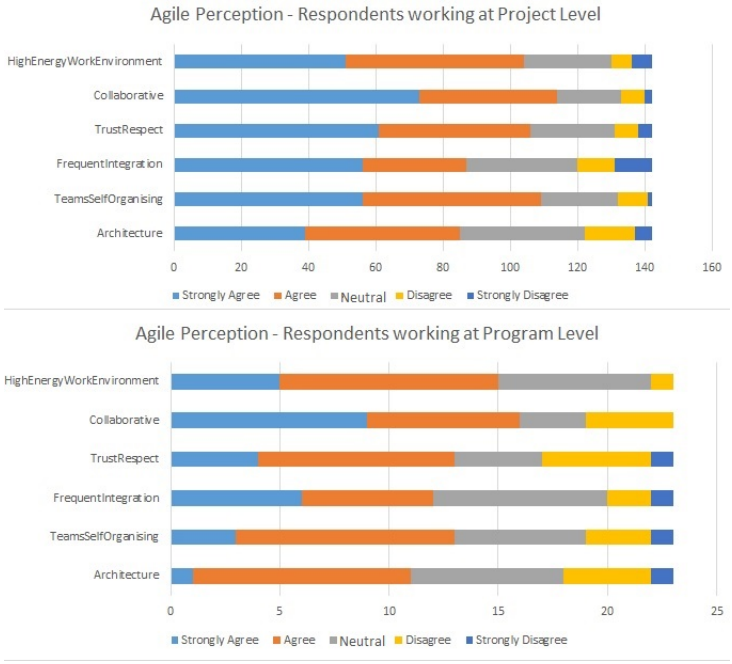

Figure 5.45: Comparison of perception of agile at different levels

From the bar graphs in Figure 5.45 and 5.46 we can see that the program level respondents agree with the positive impact of agile development more than the project level respondents.

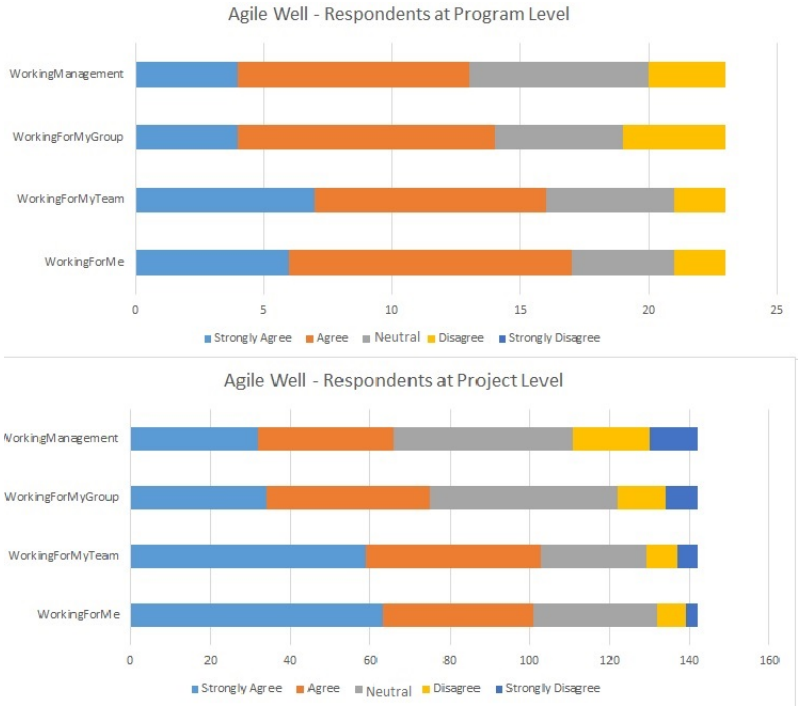

Figure 5.46: Comparison of how well agile works at different levels

The graph in Figure 5.47 shows the side by side comparison of benefits and challenges as perceived by project level and program level respondents. As the number of respondents in these two groups differed significantly we converted the counts to percentages and created the graphs below. There is no significant impact of level on the listed benefits and challenges of agile development.

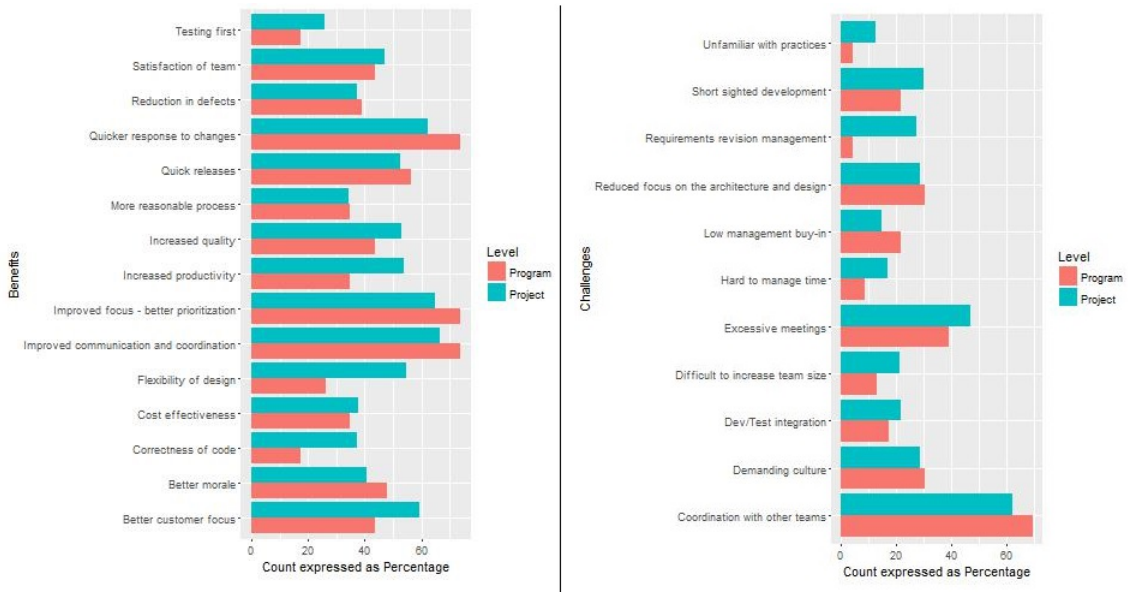

Figure 5.47: Comparison of benefits and challenges by respondents level

Discussion: From the analysis based on the level at which respondents work, we can confirm that the respondents at the project level perceive agile slightly more positively that those working at the program level. This could be mostly because of the issues associated with scaling agile. Agile can be easily implemented at the project level; however, scaling it is the real challenge. However since there are no significant differences, this is an indicator of success in the implementation of agile techniques at the program level.

5.15 Grouping responses based on agile training

For the comparison of perception of agile development based on training we divide the agile users in two groups: respondents with some training in agile techniques and those who have no training in agile techniques. Out of 181 responses we have 16

respondents with no training while 165 have some sort of training in agile methods. We analyze their responses about the perception of agile development, benefits and challenges. From the bar graphs in Figure 5.48 for trained and untrained respondents we can see that respondents in the trained group respond more positively than the untrained group.

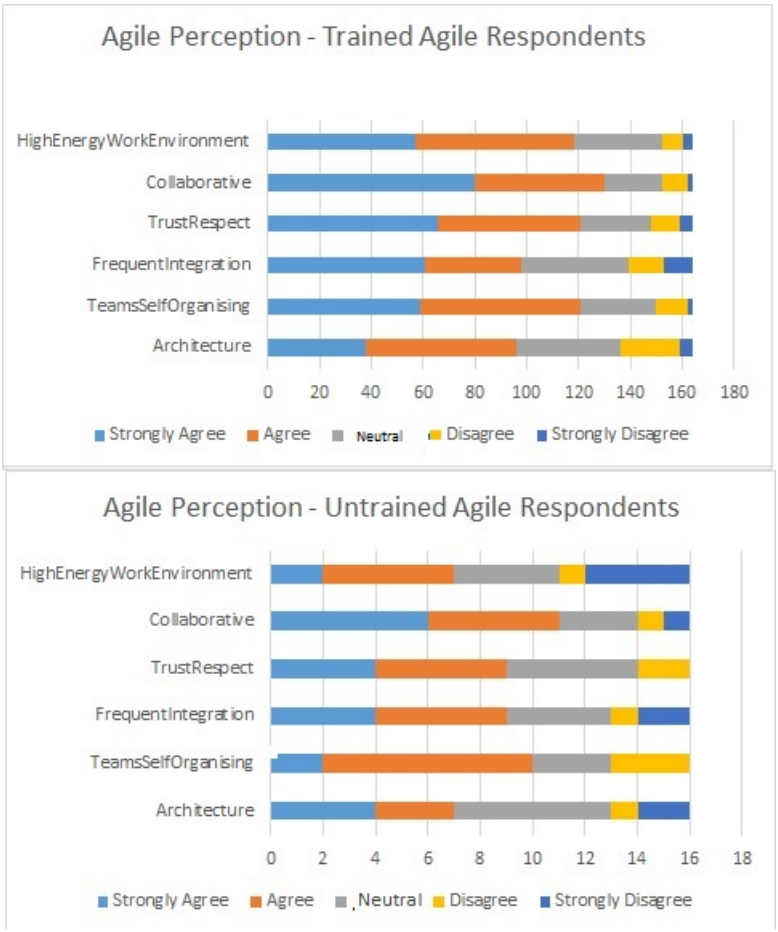

Figure 5.48: Comparison of the perception based on training

From the bar graphs in Figure 5.49 for how well agile works for the trained and untrained respondents we can see that respondents agreeing with the positive impact

of agile development is considerably higher in all categories in the trained group vs the untrained group.

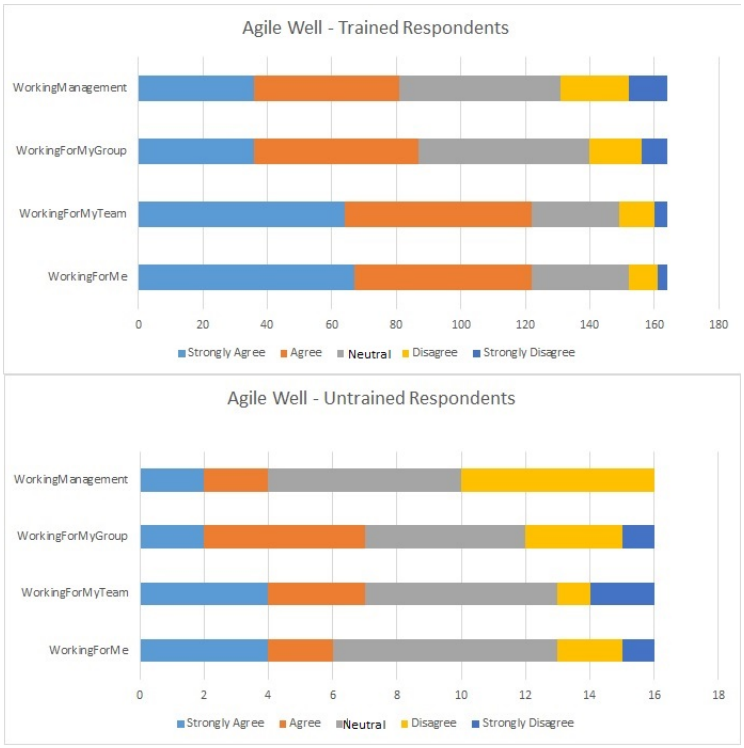

Figure 5.49: Comparison of how well agile works based on training

The graph in Figure 5.50 shows a side by side comparison of benefits as perceived by trained and untrained respondents. As the number of respondents in these two groups differed significantly we converted the counts to percentages and created the graphs below. It can be clearly seen that the respondents with some sort of training have more benefits to list that the ones without any training. This certainly asserts the importance of training in agile methods to improve the success in implementation. This graph also shows a side by side comparison of the challenges as perceived by trained and untrained respondents. As the number of respondents in these two groups differed significantly we convert the counts to percentages and create the

graphs below. It can be clearly seen that the respondents without any training have more challenges than the respondents with some training in agile development.

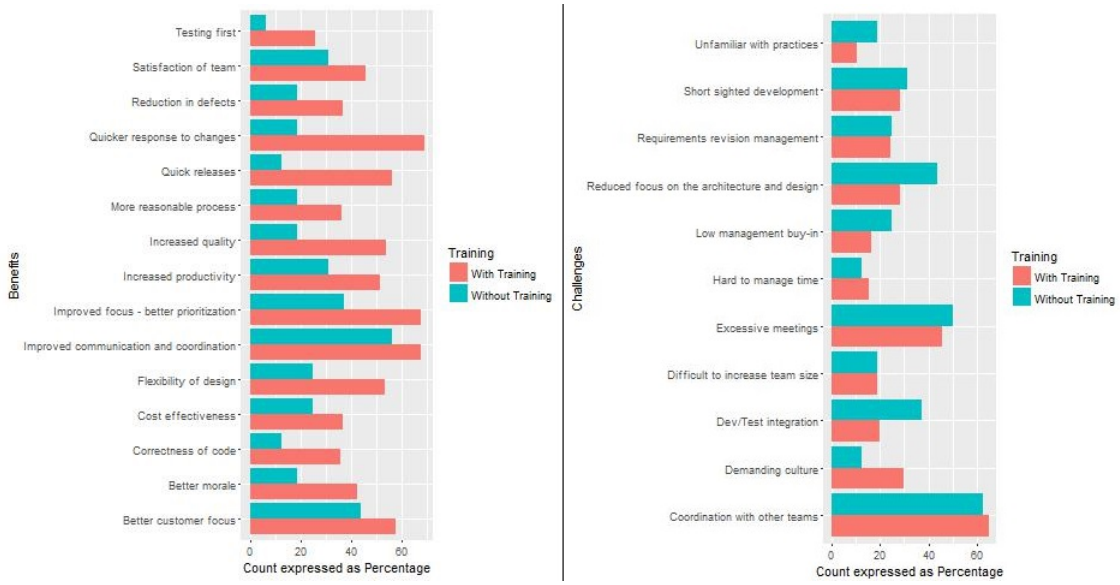

Figure 5.50: Comparison of the benefits and challenges based on training

Comparison of Experience and Training We ask the respondents about the trainings they have done in agile techniques and based on the their responses we create a data set with the count of trainings and plot those against the respondent experience. The graph below shows the distribution of the number of trainings completed against experience. It can be seen in Figure 5.51 that the number of trainings done increases with the experience, however it does not increase significantly.

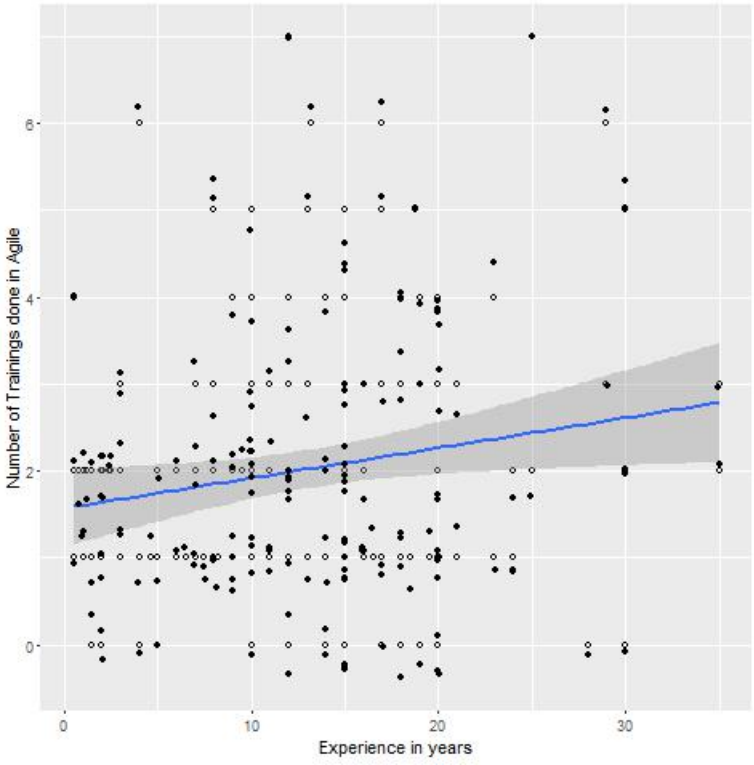

Figure 5.51: Comparison of the trainings and experience

Discussion: From the analysis it can be clearly seen that training has a positive impact on the respondents perception of agile development and the rigor of practising agile practices. Thus, conducting more trainings will positively impact the productivity and respondents comfort level with agile development in the organization potentially leading to increased productivity.

5.16 Grouping based on non-agile experience

We divide the respondent population in two groups based whether they have past non-agile work experience. We analyze the data to see if there are any differences in the perception of agile techniques, perceived benefits and challenges. The graph in

Figure 5.52 shows that respondents with only agile development experience have a more positive perception of agile development.

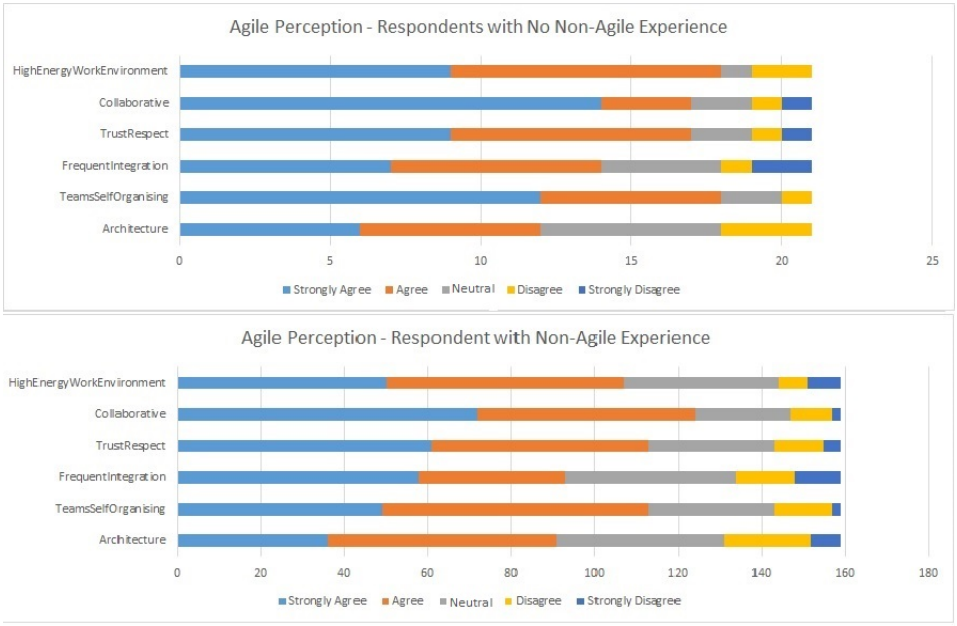

Figure 5.52: Comparison of the perception based on past experience

The graph in Figure 5.53 shows that respondents with only agile development experience have more positive experience about how agile development is working for them.

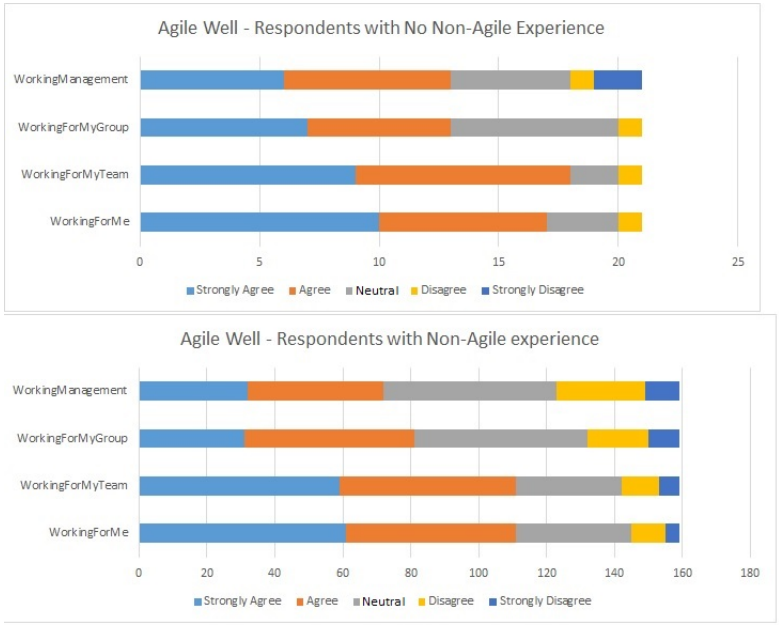

Figure 5.53: Comparison of the perception based on experience

The graph in Figure 5.54 shows the side by side comparison of the benefits and challenges as perceived by both groups of respondents. As the number of respondents in these two groups differs significantly, we convert the counts to percentages and created the graphs below. It can be clearly seen that the respondents with only agile experience list significantly more benefits that others. No significant differences can be seen in the challenges between the two groups.

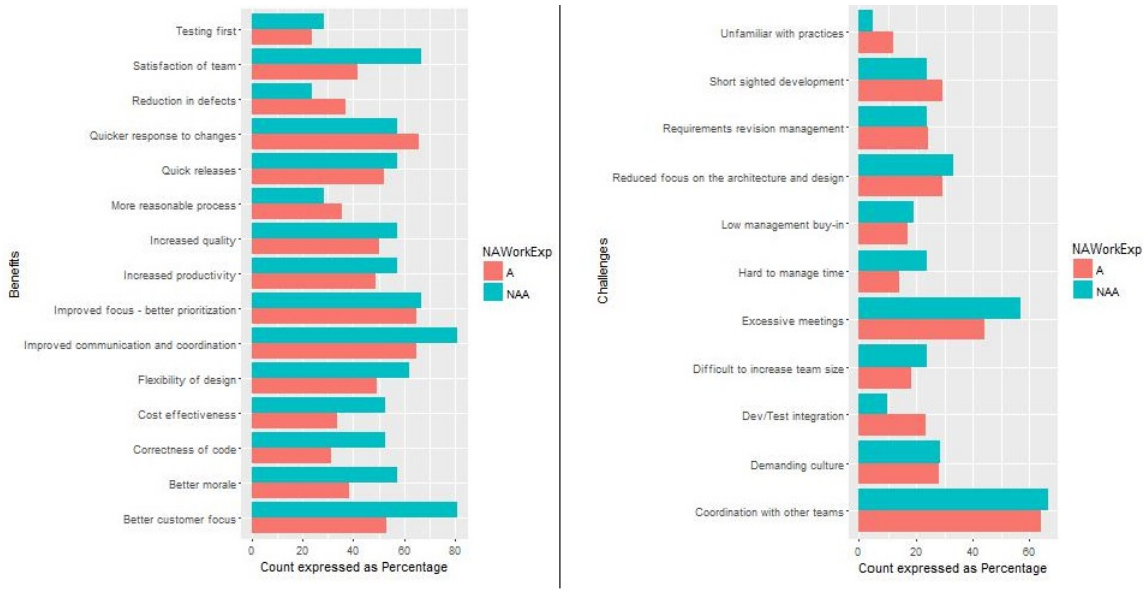

Figure 5.54: Comparison of benefits and challenges based on experience

Discussion: From the graphical analysis above we can see that the respondents in the group with only agile experience have a more positive perception of agile development and practise agile techniques with higher rigor.

Chapter 6

Conclusions and Future Work

6.1 Conclusion

The conclusions of our research are:

- There is an overall positive perception of agile across the respondent population. The agile implementation at the team and program level is extremely effective.
- Most of the respondents using agile development are willing to continue with the agile development methodology while majority of the respondents using non-agile development methods are willing to switch to agile development methods.
- Agile development works well at the individual level and team level. It is less effective at the group level and interactions with management are challenging.

- The total count of listed benefits is over 50 percent higher than the total count of challenges with agile development confirming the respondents positive experience towards agile development.

- Experience does not have an impact on the rigor or the perception of agile development practices.

- Respondents who have worked only on agile development perceive it more positively and tend to practice more rigorously than the respondents who have some past non-agile development experience.

- It is observed that the team size has an impact on the rigor of practising agile techniques. Larger teams implement agile techniques with higher rigor.

- When respondents are grouped based on their level, it is observed that agile development at the project level is slightly more successful than at the program level.

- An analysis of the impact of training on the perception of agile development and the rigor of practising agile techniques shows that the respondents with training in agile methods are more rigorous and perceive agile techniques more positively.

Recommendations based on the findings are:

- Training has a significant impact on the perception and rigor of agile development and thus, there needs to be an increased focus on trainings in agile methods

- The top challenge of co-ordination between teams stresses the need for reducing dependencies between teams. This can be done by improving communication between teams during the program and team during backlog creation
- Test driven development is practiced with low rigor and testing is not seen as a benefit of agile development practices. Hence there needs to be more disciplined adherence of test driven development for higher quality of development
- There needs to be more focus on the technical aspects of architectural stability and frequent integration to further strengthen the implementation of agile methodology
- The program level implementation of agile development can be made more rigorous with introduction of increased practices across the program level and more disciplined adherence.

6.2 Future Work

This survey is performed in the Higher Education division of Pearson Education.

- In the current survey there is insufficient data for analyzing the impact of location on the implementation of agile practices in the organization. As only a few locations have been represented heavily the results are skewed towards their specific implementation methods. In the future we would like to analyze the impact of the location.
- We would like to analyze the impact of changes done based on the insights delivered from this analysis for the program level practices. Also analyze different practices (if any) followed at the program level across different divisions.

- We analyze the project and program level practices in this survey, we would like to analyze the portfolio level practices. Thus, analysing the state of agile implementation at the higher level.

- Lastly we would like to perform a similar survey in other large scale software development organizations that implement agile methods at scale and do a comparative analysis of the factors leading to the success or failure of the implementation.

Bibliography

[1] Pearson Education. Product creation framework. `https://neo.pearson.com/groups/product-creation-framework`. Accessed:05/10/2016.

[2] Satish Thatte. Agile management blog. `https://blogs.versionone.com/agile_management/2013/10/14/scalable-agile-estimation-and-normalization-of-story-points-introduction-and-` Accessed: 04/01/2016.

[3] Dean Leffingwell. *Scaled Agile Framework*, 2009 (accessed February 3, 2015).

[4] Barry Boehm. Get ready for agile methods, with care. *Computer*, 35(1):64–69, 2002.

[5] Ken Schwaber and Mike Beedle. Scrum: Agile software development, 2002.

[6] Alistair Cockburn. *Crystal clear: a human-powered methodology for small teams*. Pearson Education, 2004.

[7] Kent Beck. *Extreme programming explained: embrace change*. addison-wesley professional, 2000.

[8] Girish Kumar and Pradeep Kumar Bhatia. Comparative analysis of software engineering models from traditional to modern methodologies. In *Advanced Computing & Communication Technologies (ACCT), 2014 Fourth International Conference on*, pages 189–196. IEEE, 2014.

[9] ABM Moniruzzaman and Dr Syed Akhter Hossain. Comparative study on agile software development methodologies. *arXiv preprint arXiv:1307.3356*, 2013.

[10] Yu Beng Leau, Wooi Khong Loo, Wai Yip Tham, and Soo Fun Tan. Software development life cycle agile vs traditional approaches. In *International Conference on Information and Network Technology*, volume 37, pages 162–167, 2012.

[11] Preeti Rai and Saru Dhir. Impact of different methodologies in software development process.

[12] S Balaji and M Sundararajan Murugaiyan. Waterfall vs. v-model vs. agile: A comparative study on sdlc. *International Journal of Information Technology and Business Management*, 2(1):26–30, 2012.

[13] Gaurav Kumar and Pradeep Kumar Bhatia. Impact of agile methodology on software development process. *International Journal of Computer Technology and Electronics Engineering (IJCTEE)*, 2(4), 2012.

[14] Harleen K Flora and Swati V Chande. A systematic study on agile software development methodologies and practices. *International Journal of Computer Science and Information Technologies*, 5(3):3626–3637, 2014.

[15] Ying Wang, Dayong Sang, and Wujie Xie. Analysis on agile software development methods from the view of informationalization supply chain management. In *Intelligent Information Technology Application Workshops, 2009. IITAW'09. Third International Symposium on*, pages 219–222. IEEE, 2009.

[16] Pekka Abrahamsson and Juha Koskela. Extreme programming: A survey of empirical data from a controlled case study. In *Empirical Software Engineering, 2004. ISESE'04. Proceedings. 2004 International Symposium on*, pages 73–82. IEEE, 2004.

[17] Laurie Williams, William Krebs, Lucas Layman, A Antón, and Pekka Abrahamsson. Toward a framework for evaluating extreme programming. *Empirical Assessment in Software Eng.(EASE)*, pages 11–20, 2004.

[18] Grigori Melnik and Frank Maurer. A cross-program investigation of students' perceptions of agile methods. In *Software Engineering, 2005. ICSE 2005. Proceedings. 27th International Conference on*, pages 481–488. IEEE, 2005.

[19] Jeffrey Carver, Letizia Jaccheri, Sandro Morasca, and Forrest Shull. Issues in using students in empirical studies in software engineering education. In *Software Metrics Symposium, 2003. Proceedings. Ninth International*, pages 239–249. IEEE, 2003.

[20] Helen Sharp and Hugh Robinson. An ethnographic study of xp practice. *Empirical Software Engineering*, 9(4):353–375, 2004.

[21] Erik Arisholm, Hans Gallis, Tore Dybă, and Dag IK Sjoberg. Evaluating pair programming with respect to system complexity and programmer expertise. *Software Engineering, IEEE Transactions on*, 33(2):65–86, 2007.

[22] E Michael Maximilien and Laurie Williams. Assessing test-driven development at ibm. In *Software Engineering, 2003. Proceedings. 25th International Conference on*, pages 564–569. IEEE, 2003.

[23] Tsun Chow and Dac-Buu Cao. A survey study of critical success factors in agile software projects. *Journal of Systems and Software*, 81(6):961–971, 2008.

[24] Bruno Cartaxo, Allan Araujo, Antonio Sa Barreto, and Sérgio Soares. The impact of scrum on customer satisfaction: An empirical study. In *Software Engineering (SBES), 2013 27th Brazilian Symposium on*, pages 129–136. IEEE, 2013.

[25] Tore Dybå and Torgeir Dingsøyr. Empirical studies of agile software development: A systematic review. *Information and software technology*, 50(9):833–859, 2008.

[26] ABM Moniruzzaman and Dr Syed Akhter Hossain. Comparative study on agile software development methodologies. *arXiv preprint arXiv:1307.3356*, 2013.

[27] Francisco J Pino, Oscar Pedreira, Félix García, Miguel Rodríguez Luaces, and Mario Piattini. Using scrum to guide the execution of software process improvement in small organizations. *Journal of Systems and Software*, 83(10):1662–1677, 2010.

[28] Bernadette Murphy, Christian Bird, Thomas Zimmermann, Laurie Williams, Nachiappan Nagappan, and Andrew Begel. Have agile techniques been the

silver bullet for software development at microsoft? In *Empirical Software Engineering and Measurement, 2013 ACM/IEEE International Symposium on*, pages 75–84. IEEE, 2013.

[29] M. Laanti. Implementing program model with agile principles in a large software development organization. In *Computer Software and Applications, 2008. COMPSAC '08. 32nd Annual IEEE International*, pages 1383–1391, July 2008.

[30] Mario Pichler, Hildegard Rumetshofer, and Wilhelm Wahler. Agile requirements engineering for a social insurance for occupational risks organization: A case study. In *Requirements Engineering, 14th IEEE International Conference*, pages 251–256. IEEE, 2006.

[31] Nils Brede Moe, Torgeir Dingsøyr, and Tore Dybå. A teamwork model for understanding an agile team: A case study of a scrum project. *Information and Software Technology*, 52(5):480–491, 2010.

[32] A. Begel and N. Nagappan. Usage and perceptions of agile software development in an industrial context: An exploratory study. In *Empirical Software Engineering and Measurement, 2007. ESEM 2007. First International Symposium on*, pages 255–264, Sept 2007.

[33] Ahmed Sidky, James Arthur, and Shawn Bohner. A disciplined approach to adopting agile practices: the agile adoption framework. *Innovations in systems and software engineering*, 3(3):203–216, 2007.

[34] Ben Kovitz. Hidden skills that support phased and agile requirements engineering. *Requirements Engineering*, 8(2):135–141, 2003.

[35] Asif Qumer and Brian Henderson-Sellers. An evaluation of the degree of agility in six agile methods and its applicability for method engineering. *Information and software technology*, 50(4):280–295, 2008.

[36] Donald J Reifer, Frank Maurer, and Hakan Erdogmus. Scaling agile methods. *Software, IEEE*, 20(4):12–14, 2003.

[37] Hakan Erdogmus. Cost-effectiveness indicator for software development. 2007.

[38] A Qumer and B Henderson-Sellers. A framework to support the evaluation, adoption and improvement of agile methods in practice. *Quality control and applied statistics*, 54(4):391–393, 2009.

[39] Sridhar Nerur, RadhaKanta Mahapatra, and George Mangalaraj. Challenges of migrating to agile methodologies. *Communications of the ACM*, 48(5):72–78, 2005.

[40] Barry Boehm and Richard Turner. Management challenges to implementing agile processes in traditional development organizations. *Software, ieee*, 22(5):30–39, 2005.

[41] Wikipedia. Software development process — wikipedia, the free encyclopedia, 2016. [Online; accessed 26-March-2016].

[42] Kai Petersen, Claes Wohlin, and Dejan Baca. The waterfall model in large-scale development. In *Product-focused software process improvement*, pages 386–400. Springer, 2009.

[43] AgileManifesto. Agilemanifesto, 2001.

[44] Pekka Abrahamsson, Outi Salo, Jussi Ronkainen, and Juhani Warsta. Agile software development methods: Review and analysis, 2002.

[45] John Erickson, Kalle Lyytinen, and Keng Siau. Agile modeling, agile software development, and extreme programming: the state of research. *Journal of database Management*, 16(4):88, 2005.

[46] Frauke Paetsch, Armin Eberlein, and Frank Maurer. Requirements engineering and agile software development. In *null*, page 308. IEEE, 2003.

[47] Granville G Miller. The characteristics of agile software processes. In *tools*, page 0385. IEEE, 2001.

[48] Ken Schwaber, Jeff Sutherland, and Mike Beedle. The definitive guide to scrum: The rules of the game. *Recuperado de: http://www. scrumguides. org/docs/scrumguide/v1/scrum-guide-us. pdf*, 2013.

[49] Dan Radigan. Agile development scrum - atlassian. https://www.atlassian.com/agile/scrum.

[50] Christian Beck. Scrum roles. http://www.agile42.com/en/agile-info-center/scrum-roles/. Accessed: 04/01/2016.

[51] Victor Szalvay. Scrum terminology. https://www.scrumalliance.org/community/articles/2007/march/glossary-of-scrum-terms. Accessed: 04/01/2016.

[52] Kanban development - atlassian. https://www.atlassian.com/agile/kanban. Accessed: 04/01/2016.

[53] Dan Radigan. Scrum delivery vehicles. https://www.atlassian.com/agile/delivery-vehicles. Accessed: 04/01/2016.