ANALYSIS OF REQUIREMENTS PRIORITIZATION IN DISTRIBUTED SCRUM FOR REDUCING SOFTWARE FAILURE

By

MUHAMMAD SOHAIB

NATIONAL UNIVERSITY OF MODERN LANGUAGES

ISLAMABAD

2024

Analysis of Requirements Prioritization in Distributed Scrum for Reducing Software Failure

By

MUHAMMAD SOHAIB

BSIT, PIR MEHR ALI SHAH ARID AGRICULTURE UNIVERSITY, RAWALPINDI, 2020

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

In **Software Engineering**

To
FACULTY OF ENGINEERING & COMPUTING

NATIONAL UNIVERSITY OF MODERN LANGUAGES ISLAMABAD

NATIONAL UNIUVERSITY OF MODERN LANGUAGES   FACULTY OF ENGINEERIING & COMPUTER SCIENCE

THESIS AND DEFENSE APPROVAL FORM

The undersigned certify that they have read the following thesis, examined the defense, are satisfied with overall exam performance, and recommend the thesis to the Faculty of Engineering and Computing for acceptance.

Thesis Title: ANALYSIS OF REQUIREMENTS PRIORITIZATION IN DISTRIBUTED SCRUM FOR REDUCING SOFTWARE FAILURE

Submitted by: Muhammad Sohaib   **Registration #:** 7 MS/SE/F20

Master of Science in Software Engineering
Title of the Degree

Software Engineering
Name of Discipline

Dr. Basit Shahzad Name of Research Supervisor	Signature of Research Supervisor
Dr. Jaweria Kanwal Name of Co-Supervisor	Signature of Co-Supervisor
Dr. Sumaira Nazir Name of Head of Department SE	Signature of Head of Department SE
Dr. Noman Malik Name of Dean (FE&C)	Signature of Dean (FE&C)

September, 2024
Date

AUTHOR’S DECLARATION

I Muhammad Sohaib

Son of Zahid Maqsood

Registration # 7 MS/SE/F20

Discipline Software Engineering

Candidate of **Master of Science in Software Engineering (MSSE)** at the National University of Modern Languages do hereby declare that the thesis **Analysis of Requirements Prioritization in Distributed Scrum for Reducing Software Failure** submitted by me in partial fulfillment of MSSE degree, is my original work, and has not been submitted or published earlier. I also solemnly declare that it shall not, in the future, be submitted by me for obtaining any other degree from this or any other university or institution. I also understand that if evidence of plagiarism is found in my thesis/dissertation at any stage, even after the award of a degree, the work may be canceled and the degree revoked.

____

Signature of Candidate

Muhammad Sohaib

Name of Candidate

September, 2024

Date

ABSTRACT

Analysis of Requirements Prioritization in Distributed Scrum for Reducing Software Failure

Requirement Prioritization is an essential part of the software development life cycle. The success and failure of software heavily depend on requirement prioritization. Scrum gaining popularity in software development to get mutual benefits of scrum and distributed team environment. All the stakeholders in distributed Scrum are usually distributed by time and geography, so prioritization of requirements becomes challenging. Therefore, in this comprehensive research, requirement prioritization is navigated in the context of distributed Scrum to reduce software failure. The study started by carefully identifying the problems from a thorough literature review with practical analysis. Then validating the identified challenges found by the literature review with the help of a survey. Reviewers were distributed Scrum practitioners. Later on, interviews were conducted to find the possible solutions to the challenges. Their extensive experience not only validates the validity of the challenges that have been identified but also provides our study with a more profound comprehension of the practical implications. Building on this foundation, a set of guidelines were proposed to address the challenges of requirement prioritization in distributed scrum to reduce software failure. These solutions, which provide an organized framework that practitioners and organizations can easily use, are the result of the collective experience of the agile community. The proposed solution was rigorously validated by the Focus Group to strengthen its applicability and practicality. This collaborative refinement ensures that our guidelines align seamlessly with the requirement prioritization challenges faced by distributed Scrum teams.

The research's output is a well-balanced combination of theoretical understanding and real-world experience. This comprehensive method not only adds to the body of knowledge in the field of requirement prioritization research, but it also offers organizations and practitioners a useful road map for negotiating the challenges associated with distributed scrum and software failure in requirement prioritization.

TABLE OF CONTENTS

List of Figures

List of Tables

List of Abbreviations

ASD ……………	*Agile Software Development*
RP…………….	*Requirement Prioritization*
LR…………….	*Literature Review*
IEEE ……………	*Institute of Electrical and Electronics Engineers*
ACM ……………	*Association for Computing Machinery*

CHAPTER 1

INTRODUCTION

1.1. Introduction

The term "software engineering" was first used in 1968 at a NATO-sponsored scientific meeting. For over 40 years, this concept has been gradually diversified and improved throughout programming environments. A request for a new class of programmers to become the engineers of software development was made in response to the software crisis, which was officially characterized as such in a 1968 NATO conference. Many people believe that the term software engineering originated at the NATO Conference on Software Engineering in 1968 [24] [25] [26]. Software engineering is a set of approaches, techniques, and disciplines for planning, developing, maintaining, and deploying software support in a mass-production environment. It is the engineering side of the field, and it encompasses all parts of the creation of SS, from the formulation of requirements to product development, maintenance, and removal from service. Software engineering is a set of methods and tools for programming, making plans, and team process management in the production of computer software products, as well as methods for measuring and estimating the interoperability of their various characteristics with customer requirements. Base items are utilized in production, as are automated activities comparable to those used in production. Software engineering is a scientific and engineering field that focuses on the development of software products [25].

The process of collecting, describing, negotiating, prioritizing, specifying, verifying, and managing a system's needs is known as requirement engineering. The RE process includes identifying stakeholder needs, comprehending the context of requirements, modelling, negotiating, validating, documenting, and managing these requirements. The requirement engineering phase becomes more difficult for scalable software's, the collaboration of software's to its environment and the growing globalization of software development. [1].

Prioritization of requirements is a critical decision-making process used in requirements engineering and software development to identify which requirements should be implemented and delivered. There are many ways to prioritize requirements using different requirements prioritization criteria, including approaches, techniques, and tools. Prioritizing requirements, deciding which are the most important involves making decisions based on one or more criteria, such as budget, time constraints, technical constraints that are development cost and risk, business aspects that are market competition and regulations, customer satisfaction, or business value [2]. Ignoring the requirement prioritization activity could lead to different challenges and one of the main factor is software failure. The delivery of non-significant (i.e., "nice-to-have" versus "must-have") requirements to customers may cause this problem and ultimately result in project failure. The main reason behind the failure of software development is lack of the customer involvement at the time of requirement prioritization the result of which leads to lack of customer expectations related to software. When stakeholders are actively involved, requirement prioritization may help reduce the risk of project failure and increase the success rate of projects. The chance of software development success increases because requirements are prioritized on the preference of the customer [10].

Agile is a software development methodology that is intended to manage and assist in the incremental and iterative development of business systems in environments where change is frequent. Agile methodologies focuses on small empowered teams and encourage always customer participation. An agile method focuses on creating good working software quickly by relying on communication, feedback, learning, and regular meetings rather than modelling and documentation. To attain these objectives, agile methods modify conventional software development techniques [55].

Scrum is a procedural framework for managing work on complicated products. Scrum is not a method, process, or approach. Scrum is rather, a structured framework that employs multiple techniques. Scrum gives you a comparative effectiveness of work methods and product management, enabling to improve the team, the product, and the working environment over time. Small teams are the foundation of Scrum. The team as a whole is highly adaptable. These capabilities are still being used in single, multiple, numerous, and networked teams that produce, release, and run software. Through advanced development architectures and target release environments, they collaborate and interoperate [56].

Distributed scrum plays an important role in many aspects of software development. Which includes handling distributed teams in an organized manner, connecting geographically separated teams, outsourcing development partially, and maintaining project lifecycle. However, it is observed that distributed scrum may also increase chances for lack of communication, coordination, and control in terms of the current industrial environment [4].

The product backlog is a place where the product owner can add further requirements. However, in a distributed environment it is a real challenge to obtain full requirements due to lack of communication between the stakeholders. Requirements can be misinterpreted and are subsequently incorrectly prioritized. The prioritization of product backlog items often does not take place systematically. The product owner is usually the only responsible person for requirement prioritization. Who needs to be competent and informed well to prioritize the requirements in a meaningful way. Developers usually have great competency and technical skills to take part in the prioritization however they actually have very little involvement in the process [5].

1.2. Problem Statement

Requirement prioritization is important in software development for project management, budgeting and time constraints. If requirements are not properly prioritized it could lead to software failure [10]. For the appropriate requirement prioritization each stakeholder and team member should be the part of the process whereas, in Scrum, the Product Owner is the only person who is responsible for the prioritization of requirements [6] [60]. Due to the little involvement of all stakeholders, requirement prioritization become a challenging task in Scrum. The problem becomes more challenging in case of distributed Scrum because the distance may create communication barriers among stakeholders which may lead to incorrectly prioritized requirements and software failure eventually [5]. In the literature, the problem of requirements prioritization in distributed Scrum is at initial research stage. In this regard, there is a need to conduct an in-depth study to identify the challenges faced by the industry in requirement prioritization in distributed Scrum the study may suggest guidelines to address these challenges.

1.3. Research Objectives

- To identify the current challenges regarding requirement prioritization in distributed Scrum that could lead to software failure.
- To develop the possible guidelines for solving the requirement prioritization challenges in distributed Scrum.

1.4. Research Questions

RQ1: What are the challenges regarding requirement prioritization in distributed Scrum that could lead to software failure?

RQ2: What could be the possible guideline for solving the requirement prioritization challenges in distributed Scrum?

1.5. Scope of Study

The following is the scope of our research:

- This research is based on challenges related to requirement prioritization in distributed Scrum.
- The challenges faced by the distributed Scrum team during requirement prioritization are precisely identified through a literature review.
- Conducted LR of the past 10 year's papers that are from 2013-2023.
- Conducted a survey to identify challenges related to requirement prioritization in distributed Scrum.
- Proposed guidelines to mitigate challenges identified through the industrial survey.
- Performed focus group interviews to verify the proposed mitigation strategies

- The compiled results of the study will provide a roadmap for the rectification of the shortcomings of requirement prioritization in distributed Scrum to overcome the factor of software failure.

1.6. Thesis Organization

The research thesis consists of 5 chapters. The first chapter gives the overview of the whole thesis. It comprises of introduction of requirement prioritization challenges in distributed Scrum then the problem statement is discussed which is the main concern of the thesis. After that research question, research objective and scope of the study are described. This chapter is the foundation of the thesis.

The second chapter is a literature review which discusses the existing literature related to the requirement prioritization challenges in distributed Scrum. This chapter also discusses the research gap for the requirement prioritization. The third chapter is “research methodology” in which different methodologies are discussed through which the research study is carried out like qualitative, quantitative, and mixed method approach. In chapter 4 the results are analyzed through different techniques and a final discussion of the thesis is done in which the result is concluded. Chapter 5 is the conclusion, limitations and future work of the thesis.

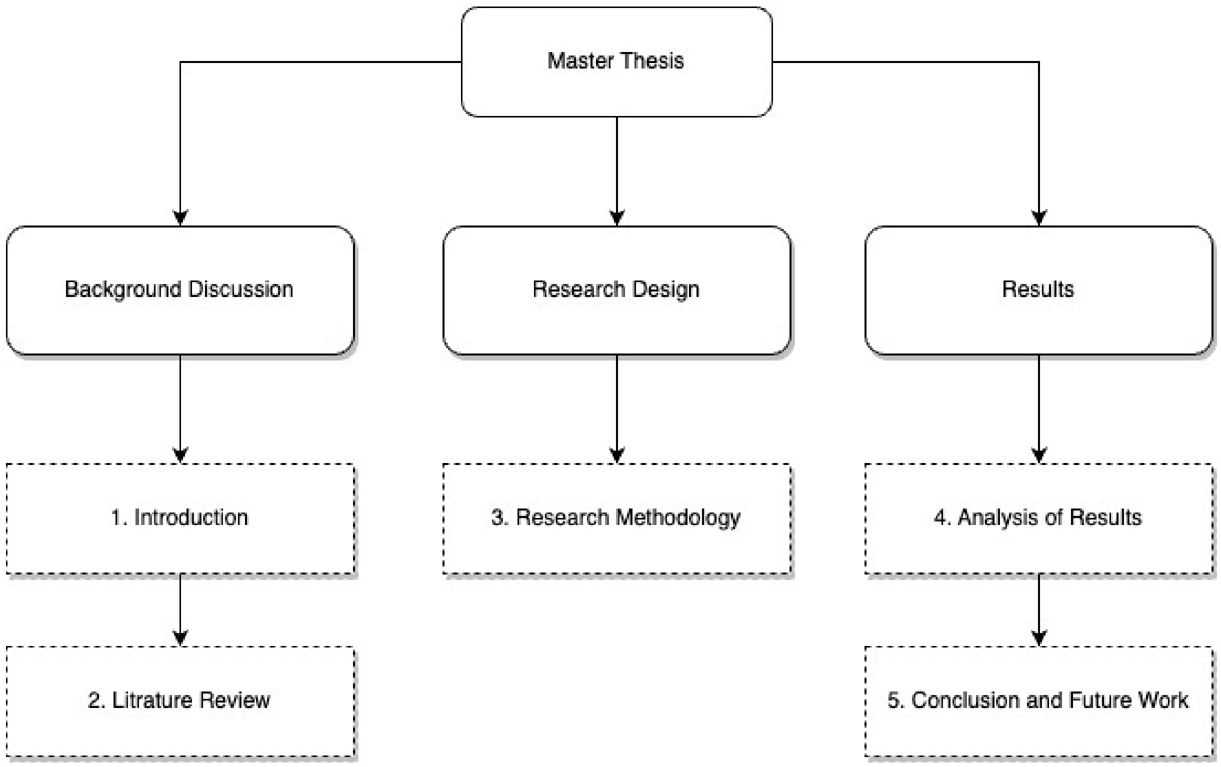

Figure 1.1: Thesis Organization

1.7. Summary

The first chapter discusses the difficulties in prioritizing requirements in teams that use distributed Scrum. Scrum is a way of working, and sometimes, the team is in different places. The challenges of this situation are closely examined. The main questions to be answered are introduced, and the research's goals are outlined. The specific areas under investigation are explained.

The chapter also outlines the research plan, which tells each step in detail. This chapter serves as the foundation of a building, helping understand the complexities of prioritizing requirements in Scrum when the team is in different locations. The structured research plan also sets a path for the next chapters, guiding the journey of finding ways to reduce problems in software when using distributed Scrum.

CHAPTER 2

LITERATURE REVIEW

2.1. Introduction

This chapter explains the challenges related to requirement prioritization in disturbed Scrum. Requirement prioritization is an essential part of the software development process, however, there are challenges in prioritizing them appropriately. It is observed that requirement prioritization becomes more challenging in distributed Scrum due to lack of communication, coordination and control. A literature review was conducted to find out the challenges related to research questions.

2.2. Background

Software engineering is a different concept whose purpose is to provide high-quality software at a reasonable cost. As the problems that its practitioners face continue to expand their abilities, this field is continuously evolving. One of the sources of dissatisfaction with the software community in the 1960s and 1970s was the pressure to deliver more with less. There were numerous examples of attempts to build software that resulted in massive budget overruns, failed to deliver the function promised, and failed unexpectedly and disastrously. "Why isn't software production as predictable as engineering?" was common. At least in part, the solution was to try to establish a software engineering discipline to address these issues. Software reliability, software management, and developer productivity have all been thought to be three basic areas of interest in the field of software engineering [24]. Cost, timeline, complication, functionality, performance, reliability, and security, as well as legal and ethical forces, must all be balanced in software engineering [26].

The first and most crucial step in the software development process is requirement engineering. Good requirement engineering processes are important to the achievement of any software project.

The goal of requirement engineering approaches is to elicit all intended user demands and document them for future reference and understanding. Elicitation is the process of obtaining user requirements and software system limitations. It determines whether a software project succeeds or fails [28]. A branch of software engineering called requirements engineering addresses the objectives, features, and limitations of software systems development. It's also interested in how these parameters relate to exact software behavior standards, as well as how they've changed over time and across software families. The elicitation of requirements is likely the most commonly regarded activity as the first stage in the RE process. To prevent the implication that requirements may be gathered simply by asking the proper questions, the term "elicitation" is chosen over "capture." RE is a multi-disciplinary activity that employs a wide range of methodologies and technologies at various stages of development and for various types of application domains. Methods are a systematic way of merging multiple approaches and notations, and method engineering is crucial in developing the RE process for a specific problem or area [29].

Requirements prioritization is a fundamental decision-making process used in requirements engineering and software development which helps to decide which user requirement should be added and delivered first for the software system. There are a variety of approaches/techniques/tools for prioritizing requirements, each with its own set of criteria. Prioritizing requirements , deciding which are the most important involves making decisions based on one or more criteria, such as budget, time constraints, technical constraints that are development cost and risk, business aspects that are market competition and regulations, customer satisfaction, or business value. [30].

Scrum was developed to promote effective performance communication at all levels, define a performance-focused culture, speed up development, and align the values of individuals and organizations. Scrum offers a flexible approach to work on various projects with a range of requirements. A scrum master, a product owner, and a scrum team are all involved in the scrum process. The scrum team is cross-functional which consists of a tester, developer and other experts in development who develop an innovative end product that satisfies the customer. Scrum teams are encouraged to come up with creative ideas since Scrum places a strong emphasis on productivity through planning and communication. Other benefits of Scrum include reduction of costs as a result of persistent communication and improves the quality of software's by informing and updating every team member on challenges and changes during development [3].

The advantage of adopting distributed development are delivering projects on time in the market, interacting closer to the potential client and also satisfying the local market demand for the companies which are international. Due to limited face-to-face communication, lack of scrum team spirit, and lack of trust between remote employees are the major challenges in distributed scrum [4].

In agile the requirements are analyzed with close collaboration with the customer so that the requirement change at any moment could be achieved. As requirements are prioritized based on the immediate business value, system architecture or requirements related to system improvement may initially be disregarded. It may be challenging to come to an agreement on the priorities of the requirements because different customers may have conflicting needs. Clients may be unable or unwilling to directly prioritize requirements in agile development [6].

2.3. Literature Review Study

Table 2.1: Literature Review

P.N	Authors Names	Title	Year	Contribution	Limitation
1.	Apoorva Srivastava; Sukriti Bhardwaj; Shipra Saraswat [3]	SCRUM model for agile methodology	2017	Proposed a framework for scrum model	Framework has not been verified from industry
2.	Kleophas M odel; Carolin Mombrey; Georg Herz wurm [5]	Paving the Way to a Software Supported Requir ements Prioritizati on in Distributed Scr um Projects	2022	Proposed a Software-Supported Requirements Prioritization in Distributed Scrum Projects	A qualitative content analysis has not been completed before the publication
3.	Leonardo S anches dos Santos; Alexandre L' Erario; Tiag o Pagotto; J oao Ricardo Moreno Camilo; Fabr icio Sousa Oliveira; Jos e Augusto Fabri [7]	A SCRUM-Based Process to Distributed Projects in Multidisciplinar y Teams: A Case Study	2018	Introduce a new role of Integration owner for the distributed teams of scrum	Collected data by only one company for the case study

4.	Ibrahim Seckin; Tolga Ovatman [9]	An Empirical Study on Scrum Application Patterns in Distributed Teams	2018	Scrum Master should be co-located with the development team to ease the communication and to ensure that Scrum is applied in the correct way.	In the study, responses were collected from different offshore centers but all of them were taken from the same company
5.	Kaaenat Ali; Junaid Ali Khan; Farwah Aizaz; Mansoor Ahmed [13]	Software Requirements Prioritization in the Context of Global Software Development	2021	Presented a technique of software requirements prioritization in the context of global software development with the help of the communicate bond belong (CBB) theory to enhance communication among the stakeholders	Conducted a controlled group experiment for the validation of this study which represents a fewer domain.
6.	Muhammad Younas; Dayang Norhayati Abang Jawawi; Muhammad Arif Shah; Ahmad Mustafa; Muhammad Awais; Muhammad Kamran Ishfaq; Karzan Wakil [18]	Elicitation of Nonfunctional Requirements in Agile Development Using Cloud Computing Environment	2020	Proposed a Non-Functional Requirements Elicitation methodology using Agile Development in Cloud Computing Framework and Natural Language Processing Based tool	Model evaluated only on a single dataset. Should be evaluated by more data sets for accuracy
7.	Aleksander Jarzębowicz*, Natalia Sitko [31]	Agile Requirements Prioritization in Practice: Results of an Industrial Survey	2020	This paper provides an update on how RP is done in agile practice on the basis of survey responses gathered from 69 IT industry practitioners.	Even when trying to reach people from various company types and application domains, it is hard to ensure that the sample represents the whole population. And even if it is representative of the IT industry, there can be differences between different countries.

8.	NOOR HAZLINI BORHAN , HAZURA ZULZALIL , SA'ADAH HASSAN , NOR HAYATI MOHD ALI [32]	REQUIREMENTS PRIORITIZATIO N IN AGILE PROJECTS: FROM EXPERTS' PERSPECTIVES	2022	The study provides an update on how RP is done in practice based on survey responses gathered from 20 IT industry practitioners including few academicians (knowledge experts) in few parts of Malaysia.	By making an anonymous survey, the researchers tried to minimize threats of guessing answers and providing false, "better-looking" answers, but the researchers cannot totally exclude such possibilities
9.	Noor Hazlini Borhan, Hazura Zulzalil, Sa'adah Hassan, Norhayati Mohd Ali [33]	Requirements Prioritization Techniques Focusing on Agile Software Development: A Systematic Literature Review	2019	The main objective of this SLR was to identify the RP techniques used in ASD and the current issues related to the RP techniques	There are still some issues, limitations, and challenges for RP in ASD that can be taken into consideration in future works.
10.	Najia Saher, Fauziah Baharom and Rohaida Romli [34]	A Review of Requirement Prioritization Techniques in Agile Software Development	2018	This paper has provided information about the current state-of-the-art techniques and practices for requirement prioritization in agile and the research gaps in existing works. The strengths, weaknesses, and comparison of well-known prioritization techniques for ASD are identified.	Complete guidelines for the selection of suitable prioritization technique(s) after the process of requirement change in ASD is still to be explored.
11	Zomitza Racheva, Andrea Herrmann, Roel J. Wieringa [35]	A Conceptual Model and Process for Client-driven Agile Requirements Prioritization	2010	Proposed a conceptual model for understanding the inter-iteration prioritization process in agile software.	The paper does not explicitly discuss the limitations of the study. However, the study was conducted using qualitative empirical data from a limited number of sources, which may limit the generalizability of the findings.

12	Noor Hazlini Borhan, Hazura Zulzalil, Sa'adah Hassan, Norhayati Mohd Ali [36]	A Hybrid Prioritization Approach by integrating non-Functional and Functional User Stories in Agile-Scrum Software Development (i-USPA): A preliminary study	2022	Outlined a research plan for creating an integrated user story prioritizing approach. Moreover, identified the gap in the literature regarding the prioritization of both functional and non-functional user stories simultaneously	the study only gathered data from a small group of experts or software practitioners who use Agile
13	Ville T. Heikkila, Casper Lassenius, Daniela Damian, Maria Paasivaara [37]	A Mapping Study on Requirements Engineering in Agile Software Development	2015	Conducted a mapping study of research literature on requirements engineering in agile software development to identify strong areas of knowledge and gaps in knowledge.	The search was limited to the Elsevier Scopus abstracts database. The study did not include a systematic quality assessment of the articles analyzed.
14	Hamzah Alaidaros, Ahmed Bakodah, Salim F. Bamsaoud [38]	A Review on Requirements Prioritization Approaches of Software Project Management	2022	Reviewed the existing requirements prioritization approaches used in software project management. Also, Evaluated and summarized a body of scientific literature from different sources using a narrative review methodology	The review is restricted to only 17 different studies. The paper does not provide empirical evidence of the effectiveness of the Whale Rank approach in comparison to other existing approaches.
15	Woogon Shim [39]	An Agile Method of Representing, Organizing, and (Re) Prioritizing Requirements in a Large Enterprise	2019	Providing practical guidance on how to manage changes in agile requirements engineering. Proposing a way to deal with these themes without compromising on the essence of agile.	The three key themes in requirements engineering cannot be handled separately, which may make it difficult to focus on just one of them. It is questionable whether the same algorithm can be used to assess mutually dependent requirements.

16	Ethan Hadar, Amin Hassanzadeh [40]	Big Data Analytics on Cyber Attack Graphs for Prioritizing Agile Security Requirements	2022	Proposing the Agile Security methodology and set of technologies for prioritizing security requirements in large-scale environments. Providing an automated end-to-end approach that does not require any changes in the network architecture and current designs of business processes prior to assessment.	The paper does not explicitly mention any limitations of the proposed Agile Security methodology and technologies. The implementation of the proposed approach may require significant resources and expertise, which may not be feasible for all organizations
17	Sanjaya Kumar Saxena, Rachna Chakraborty [41]	Decisively: Application of Quantitative Analysis and Decision Science in Agile Requirements Engineering	2019	Introduced the Decisively tool, which brings a new perspective to automation in the RE process through the application of QUADS techniques.	The Decisively tool in addressing Agile RE challenges may vary depending on the specific context and requirements of each project.
18	Zornitza Racheva, Maya Daneva, Klaas Sikkel, Roel Wieringa, Andrea Herrmann [42]	The Decisively tool presented in the paper applies Quantitative Analysis and Decision Science (QUADS) techniques to prioritize requirements in Agile RE. The tool uses an Analytical Hierarchical Process (AHP) for prioritization and estimation, which can be useful in distributed scrum where multiple stakeholders are involved.	2010	Presenting an empirical investigation of the continuous prioritization of requirements during agile software development projects. Providing insights into the assumptions of agile requirement prioritization approaches and their applicability in different agile project contexts.	With any empirical study, there may be limitations in terms of the generalizability of the findings due to the specific context and sample size of the case study.

19	Heera Sheemar, Gurpreet Kour [43]	Enhancing User-Stories Prioritization Process in Agile Environment	2010	The contribution of this paper is that it highlights the importance of the user-stories prioritization process in Agile software development life cycle. The paper emphasizes that the user-stories prioritization process starts with the planning phase and remains throughout the project duration.	The proposed method for prioritizing user stories may not be applicable to all software development projects. The effectiveness of the proposed method may depend on the specific requirements and constraints of each project.
20	Najia Saher, Fauziah Baharom, Rohaida Romli [44]	Guideline for the Selection of Requirement Prioritization Techniques in Agile Software Development: An Empirical Research	2020	Highlighting the importance of requirement prioritization in Agile software development. Identifying the strengths and weaknesses of existing requirement prioritization techniques commonly used in Agile.	The empirical study was conducted in Pakistan and the results may not be generalizable to other countries or regions.
21	Nikhil Govil, Ashish Sharma [45]	Information Extraction on Requirement Prioritization Approaches in Agile Software Development Processes	2010	The paper provides information on requirement prioritization approaches in Agile Software Development processes. It extracts information on discrete requirement prioritization approaches along with their strengths and weaknesses.	The information provided in the paper is based on the authors' research and analysis, and not necessarily be applicable to all software development projects.

22	Joseph Gillain, Ivan Jureta, Stephane Faulkner [46]	Planning Optimal Agile Releases via Requirements Optimization	2016	Formulated the agile release decision problem as an optimization problem. Provided a modeling language to represent instances of this problem as requirements models.	The proposed approach may not be applicable to all types of agile projects and may require customization based on the specific needs of the project.
23	Rashmi Popli, Naresh Chauhan, Hemant Sharma [47]	Prioritising User Stories In Agile Environment	2014	Proposed factors that can calculate the importance of user stories and effort per user-story in an agile dynamic environment. Described the Agile requirement spectrum for prioritization.	The proposed prioritization technique is not be applicable to all agile environments and may require customization based on the specific needs of the project.
24	Md Shamsur Rahim, AZM Ehtesham Chowdhury, Shovra Das [48]	RIZE: A Proposed Requirements Prioritization Technique for Agile Development	2017	Proposed a new technique for requirements prioritization in agile development. Introduced RIZE, a new requirements prioritization technique that is simple to understand, takes less time to determine priority, is flexible to customization, and can deal with the issue of starvation.	It recommends performing several empirical studies on requirement prioritization using the proposed technique and reporting the effectiveness of this proposed approach further as future work.

25	Amjad Hudaib, Fatima Alhaj [49]	Self-Organizing Maps for Agile Requirements Prioritization	2019	Provided an approach for requirement prioritization (RP) within the agile development model. Highlighted the importance of deciding the order of requirements in the project timeline to ensure the overall success of a software product.	The effectiveness of the approach may depend on the quality and completeness of the input data.
26	Rietz, T., Schneider [60]	We See We Disagree: Insights from Designing a Cooperative Requirements Prioritization System	2020	Provided a design science research methodology to propose design principles for a cooperative requirements prioritization system using the MuSCoW method	Conducted a controlled group experiment for the validation of this study which represents a fewer domain.

As in the above table papers related to requirement prioritization related to scrum and distributed scrum are discussed. Since the distributed scrum environments are incompatible with the characteristics of agile software development, implementing agile methodologies in distributed software development projects creates problems for research and practice. When dealing with distributed stakeholders, High levels of coordination and communication are needed for Scrum projects, which can no longer be completed synchronously and individually without extra effort. [5].

The development of software products is severely disturbed by a number of communication and information-sharing issues. The development process is constantly interrupted by direct communication. There is an excessive number of informal communication. It is impossible to spread knowledge because of the information centralization among certain members [7].

One of the prominent issue analyzed was focused on the feedback that must be given more frequently in distributed scrum teams. Along with many other findings, the significance of site visits and regular feedback is highlighted in distributed scrum; however, site visits do not always improve communication. Another significant finding is that the Scrum Master shares space with the development team to facilitate communication and guarantee proper Scrum implementation [9].

The requirement prioritization is little different when it’s practiced in global software development. The most critical issue is communication which happens due to the globally dispersed environment (including time differences, language barriers, and culture differences). This problem has been addressed by using the communicate bond belong (CBB) theory to reduce the problems aroused by global disperse [13].

Agile methodologies and practices are not meant to be applied directly to distributed agile development; instead, they are intended for and most suitable for co-located software development. However, it has been observed that distributed teams can be just as productive as small co-located teams if Scrum can be implemented effectively, and efficiently [8].

The paper provides insights into the challenges faced in agile requirements prioritization and presents a conceptual model that can be used to structure future empirical investigations about this topic [35].

Paper [36] provides insights into Agile-Scrum Software development and how non-functional and functional requirements are prioritized. The paper proposes a hybrid approach to prioritization that combines non-functional and functional user stories, which can be useful in the distributed scrum where teams work remotely.

The problematic area of prioritization of requirements was identified in the study as it is one of the key aspect challenges in agile requirements engineering software development. The study also proposed solutions to problems related to requirement engineering in agile software development [37].

One crucial software project management task is requirements prioritizing, which is used to decide which features or requirements should be included in a certain release or implemented first.

However, the approaches currently in use for prioritizing requirements do not account for all the factors that need to be taken into consideration [38].

Paper [39] proposes a framework or approach which could handle requirement reprioritization in agile at the end of each sprint regularly. According to the author, priority evaluation is an important task that should be done at least after every iteration.

Agile security techniques and technologies for identifying, modelling, and continuously prioritizing security requirements were presented in the paper in an agile. The term "agility" describes the concentration and speed required to reorganize the backlog of necessary improvements in a company's safety record in order to be in line with the constantly shifting demands of an enterprise environment [40].

The decisively tool presented in the paper applies Quantitative Analysis and Decision Science (QUADS) techniques to prioritize requirements in Agile RE. The tool uses an Analytical Hierarchical Process (AHP) for prioritization and estimation, which can be useful in the distributed scrum where multiple stakeholders are involved [41].

The research findings indicate an important difference between the actual experiences of practitioners and the presumptions presented in the literature on agile requirements engineering. Three main theoretical approaches which are not being followed in scrum practice are: (i) the client's essential role in the value creation process; (ii) business value's dominant status as a primary criterion for prioritization; and (iii) the prioritization process's contribution to project goal achievement. [42].

Paper [43] provides insights into the importance of the user-stories prioritization process in the agile software development life cycle and proposes a method for prioritizing user stories based on various factors. As distributed scrum is a type of agile methodology, the proposed method for user-stories prioritization can be applied in distributed scrum as well.

Najia Saher et al. present guidelines for choosing appropriate methods of requirement prioritization in agile software development. The purpose of this study's findings is to increase software practitioners' skill in selecting appropriate requirement prioritization strategies based on criteria for assessment during the agile inter-iteration process [44].

This study presents a comparative examination of different approaches to requirement prioritizing in agile methodology. Additionally, they discussed the comparison based on discrete criteria, which directly affect any software project's success [45].

Joseph Gillain et al. proposed a technique that can help agile projects optimize their release planning. One of the initial suggestions was to utilize goal frameworks to model the agile requirement problem. An additional contribution involved the mapping of a Mixed-Intger Program to the objective model [46].

Rashmi Popli et al. proposed criteria related to efficiency and importance that influence the order in which user stories are prioritized within a project. The importance-to-effort ratio is computed to determine which user stories should come first. The established method is very straightforward, easy to comprehend, and useful for prioritization in an agile [47].

Md Shamsur Rahim et al proposed the technique RIZE a new requirements prioritizing technique that can address the problem that is easy to comprehend, quick to establish priority, and responsive to change. As RIZE is a highly customizable, scalable, and time-saving technique [48].

Amjad Hudaib et al proposed approach which helps any software development project that follows the agile methodology, including distributed scrum. The approach can help prioritize requirements based on their importance and relevance to the project goals, which can be especially important in a distributed scrum environment where communication and coordination can be challenging [49].

In agile the requirements are analyzed with close collaboration with the customer so that the requirement change at any time could be achieved. As most requirements are prioritized based on immediate business value, system architecture or requirements related to system improvement may initially be disregarded. It may be challenging to come to an agreement on the priorities of the requirements because different customers may have different requirements needs. Clients may be unable or don’t want to directly prioritize requirements in agile development [6].

Another major topic in RP is how to deal with requirement interdependencies. The majority of current RP techniques consider that all requirements are independent, and treat concerns about interdependencies as future work. The need for requirement dependencies during the RP process was not recognized until recently. Multi-aspect-based RP, multi-decision-maker RP via multi-objective optimization, SNIPR, value-based RP, mathematical programming, RP under no additive-value condition, social network analysis for RP, and interactive RP have all attempted to meet this demand. Despite the presence of existing RP approaches, the data demonstrated that limits still persist. Complexity, scalability, a lack of automation and intelligent terminology, dependencies, a lack of quantification and prioritization for the involved stakeholders, and the requirement for significant professional engagement are some of the constraints [10].

All of the available techniques for requirement prioritization come with certain pros and cons. However, choosing the right technique is the most critical part of this process. All of these techniques (AHP, NAT, CV, Ranking, Top-Ten, MoSCoW, Priority Groups, CBRanking) aim to save time, cost and manpower. However, still, a wrong technique will take us to resource wastage [11].

All of the requirements are naturally not prioritized. All stakeholders contribute to the most important part of software engineering which is requirement prioritization. All techniques have different usage and ways to prioritize. This paper has a detailed analysis of all those techniques to help stakeholders choose the most effective based on their needs [12].

The number of Software’s being developed is increasing day by day. With each passing day, the challenges are increasing due to rapid requirements, changing environment and global effectiveness. A more in-depth analysis is needed to pinpoint the difficulty being faced in requirement prioritization and also to eliminate the risk and cop up with the time constraints [14].

The software development industry has been evolving from day one. However, in recent years, it has been seen that the software industry is showing a soft corner for agile software development. This study considers all available literature and surveys conducted in the industry to identify the most in-use software development technique. It has been noticed that among all techniques (Scrum, Extreme Programming, Crystal, FDD, DSD) scrum is the most popular one. Researchers and developers are practicing and working on Scrum a lot [15].

As agile software development techniques like Scrum allow teams to focus on producing products and enhancing communication, it is the simplest and best software development methodology available. On the other hand, since these agile methods were created for in-house software development, they cannot be used in accordance with distributed agile development. Face-to-face communication is lacking in distributed projects, but there is no other option but to replace it with "rich" communication channels and "simulations" of high-speed, high-quality face-to-face interactions. A distributed team needs to use several communication tools [16].

Scrum's core components are built to take advantage of physical colocation and extensive face-to-face interactions, so moving Scrum to a distributed environment is seen as an important hurdle to its adoption. In particular, it can be seen how the absence of co-location has a significant impact on the roles (for example, understanding and implementation, ceremonies (for example, attending meetings), necessitating the design of additional elements. It is suggests using individual, role- or team-specific backlogs to get around bottlenecks related to artifacts, using a shared backlog, even going so far as to encourage users to report bugs through the product backlog [17].

While companies are trying to make software development successful, with every-evolving agile and scrum methodology. One of the reason that causes software to fail is nonfunctional requirements. NFRs are usually overlooked. Therefore, this paper addresses this problem with the help of NLP (natural language processing). The proposed methodology helps software development to be more successful by addressing NFRs [18].

The requirements engineering process consists of seven steps out of which the most complicated is elicitation. The method of elicitation changes based on the stakeholder's geography and project nature. The purpose of this study is to help Project Managers and developers identify the best elicitation technique by comparison [19].

The traditional and Agile Requirement Engineering approaches are usually used to gather requirements. The studies have shown that agile has proven it to be more transplant, interactive and likeable for stakeholders. Also, Agile has been seen to improve overall efficiency and save resources [20].

Agile development is a widely used process in requirement engineering. This paper addresses flaws in the requirement engineering process and improves them by proposing some changes. It also compares the new methodology with the old one on a larger scale [21].

In modern agile methods like Scrum, the important job of prioritizing requirements is given to roles like the product owner. However, this can slow things down and cause misunderstandings and conflicts among stakeholders. Allowing stakeholders to work together on prioritizing requirements can reduce the load on product owners and get stakeholders more involved in this key step, making them feel more connected to the final product. So, it's important for stakeholders to have a shared understanding of the requirements [60].

2.4. Limitations in Existing Literature

2.4.1. Face to Face Communication

One of the most significant limitations encountered in collaborative work, particularly in project management and development, is the absence of face-to-face communication [4] [16]. This lack of direct interaction presents substantial challenges in the processes of requirement prioritization and requirement gathering. In-person communication allows stakeholders to engage in real-time discussions, ask clarifying questions, and provide immediate feedback, all of which are critical for developing a shared understanding of project requirements [5].

2.4.2. Temporal Difference

Temporal distance is frequently identified as a significant limitation in existing literature on collaborative work and project management. The challenge of coordinating activities and communications across different time zones introduces a range of complications. Each individual operating in a distinct time zone encounters unique limitations that can impede the efficiency and effectiveness of the collaborative process. The asynchronous nature of communication necessitated by temporal distance often leads to delays in decision-making and feedback, as responses are dependent on the availability of team members who may be working hours apart [4] [16] [17] .

2.4.3. Communication, Coordination and Control

Communication, coordination, and control are widely recognized as significant limitations in the context of distributed Scrum. The inherent challenges of managing these aspects become even more pronounced in distributed teams where members are geographically dispersed [8]. Effective communication is often hindered by the lack of face-to-face interactions, leading to potential misunderstandings and misinterpretations of information. Coordination of tasks becomes a complex goal as team members operate in different locations and possibly different time zones, complicating the synchronization of work efforts and the seamless integration of contributions from various team members [4] [16] .

2.4.4. Information Dissemination

Information dissemination is also a significant limitation, particularly in the context of distributed teams and complex project environments. The challenge of effectively distributing information across a dispersed team can lead to numerous issues that hinder overall productivity and coherence [7]. In traditional, co-located settings, information can be easily shared through informal conversations, spontaneous meetings, and direct observations, ensuring that everyone remains informed and aligned. However, in distributed teams, the dissemination of information relies heavily on digital communication tools, which can often lead to delays, information silos, and miscommunication [4] [5].

2.5. Summary

This chapter provides an overview of the existing literature that was related to the research work. The main purpose of LR was, to identify the challenges related to requirement prioritization in distributed Scrum while practicing it in the industry. It was important to understand that if requirements are not properly prioritized in distributed Scrum it leads to the software production failure. A comprehensive LR was done to find out the challenges related to distributed Scrum for prioritizing requirements.

CHAPTER 3

RESEARCH METHODOLOGY

3.1. Introduction

This chapter explains the research methods and the different techniques that are used to carry out the research. Research methodologies come in a variety of types, and each has its unique way of approaching things. Deciding which method to use depends on several factors, like the specific research question trying to be answered, what to discover, the tools available, and the particular field of study exploring. In this study, a mixed-method approach is used to get the best result.

The qualitative method is all about exploring the thoughts and feelings of people. It involves techniques such as conducting interviews, observing situations, and analyzing written or visual materials to gain an in-depth understanding of what's happening. When attempting to explain how and why something occurs, qualitative research is especially helpful. On the other side, there is quantitative research. This approach involves counting, measuring, and using numbers to identify patterns and connections. It's highly effective when trying to determine if one thing causes another. Quantitative research relies on surveys and other numerical data to provide clear and concise answers to research questions. Lastly, the mixed-method approach combines the best of both qualitative and quantitative. By blending techniques like talking to people, observing events, and crunching numbers, researchers get a more comprehensive view of the subject. This approach is incredibly valuable when the research question demands a well-rounded understanding of the situation.

This section of the thesis elaborates the challenges related to requirement prioritization while practicing distributed Scrum that are found through the literature review. Then the challenges found through the literature review are cross-validated through survey. Based on the challenges found guidelines are proposed to mitigate these challenges.

3.2. Research Questions

The table represents research question along with their methodology and outcome. These two research questions address challenges related to requirement prioritization in distributed Scrum.

Table 3.1: Research Questions and their Corresponding Outcomes

NO	Research Questions	Methodology	Outcome
RQ1	What are the challenges regarding requirement prioritization in distributed Scrum that could lead to software failure?	LR and Survey Questionnaire	A list of identified challenges for requirement prioritization in distributed Scrum.
RQ2	What could be the possible guideline for solving the requirement prioritization challenges in distributed Scrum?	Interviews and Focus group	Mitigation strategies will be proposed through survey and interviews.

3.3. Methodological Framework

In this research, a thorough approach is used to gather and analyze data. Information is looked at in two ways: by reading and summarizing what other researchers have written which is called a Literature Review, and by asking people structured questions through surveys. This research consists of detailed conversations with some people to understand their viewpoints better. The study focuses on three main questions: finding challenges, sorting these challenges into categories, and suggesting ways to tackle them. The diagram below tells how research is conducted:

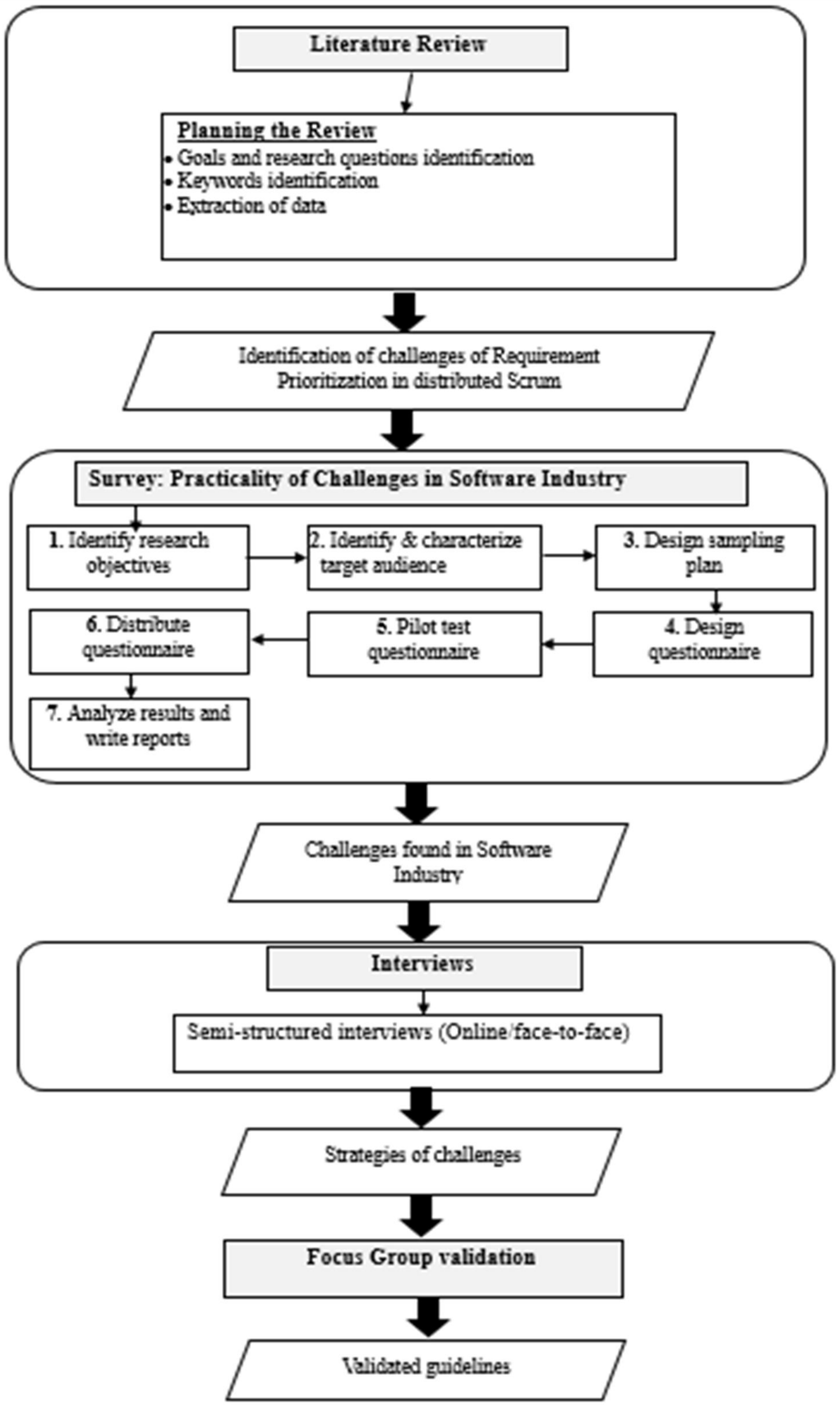
Literature Review
Planning the Review
• Goals and research questions identification
• Keywords identification
• Extraction of data
Identification of challenges of Requirement Prioritization in distributed Scrum
Survey: Practicality of Challenges in Software Industry
1. Identify research objectives
2. Identify & characterize target audience
3. Design sampling plan
4. Design questionnaire
5. Pilot test questionnaire
6. Distribute questionnaire
7. Analyze results and write reports
Challenges found in Software Industry
Interviews
Semi-structured interviews (Online/face-to-face)
Strategies of challenges
Focus Group validation
Validated guidelines

Figure 3.1: Research Methodology for the analysis of requirement prioritization in distributed Scrum for reducing software failure

3.3.1. Literature Review (LR)

A Literature Review (LR) is a rigorous and structured method of analyzing a broad range of existing research literature on a specific topic or research question. It aims to provide a comprehensive and unbiased synthesis of current knowledge in a particular field or subject area. The reason for choosing the LR is that it helped in a comprehensive examination of existing knowledge, minimization of bias, and finding research gaps. In the given topic, it's also best suited because most of the identification needs to be unbiased and should come from comprehensive extraction of existing knowledge.

3.3.1.1. Databases and Sources

A set of reputable academic databases and sources were selected to conduct the literature search. These included, but were not limited to, IEEE, ACM Digital Library, Google Scholar, Springer, Wiley, Elsevier and specific subject-based repositories. These databases were chosen for their extensive coverage of research articles across various disciplines.

3.3.1.2. Keywords and Search Terms

The search terms and keywords were carefully chosen to reflect the various aspects of the research topic. A combination of specific and broad terms related to the research question, concepts, and subtopics were utilized. Boolean operators (AND, OR, NOT) and other advanced search techniques were employed to construct effective search queries.

Example:

Keywords: "Distributed Scrum," "Requirement Prioritization," "Challenges of requirement prioritization in distributed Scrum," "Challenges of Requirement Prioritization in Agile"

Search Query: ("Distributed Scrum" OR "Requirement Prioritization") AND "Agile"

3.3.2. Quantitative Research

Quantitative research is all about measuring things and using math to understand data. The main goal is to find patterns or trends in a big group of things, and the results can apply to other similar groups. In quantitative research, it should be clear about how data is collected to make sure it's reliable and accurate. Quantitative research has its strengths. One big advantage is that it can often apply to lots of different situations because it uses a structured approach and studies many cases. It also uses math to show how things are related. And when someone else does the same study, they should get similar results. However, there are limits to quantitative research. It doesn't go as deep as qualitative research, which looks at details and the context of things. Sometimes, being too structured means missing out on important details or differences. Also, because it's all about numbers, it might not fully capture the complexity of human behavior and experiences. By adding more words to the simplified version, we've expanded on the explanation of quantitative research and its strengths and limitations [50].

3.3.2.1. Type of Quantitative Research: Survey

Surveys:

A survey is a type of research methodology that consists of a series of questions arranged in a specific order to collect data directly from participants. Being able to gather information about a particular phenomenon by creating questions that represent the beliefs, attitudes, and actions of a group of people makes it one of the most popular quantitative techniques. There are many advantages to surveys. The method's low cost in comparison to other options and its high representativeness of the entire population are two of its most significant advantages. Conversely, the validity of survey results relies strongly on the design of the questionnaire and the accuracy of responses given by participants. Structured questionnaires were given to participants, often with closed-ended questions [53].

The survey served as an important tool to systematically collect data from a targeted group of individuals. Its objective was to gain a deeper understanding of their opinions, experiences, and perceptions related to the research topic. This data was essential for achieving a comprehensive view and adding empirical evidence to the study.

Surveys are one of the major ways for the study as not much of the data regarding this was available. The survey was conducted with well-renowned professionals to see how they are currently catering to this situation. It was evident that RP in distributed scrum is quite a challenge.

The survey followed the guidelines presented by M. Kasunic in his book "Designing an Effective Survey" [54].

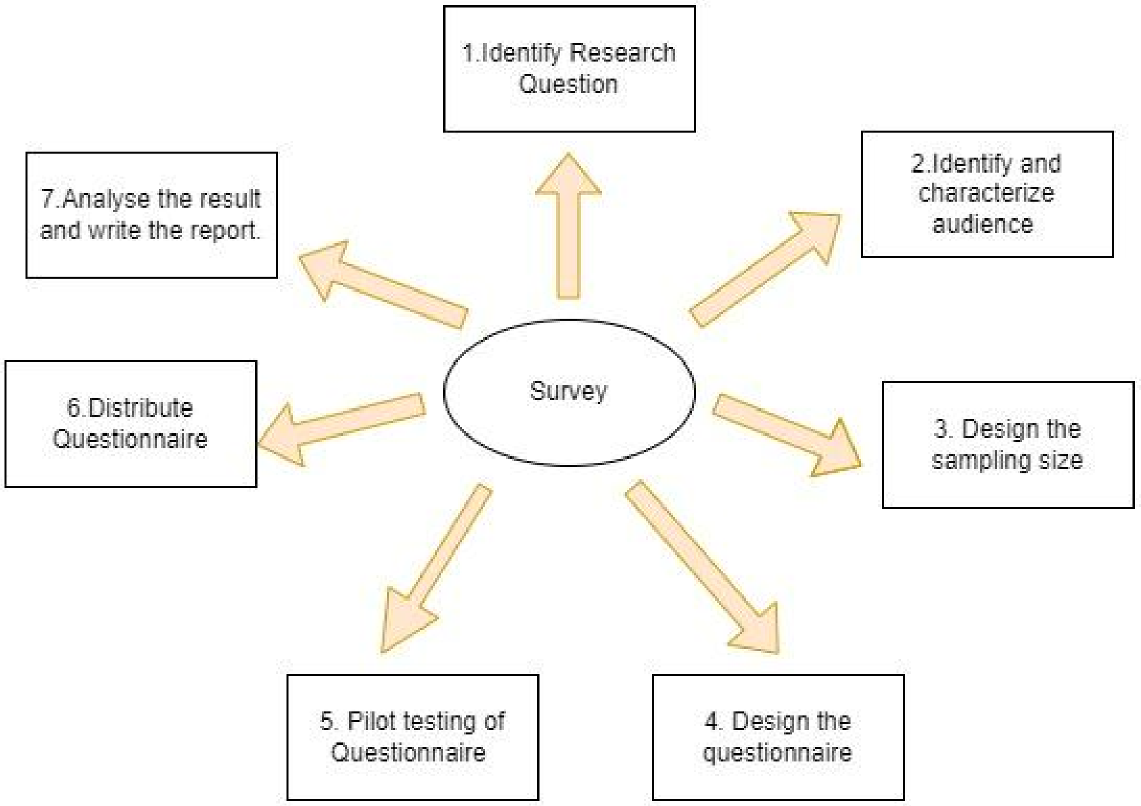
1.Identify Research Question
2.Identify and characterize audience
3. Design the sampling size
4. Design the questionnaire
5. Pilot testing of Questionnaire
6.Distribute Questionnaire
7.Analyse the result and write the report.
Survey

Figure 3.2: Survey Steps [54]

3.3.2.1.1. Identification of Survey Objective

The first and foundational step in our research is the identification of clear and well-defined research objectives. This initial phase serves as the base scope that guides our survey, ensuring that it remains closely aligned with the aims of our study. This phase begins with a comprehensive problem formulation. The focus was on the complex environment of requirement prioritization within the framework of distributed Scrum teams, as these distributed agile setups present unique challenges and complexities. The goal is to pinpoint the specific issues, gaps, or inefficiencies within existing requirement prioritization practices, particularly when distributed Scrum teams are involved.

To bring clarity to this research, a set of well-crafted research questions was established. These questions serve as the foundation upon which our survey is built, guiding the structure and direction of our data collection and subsequent analysis in addition to research questions, broader objectives were defined and the specific scope of survey.

3.3.2.1.2. Identification and Characterization of the Targeted Respondents

In the pursuit of our research objective, the identification and characterization of the targeted respondents emerge as a crucial step. These respondents serve as the primary sources of insights, holding valuable knowledge relevant to requirement prioritization in distributed Scrum. Our focus is on agile practitioners, Scrum Masters, product owners, and team members engaged in requirement prioritization within distributed Scrum. By identifying these important stakeholders, the foundation for gathering in-person knowledge was created.

Furthermore, we look into the distinctive characteristics of our respondents. This includes factors such as geographical locations, organizational roles, experience levels, and industry sectors. This characterization allows us to gain a holistic perspective, ensuring a comprehensive view of the challenges and strategies involved in requirement prioritization within distributed Scrum teams. The insights gathered from this broader group of respondents will enrich our research, contributing to a deeper understanding of agile methodologies in distributed Scrum.

3.3.2.1.3. Designing of Sampling Plan

In the survey methodology, the third pivotal step involves the careful design of a sampling plan. This step is instrumental in how we select and engage with our respondents, ensuring a representative view of prioritizing requirements in distributed agile environments. To achieve this, a simple random sampling approach was used, categorizing our target respondents into subgroups based on factors like geographical location, roles, and experience levels. Each subgroup will be sampled randomly, ensuring a diverse representation of perspectives within distributed Scrum teams. This systematic approach will enhance the comprehensiveness of our data.

The population size has an impact on the sample size calculation, especially in cases where the population is small. The sample size is calculated using a variety of formulas based on the size of the population. For these kinds of computations, the sample size is determined using Solvin's formula.

Precision:

Precision refers to how closely the characteristics of our estimated population match the actual population. The level of precision depends on the acceptable level of risk in decision-making. Increasing precision means reducing the margin of error, which often requires a larger sample size.

Confidence Interval:

A confidence interval is a range of values within which we are confident the true population parameter lies. It's like setting boundaries on where the actual values are likely to be. Standard deviation helps calculate this interval for samples or populations.

Confidence Level:

Confidence level reflects how confident we are in our chosen sample representing the entire population. For example, if we say 95 out of 100 samples are risk-free, it means we're 95% confident in our sample selection. Confidence level is determined using confidence intervals and statistical tools like the Standard Normal distribution and Central Limit Theorem.

Population Size:

The size of the population also influences sample size calculations, especially when the population is small. Different formulas are used to calculate sample size based on whether the population is large or small, ensuring the sample is representative of the entire population without being too burdensome to collect.

3.3.2.1.4. Design and Write the Survey Questionnaire

In the subsequent crucial phase of our survey methodology, the focus was on the formulation and development of the survey questionnaire. This step is central to our data collection efforts, serving as the primary means for gathering insights relevant to the research's core themes related to requirement prioritization within distributed Scrum teams.

The questionnaire's design was marked by precision and careful consideration. The questions have been carefully designed to perfectly match our study goals making sure they gather the particular challenges in requirement prioritization within the framework of distributed Scrum teams. This comprehensive approach involves a closed-ended questions. Closed-ended questions offer structured responses, enabling quantitative analysis. Every aspect of the questionnaire's design is approached with an emphasis on question clarity, neutrality, and relevance, reflecting our commitment to extracting meaningful data that will significantly enhance our understanding of requirement prioritization practices in distributed Scrum. Surveys frequently use the Likert scale to determine respondents' opinions about particular subjects. For a particular question typically five to seven opinions are given like: Strongly Agree, Agree, Neutral, Disagree and Strongly Disagree in a close-ended questions. The respondent is bound to tick any one of the option in Likert scale. The survey questionnaire for the responses from industrial practitioners is mentioned in the appendices section as depicted in Appendix A.

3.3.2.1.5. Pilot Test Questionnaire

The pilot study is an early study conducted to assess the feasibility of a more in-depth study that could be conducted in the future. The pilot study investigates the feasibility of the full-scale investigation and attempts to identify potential concerns or constraints that could arise in a real study. The pilot study result is used to make revisions before proceeding to a full-scale study.

In this pilot study, the questionnaire was sent to 15 members out of which 12 filled out the form. The targeted audience is working in the distributed scrum. The study was conducted to future validate the questionnaire in terms of the validity of questions, and whether the problems identified are valid and understandable.

Some respondents pointed out challenges related to understanding the terminologies and lack of purpose. A few of them mentioned the grammatical mistakes and the questions about whether they were valid or not. One respondent felt repetition in questions. Another gave me a suggestion to add a few lines of description about the questionnaire so the respondent could understand the aim of the research. So, before pilot testing 23 questions after the pilot testing evaluation 23 questions were left and 3 questions were revised.

Once the questionnaire was updated as per the feedback, we received 12 responses. The result of the questionnaire was diverse and no amendment requests were received. Therefore, now the questionnaire is pretty much in good shape.

3.3.2.1.6. Distribute the Questionnaire

In the subsequent phase of our research methodology, the research moves forward with the distribution of the survey questionnaire. This step is important in gathering insights and data related to requirement prioritization within distributed Scrum environments, marking a crucial milestone in our data collection process.

The distribution phase involves reaching out to our identified target respondents. Given the unique context of distributed Scrum, our approach includes an online survey. For those respondents geographically dispersed, online survey platforms like LinkedIn and electronic communication to ensure ease of access and participation were used. Through this thoughtful distribution strategy, the aim was to engage with a diverse group of stakeholders, capturing a wide spectrum of experiences and insights that can shed light on the challenges and strategies related to requirement prioritization in distributed Scrum teams.

3.3.2.1.7. Analyze Results

Through the analysis of results gained by the questionnaire, the aim was to explain the complexities and best practices associated with requirement prioritization in distributed Scrum, ultimately providing valuable insights and recommendations. This phase is instrumental in the progression of our research, shaping the findings that will contribute to the broader knowledge of requirement prioritization in distributed Scrum.

3.3.2.2. Sampling

Gathering data from an entire population isn't feasible in research to address research questions. Therefore, there must be a sample size that accurately represents the whole population. There could be distinct stages that could be followed during sampling, including:

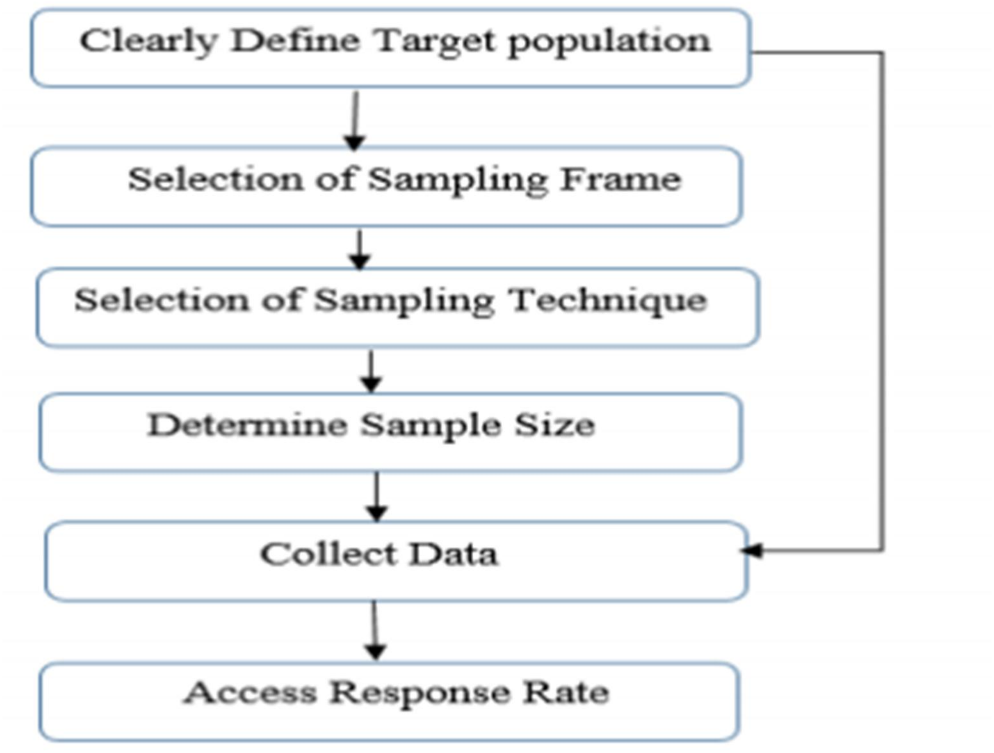
Clearly Define Target population
Selection of Sampling Frame
Selection of Sampling Technique
Determine Sample Size
Collect Data
Access Response Rate

Figure 3.3: Steps of Sampling Process [59]

3.3.2.3. Clearly Define Target Population

In this stage population is clearly defined which is to be target. Population is defined as the total number of people related to that specific field or area. In this research study, the targeted respondents were product owners, scrum masters and developers working in distributed Scrum.

3.3.2.4. Selection of Sampling Frame

The sample usually shows what the whole population is like. A sampling frame is a list of actual cases used to pick the sample. In this research study, the sampling frame were product owners, scrum masters and developer working in distributed Scrum.

3.3.2.5. Selection of Sampling Technique

Sampling means picking out a smaller group from a larger one, like from a list or the whole population, to make conclusions about the bigger group or relate to existing theories. There are mainly two types of sampling: probability sampling and non-probability sampling, each with its own methods**.**

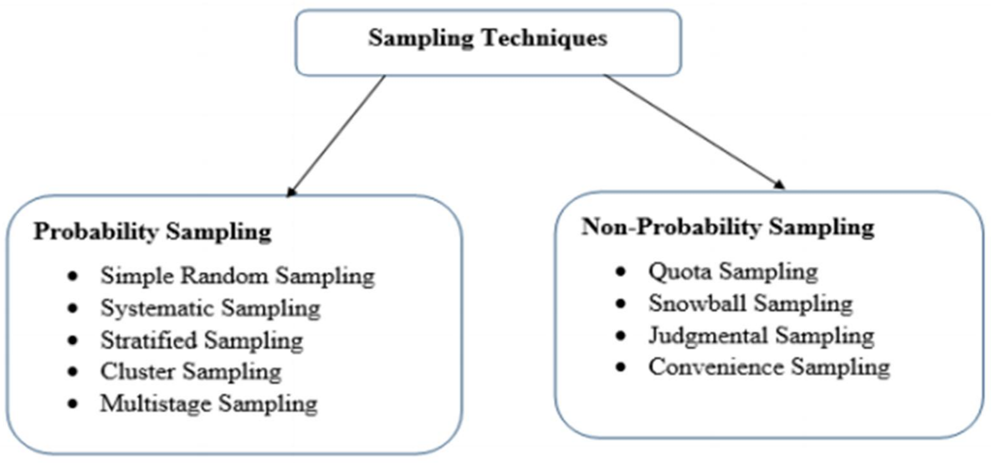

Figure 3.4: Sampling technique [59]

3.3.2.5.1. Probability Sampling:

Probability sampling ensures that every item in the population has an equal probability of being selected for the sample. One approach to conducting random sampling involves first creating a sampling frame, and then using a computer program to generate random numbers to select the sample from this frame. While probability sampling offers the highest level of freedom from bias, it can also be the most time-consuming and labor-intensive method in terms of the resources required to achieve a desired level of sampling error.

i. Simple Random Sampling:

A simple random sample ensures that every individual in the population has an equal chance of being included. However, there are several drawbacks to this method:

It requires a comprehensive list of all units within the population.

In certain studies, such as those involving face-to-face interviews, the costs can be substantial if the sample units are dispersed over a large geographic area.

The standard errors of the estimates can be relatively high.

ii. Systematic sampling:

Systematic sampling involves selecting every nth individual after a randomly chosen starting point. For example, in a survey, any criteria for nth number might be selected like every fifth consumer from the sample. This method is advantageous because of its straightforwardness.

iii. Stratified Sampling:

Stratified sampling involves dividing the population into distinct subgroups, or strata, and then taking a random sample from each subgroup. These subgroups are natural categories, such as company size, gender, or occupation. This method is particularly useful when there is significant variation within the population, as it ensures that each subgroup is properly represented.

iv. Cluster sampling:

Cluster sampling involves splitting the entire population into distinct clusters or groups. Then, a random sample of these clusters is selected, and every member of the chosen clusters is included in the final sample. This method is particularly advantageous for researchers dealing with subjects spread across large geographical areas, as it helps save time and money.

v. Multistage Sampling:

Multi-stage sampling involves gradually narrowing down a broad sample through several steps. For example, if a certain country automobile magazine publisher wanted to conduct a survey, they could randomly sample automobile owners across that country. However, this approach would be costly and time-consuming. A more economical method is multi-stage sampling. This process begins by dividing a certain into several geographical regions. Then, a random selection of these regions is made, followed by further divisions, such as local authority areas. Random selections are made again from these subdivisions, and the process continues, perhaps down to towns or cities. The main goal of multi-stage sampling is to concentrate the sample within a few geographical areas, thus saving both time and money.

3.3.2.5.2. Non Probability Sampling:

Non-probability sampling is commonly linked with case study research design and qualitative research. In these contexts, case studies usually involve small samples aimed at exploring real-life phenomena rather than making statistical generalizations about the larger population. The sample of participants or cases doesn't need to be representative or randomly selected. However, it's important to have a clear explanation for why certain cases or individuals are included over others.

i. Quota sampling:

Quota sampling is a non-random sampling method where participants are selected based on specific predetermined characteristics. This ensures that the overall sample reflects the same distribution of characteristics as found in the broader population.

ii. Snowball sampling:

Snowball sampling is a non-random sampling technique that relies on initial participants to recruit additional participants, thus expanding the sample size. This method is particularly useful for reaching small, hard-to-access populations, such as secret societies or exclusive professions.

iii. Convenience sampling:

Convenience sampling involves selecting participants simply because they are easily accessible. This method is often preferred by students due to its low cost and simplicity compared to other sampling techniques. Convenience sampling can help overcome various research limitations, such as using friends or family members as participants, which is easier than targeting unknown individuals.

iv. Purposive or judgmental sampling

Purposive or judgmental sampling is a deliberate strategy where specific settings, individuals, or events are chosen to gather crucial information that cannot be acquired through other means. In this approach, the researcher selects cases or participants for inclusion based on their belief that these selections are important or relevant to the study**.**

In this research simple random sampling technique is used which is the type of probability sampling. Major advantages of simple random are its simplicity for gaining the sample size and the chances of biasness are less.

3.3.2.6. Determination of Sample Size

To make sure our sample represents the whole population and to avoid mistakes and bias, it's crucial to have a big enough sample size. Having more people in our sample reduces the chances of getting things wrong. There are lots of formulas to figure out how big our sample should be. However, the specific formula use doesn't really change how well our sample represents the population.

Solvin's formula was used to calculate the sample size. The sample size for this research was for the survey was 259. Responses were collected from different respondents who work in the distributed Scrum framework.

Solvin's formula: $n=N/1+Ne^2$ where n is sample size, N is population size and e is margin of error.

Where population size is 1200, margin of error is 5.5%

Computing the values in formula:

$n=N/1+Ne^2$

$n =1200/1+1200(0.005)^2$

$n =1200/1+1200(0.003025)$

$n =1200/4.63$

$n =259$

3.3.2.7. Data Collection

Once the targeted population, sampling frame, sampling techniques, and sample size are determined, data collection takes place. Data was collected by distributing questionnaires in different IT industries through LinkedIn emails.

3.3.2.8. Assess Response Rate

The number of cases that have consented to participate in the research and the rate of response. These examples were chosen from an actual sample. A 100% response rate is extremely uncommon for researchers to get for a number of reasons, including inability, unwillingness, and ineligibility. To reply, respondents may be present, but researchers are unable to get in touch with them. Furthermore, the response rate is crucial because each nonresponse contributes to the bias in the final sample. Therefore, employing big samples, appropriately defining the sample, and using the right sampling procedure can all assist in lessening the bias in the sample [59].

3.4. Qualitative Research

Qualitative research is all about exploring and understanding the details of how people think, act, and see the world. It goes deep into what things mean to people and how they act in different situations. Instead of just numbers, it cares about the stories and descriptions people share during interviews or observations. In this kind of research, there are two ways of thinking: interpretive and critical. The interpretive way is about seeing things from the perspective of the people being studied. It knows that everyone's experiences are a bit different. The critical way goes even further. It doesn't just understand; it also tries to question and change the way things are in society. Qualitative research has some strong points. It really digs deep into what people experience, and that can uncover important stuff that you might miss with numbers alone. Plus, it's flexible, so researchers can adjust things as they learn more. This keeps the research on track and useful. But, like all methods, there are challenges. Because it's about people's thoughts and feelings, there can be biases from both the people being studied and the researchers themselves. Going deep can also mean you don't have answers for everyone, and not all the findings apply everywhere. And because it's about understanding, different researchers might see different things in the same information [57].

3.4.1. Interviews

Engaging in one-on-one interactions where the researcher poses open-ended questions to participants is a dynamic approach to collect qualitative data. This method, commonly known as interviews, establishes a direct line of communication between the researcher and the interviewee, whether conducted face-to-face, over the phone, or online. Within the framework of an interview. Despite the undeniable importance of interviews as a means of gathering rich qualitative data in research studies, the process of designing an effective interview can present challenges. The goal is to construct an interview that not only elicits unbiased responses but also ensures that the collected data is comprehensive and accurate. When designing a good interview, there is a need to think about different things. Questions should be created that let people share their thoughts freely, and there is also a need to make a comfortable space for open communication. It's important to be aware of possible biases and problems that can happen during the interview. This helps to keep the information collected honest and true. [58].

Interviews can be conducted in various formats, with three common types being structured, unstructured, and semi-structured interviews. Typically, researchers choose the format based on their specific needs, although unstructured and semi-structured interviews are often preferred for qualitative data collection.

The primary objective of conducting interviews was to obtain in-depth insights and perspectives directly from individuals with expertise.

After the survey, interviews were conducted with selective professionals to further summarize the problem and there solutions. Moreover, after a detailed discussion with each candidate, the study moved into the refining phase. That helped the research to shape into proposing guidelines.

3.4.1.1. Semi Structure Interviews

Mostly collecting qualitative data semi-structured interviews are used. In semi-structured interviews, interview questions are based on predetermined goals like research questions but the interviewers are also allowed to follow up with additional questions if needed to get clarifications. Mostly questions in semi structure interview are open-ended [58].

After collecting and analyzing result from the questionnaire challenges identified in distributed Scrum for requirement prioritization interview questions were organized to get the answer on how to mitigate the challenges and problems.

3.4.1.2. Participant Selection

Participant selection is one of the major step in any survey. The details of this section is given below.

Selection Criteria:

The selection of participants for the interviews was a careful process, guided by well-defined criteria. The carefully designed criteria were intended to identify people who either directly experienced or possessed the necessary essential expertise related to the field of distributed Scrum development. The goal was to ensure that the selected participants could provide meaningful insights and perspectives essential to achieving the research objectives.

3.4.1.3. Question Development

The formulation of interview questions was carefully considered to make sure it aligned with the research question. The questions were carefully designed to gather in-depth answers, allowing an in-depth examination of the targeted subjects that were related to requirement prioritization in distributed Scrum.

3.4.1.4. Modes of Interview

Interviews were conducted using a flexible approach, accommodating participants' availability and location. Both face-to-face and remote interview options were made available. Video calls like Google Meet and Zoom were utilized to take online interviews.

3.4.2. Focus Group:

Another common method is using focus groups, where researchers gather a bunch of people to talk about a topic together. In focus groups, researchers interact with a group of people at the same time, which makes it faster to share information compared to interacting with each person separately. Focus group was the last step of the research methodology in which proposed mitigation strategies were validated through focus group working in distributed Scrum.

3.4.3. Propose Guideline

Surveys and interview approaches were implemented to get information from experts about dealing with challenges in distributed Scrum. At first, 11 problems were found, but then 8 important ones were focused based on the survey results. The average weightage was used to decide not to consider 3 of the problems.

Later, interviews were conducted with experts to come up with helpful suggestions. Challenges were discussed in detail with experts. Their suggestions were like guides that could help others facing similar challenges. Then these guidelines were validated by a focus group.

3.5. Summary

This chapter discusses research methods in terms of quantitative and qualitative methods. Then the research methods used in this research are discussed in detail in terms of research context and justification. From the quantitative method, the survey as the dominant method is used. The purpose of the survey was to validate the challenges by practitioners, identified through LR from a qualitative method, then interviews were taken with practitioners to accomplish guidelines.

CHAPTER 4

ANALYSIS AND RESULTS

4.1. Literature Review Results

The following section shows the details of the results found by the LR process.

4.1.1. Search Results

In the systematic process of this research, a rigorous approach was employed for the identification, shortlisting, and meticulous sorting of search results, all predicated on specific, carefully queries.

A wide range of trusted online databases for our data, including well-known digital libraries like IEEE, Springer, and ACM. These sources helped explore the topic thoroughly and identify common issues that have been studied before. To keep findings current and relevant, only included research from the year 2012 and later.

Table 4.1: Search Result

Serial No	Source	Initial Screening	1st Filter	2nd Filter	Selected Articles
1	IEEE	187	92	41	33
2	ACM	60	33	20	5
3	Springer	30	19	8	3
4	Wiley's	5	2	1	2
5	Elsevier	5	3	1	2
6	Others	101	48	21	7
7	Total	388	197	92	52

This diagram shows the exact number of research papers that passed through each phase. How much are selected and how they were filtered down in each iteration.

4.1.2. Data Extraction and Synthesis Results

During this phase, the main focus has been on the data extracted from the articles by reading them. Applying quality assessment methods to extract all the relevant information from these papers brought down the following challenges. Most of the articles were sharing challenges because our selection criteria for these articles were narrowed down to the title of this article.

Table 4.2: List of identified Challenges

NO	Challenges
1	**Stakeholders participation**
2	**Product Owner technical knowledge constraints**
3	**Developers Involvement**
4	**Business value**
5	**Technical Limitation**
6	**Requirement dependencies**
7	**Time constraint**
8	**Budget constraint**
9	**Geographical and temporal limitation**
10	**Communication, coordination and collaboration**
11	**Distributed Scrum requirement prioritization constraint**

The Above table list all 11 challenges identified.

4.2. Survey Results

After having a comprehensive LR, a list of challenges was compiled. The list of challenges was found from LR related to requirement prioritization in distributed Scrum. Now to validate the data, from the professionals, a survey was conducted using Google Forms. The purpose of this survey is to make sure that everything identified from the articles is aligned with the latest industrial trends.

A total of 259 professionals have responded to the survey. The selection criteria were based on multiple factors including their demographics, nature of work, hierarchical position in the organization, and affiliation with the topic.

A 5-point Likert scale was utilized to present the core question, employing two distinct types of Likert scales. The first Likert scale comprised items related to the agreement, including 'Strongly Agree,' 'Agree,' 'Neutral,' 'Disagree,' and 'Strongly Disagree,' aimed at capturing respondents' perspectives on particular challenges. This scale was applied to address 13 questions concerning the identified challenges. The second Likert scale comprised "Always", "Frequently", "Sometimes", "Seldom" and "Never". This scale was applied to address 1 question concerning the identified challenge. The respondent responses are mentioned in Appendix B.

4.2.1. Respondents Profile

The survey was broken down to first profile the respondent based on his demographics, nature of work, and hierarchical position in his organization. Then later the survey questions were asked for a better in-depth understanding of the answers given.

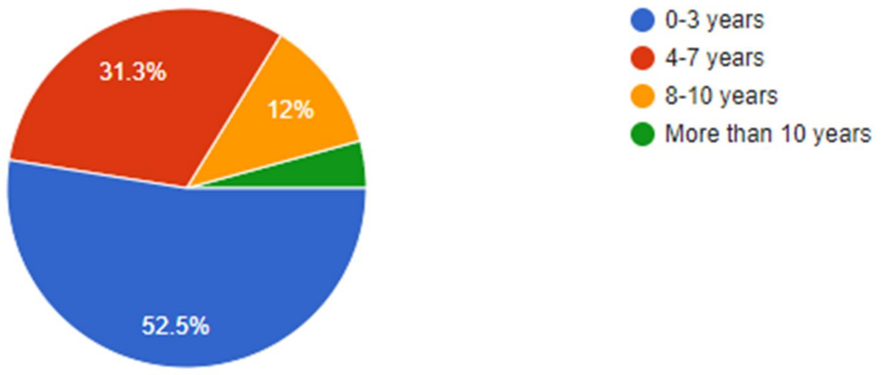

Figure 4.1: Percentage of Respondent's Experience

Table 4.3: Result of Responses from Survey

NO	Factors	Strongly Agree (2)	Agree (1)	Neutral (0)	Disagree (-1)	Strongly Disagree (-2)	Total (253)
1	**Stakeholders participation**	58*2=116	107*1=107	39*0=0	39*-1=-39	10*-2= -20	164
2	**Product Owner technical knowledge Constraint**	58*2=116	110*1=110	41*0=0	40*-1=-40	4*-2=-8	178
3	**Developers involvement**	72*2=144	123*1=123	31*0=0	18*-1=-18	9*-2=-18	231
4	**Business value**	29*2=58	106*1=106	65*0=0	41*-1=-41	12*-2=-24	99
5	**Technical limitation**	67*2=134	125*1=125	32*0=0	25*-1=-25	4*-2=-8	226
6	**Requirement Dependency**	74*2=148	118*1=118	29*0=0	30*-1=-30	2*-2=-4	232
7	**Time constraint**	48*2=96	122*1=122	40*0=0	27*-1=-27	15*-2=-30	161
8	**Budget constraint**	38*2=76	103*1=103	62*0=0	44*-1=-44	6*-2=-12	126
9	**Geographical and temporal limitation**	37*2=74	97*1=97	43*0=0	61*-1=-61	15*-2=-30	80

10	**Communication , coordination and collaboration**	72*2=144	109*1=109	35*0=0	34*-1=-34	3*-2= -1	213
11	**Distributed Scrum requirement prioritization constraints**	25*2=58	100*1=100	57*0=0	61*-1=-28	10*-2=-20	210

4.2.2. Results from Weightage Values

Average weightage is a numerical representation denoting the relative significance assigned to specific components, criteria, or factors within the study. These values serve as a quantified measure of the importance of each element in relation to the overall scoring and classification of data. The assignment of average weightage values facilitates the calculation of composite scores and rankings. These values hold significance in determining whether to accept or reject each factor.

The calculation of the average weightage value for each factor was achieved through the application of the Mean function. The calculation formula is as follows.

Avg weightage response = Weightage Value / Total No of responses

Following are the results of average weightage responses applied to each challenge.

Table 4.4: Calculated Average Weightage Response

NO	Factors	Weightage Value	Avg. Weightage Response
1	**Stakeholders participation**	**164**	**164/253= 0.64**
2	**Product Owner technical knowledge constraint**	**178**	**178/253=0.70**
3	**Developer's involvement**	**231**	**231/253=0.91**
4	**Business value**	**99**	**99/253=0.39**
5	**Technical limitation**	**226**	**226/253=0.89**

6	**Requirement dependency**	**232**	**232/253=0.91**
7	**Time constraint**	**161**	**161/253=0.63**
8	**Budget constraint**	**126**	**126/253=0.49**
9	**Geographical and temporal limitation**	**80**	**80/253=0.31**
10	**Communication, coordination and collaboration**	**213**	**213/253=0.84**
11	**Distributed Scrum requirement prioritization constraint**	**210**	**210/253=0.83**

4.2.3. Result in Sequence

Following the computation of average weightage responses, certain factors were accepted while others were rejected based on their respective average weightage results. Factors with an average value of 0.63 or higher on Likert scale were accepted. The final survey results, displaying the accepted and rejected factors in order, are presented in Table 4.8.

Table 4.5: Accepted or Rejected Results

NO	Factors	Weightage Value	Avg. Weightage Response	Results
1	**Stakeholders participation**	**164**	**164/253= 0.64**	**Accepted**
2	**Product Owner technical knowledge constraint**	**178**	**178/253=0.70**	**Accepted**
3	**Developer's involvement**	**231**	**231/253=0.91**	**Accepted**
4	**Technical Limitation**	**226**	**226/253=0.89**	**Accepted**
5	**Requirement dependency**	**232**	**232/253=0.91**	**Accepted**
6	**Time constraint**	**161**	**161/253=0.63**	**Accepted**

7	**Communication, coordination and collaboration**	**213**	**213/253=0.84**	**Accepted**
8	**Distributed Scrum requirement prioritization constraint**	**210**	**210/253=0.83**	**Accepted**
9	**Business value**	**99**	**99/253=0.39**	**Rejected**
10	**Budget constraint**	**126**	**126/253=0.49**	**Rejected**
11	**Geographical and temporal limitation**	**80**	**80/253=0.31**	**Rejected**

Out of 11 factors 8 factors that are Stakeholder participation, product owner technical knowledge constraint, developer's involvement, technical limitation, requirement dependency, time constraint, communication coordination and collaboration, distributed scrum requirement prioritization constraint have a value greater or equal to 0.63 according to defined criteria that why these factor got accepted. Whereas the remaining 3 factors that are business value, budget constraint, Geographical and temporal limitation got values less than the defined criteria got rejected.

4.3. Results Explanation

Analysis of survey results being done, there was the standard declared set just those challenges will be accepted that will have a value greater than or equal to 0.63 on the Likert scale in average weightage response. According to specified criteria, a total of 8 challenging factors from F1 to F8 out of 11 factors were accepted as their average weightage response value was greater than 0.63. Whereas factors from F9 to F11 got rejected due to their lower average response according to the defined criteria.

Cronbach's alpha value was computed in order to assess the consistency and dependability of the results obtained from the survey. The Cronbach alpha value calculated for the survey was "0.81".

4.3.1. Cronbach Alpha

Cronbach's alpha value should be computed in order to assess the consistency and dependability of the results obtained from the survey. Cronbach alpha helped in proving that the scales used to collect data are consistent and the data's results are trustworthy. Greater than "0.70" is considered to be a reliable Cronbach alpha value. The Cronbach alpha value calculated for the survey was “0.81”.

4.3.2. Low Significance Value

(i) The business value constraint with an average value of 0.39 was rejected as most of the respondents have responded to neutral (neither agree nor disagree). The reason behind this is that business value is an important factor in Scrum but it is not only one criterion.

(ii) The budget constraint was rejected as it got an average value of 0.49. The reason behind that is mostly the budget is already finalized in the initialization time of the project.

(iii) The geographical and temporal difference challenge was rejected by the value of 0.31 as most respondents disagreed with this challenge.

4.3.3. High Significance Value

Out of 11 challenges total 8 factors were accepted after the survey.

(i) All stakeholders not taking part at the time of requirement prioritization is the challenge accepted with value of 0.64

(ii) Mostly the product owner does not have enough technical knowledge due to which he could not prioritize requirements without team collaboration is also a challenge which is accepted by the value of 0.70.

(iii) The development team should always participate at the time of requirement prioritization as they have enough technical knowledge is a challenging factor in distributed Scrum is challenging factor accepted by the value 0.91.

(iv) The technical constraint is accepted by a value of 0.89.

(v) The dependency between different requirement constraint challenges is accepted by the value of 0.91.

(vi) Time constraint is the main challenge in distributed Scrum as it is accepted by the value 0.63.

(vii) The biggest challenge at the time of requirement prioritization in distributed Scrum is coordination, communication collaboration between the team this challenge is accepted by the value of 0.84

(viii) Requirement Prioritization is a challenging task in distributed Scrum and is accepted by the value of 0.83.

4.4. Discussion and Results

This study is carried out by a thorough literature analysis as part of our study to determine the difficulties faced by distributed scrum teams during requirement prioritization. Then industry survey was carried up with distributed Scrum practitioners to confirm the issues found in LR. Then interviews were conducted to find out the solutions for the challenges. In the end, guidelines were proposed.

4.4.1. Interview Questions

A series of thoughtfully designed questions was presented to our esteemed interviewees. The primary objective of this approach was to elicit a comprehensive and nuanced perspective from industry experts who offer invaluable insights and experiences pertaining to our research topic. These questions were not merely posed; rather, they were meticulously elaborated upon to establish a shared understanding between the interviewees and our research team. As experts in the field, our interviewees critically examined each question to fully grasp the nuances, context, and relevance of the inquiries. This mutual understanding not only fostered a more fruitful exchange of information but also ensured that the responses collected would be both insightful and aligned with the core objectives of our research. This rigorous and thoughtful approach to presenting and comprehending the questions underscores our unwavering commitment to extracting profound insights from the wealth of expertise provided by industry professionals.

Table 4.6: Interview Questions

Serial Number	Question
1	All stakeholders should participate in requirement prioritization for Distributed Scrum.
2	Who is the most knowledgeable member of requirement prioritization?
3	The product owner does not have enough domain knowledge to prioritize requirements
4	The development team has all the required knowledge. If they participate, will that help in requirement prioritization?
5	What is the most important constraint to be considered for requirement prioritization?
6	Technical constraints are also considered while prioritizing requirements
7	The dependencies between different requirements are also considered in requirement prioritization.
8	In the requirement prioritization of product backlog, time constraints are also considered.
9	One of the biggest challenges in distributed scrum is communication and coordination.
10	What causes most issues in distributed scrum teams when doing requirement prioritization?
11	What is the importance of requirement prioritization in distributed scrum?
12	Is requirement prioritization in distributed scrum a challenging task?
13	Lack of collaboration among distributed scrum teams causes significant issues during the requirement engineering phases.

4.4.2. Challenges and Proposed Solution

Following a meticulous evaluation of the interview transcripts, a set of significant challenges emerged as prominent themes. These challenges were identified through a thorough and in-depth analysis of the insights garnered during the interviews, characterized by a critical examination of the data. Each challenge was meticulously selected, with a deliberate focus on those that recurrently surfaced and held critical importance in the context of our research.

Moreover, our analysis extended beyond the identification of challenges to the formulation of actionable solutions. Two to three pragmatic and implementable solutions were thoughtfully proposed as a result of the comprehensive examination of the interview data. These solutions are rooted in a deep understanding of the challenges faced, serving as pragmatic pathways to address the issues and improve the existing practices. By bridging the gap between identification and solution, our research not only highlights critical industry challenges but also offers a tangible and forward-looking approach to overcoming these obstacles, thereby contributing to the advancement of best practices in the field.

Table 4.7: Proposed Guidelines

Sr. #	**Challenges**	**Proposed solutions**
1	Stakeholders participation	Use some online method where all stakeholders can participate.
		Stakeholder’s participation should happen in a sync environment.
		One common person can gather all prioritization information from team so they can be considered together when actual prioritization happens
		Focus on having all information about requirements with everyone so that everyone has a clear understanding of each requirement. That can help in requirement prioritization.
2	Product owner technical knowledge constraints	Product owner should validate requirements with development team to have their side of the story as well
		Rather than having only an overview of the requirements, the product owner should explore in-depth details for better technical understanding
		Product owner should have some technical background
3	Developers involvement	Developers should be part of requirement-gathering process to access the complexity and technical constraints
		Developers should get involved at very last stages when most of the requirements are clear to have a technical assessment.
		Developers should be the part of requirement prioritization at least one nominated person from the development team should be present
		Developers should participate in business requirement discussion to help identify technical constraints

4	Technical limitations	Technical limitations should be accessed while doing the requirement gathering and requirement prioritization to reduce software failure.
		Technical dependencies should be addressed as well
		All requirements should have a dependency score and limitation score to make them easily identifiable.
		A technical person should be always present in the process of requirement prioritization.
5	Requirement dependencies	Assign dependency score to everything in the product backlog
		Involve technical experts to access technical dependencies which will ensure software success.
		Similar dependency score requirements should be grouped to understand product backlog complexity.
		Highest dependency score requirements should be addressed first from product backlog
6	Time constraint	All stakeholders should do time estimations of requirements.
		Time constraints should also be given some value while prioritizing requirements for a sprint.
		Have some time buffer in each requirements to make sure we have some extra time to solve any unexpected challenge in the sprint.
7	Communication, Coordination & Collaboration	There should be at least 3-4 hours of time overlap in the distributed team.
		All team members should not diverge more than 3 hours of time in time zones for easier communication.
		Consider using async communication tools like Microsoft Team and Slack.
		Minimize the collaboration required. Build processes that uses documentation and lesser human involvement all the time.
8	Distributed Scrum Requirement Prioritization Constraints	Prioritize requirements scores should be really comprehensive. They should include all technical, non-technical, and geographical constraints.
		Every stakeholder should participate in the requirement prioritization process in distributed scrum
		Use a requirement prioritization framework so everyone can follow a guideline

4.5. Validation of Guidelines through Focus Group

The proposed guidelines were validated through a focus group. 5 participants participated in the process of a focus group. The thesis topic was briefly discussed with the focus group the experts shared their thoughts, ideas and knowledge. Questions that were asked in the focus group are mentioned in Appendix C. The responses that are gathered from the focus group are mentioned in Appendix D. The Likert scale values for focus group responses are represented in Appendix E.

4.5.1. Focus Group Results

The response values were then converted into the Likert scale and average weightage values were calculated. The threshold value was 0.96 the factors whose value was less than 0.96 were rejected and the factors having a value greater than 0.96 were accepted. 5 solutions out of 29 were rejected.

Table 4.8: Guidelines rejected by focus group

1	Stakeholder's participation should happen in a sync environment.
2	Developers should participate in business requirement discussion to help identify technical Constraints.
3	Similar dependency score requirements should be grouped to understand product backlog complexity.
4	All stakeholders should do time estimations of requirements
5	All team members should not diverge more than 3 hours of time in time zone for easier communication.

4.5.2. Final Guidelines

Table 4.9:Accepted guidelines by focus group

Sr. #	Challenges	Proposed solutions
1	Stakeholders participation	Use some online method where all stakeholders can participate.
		One common person can gather all prioritization information from team so they can be considered together when actual prioritization happens
		Focus on having all information about requirements with everyone so that everyone has a clear understanding of each requirement. That can help in requirement prioritization.
2	Product owner technical knowledge constraints	Product owner should validate requirements with development team to have their side of the story as well
		Rather than having only overview of the requirements, product owner should explore in-depth details for a better technical understanding
		Product owner should have some technical background
3	Developers involvement	Developers should be part of requirement-gathering process to access the complexity and technical constraints
		Developers should be the part of requirement prioritization at least one nominated person from the development team should be present
		Developers should get involved at very last stages when most of the requirements are clear to have a technical assessment.
4	Technical limitations	Technical limitations should be accessed while doing the requirement gathering and requirement prioritization to reduce software failure.
		Technical dependencies should be addressed as well
		All requirements should have a dependency score and limitation score to make them easily identifiable.
		A technical person should be always present in the process of requirement prioritization.
5	Requirement dependencies	Assign dependency score to everything in the product backlog
		Involve technical experts to access technical dependencies which will ensure software success.
		Highest dependency score requirements should be addressed first from product backlog
6	Time constraint	Time constraints should also be given some value while prioritizing requirements for a sprint.
		Have some time buffer in each requirements to make sure we have some extra time to solve any unexpected challenge in the sprint.

7	Communication, Coordination & Collaboration	There should be at least 3-4 hours of time overlap in the distributed team.
		Consider using async communication tools like Microsoft Team and Slack.
		Minimize the collaboration required. Build processes that uses documentation and lesser human involvement all the time.
8	Distributed Scrum Requirement Prioritization Constraints	Prioritize requirements scores should be really comprehensive. They should include all technical, non-technical, and geographical constraints.
		Every stakeholder should participate in the requirement prioritization process in distributed scrum
		Use a requirement prioritization framework so everyone can follow a guideline

4.6. Comparison with Existing Literature

Following table has a comprehensive comparison between research work and existing literature. Base paper 1 and 2 were the most relevant research papers for the comparison.

Table 4.10: Comparison table for existing literature

Aspect	**Research Thesis**	**Base paper 1**	**Base paper 2**
Title	Analysis of Requirements Prioritization in Distributed Scrum for Reducing Software Failure.	Requirements Prioritization in Agile Projects: from Experts' Perspective.	Agile Requirements Prioritization in Practice: Results of an Industrial Survey.
Research Questions	Q1. What are the challenges regarding requirement prioritization in distributed Scrum that could lead to software failure?	Q1.When does requirements prioritization take place?	Q1. What criteria and techniques are applied during prioritization?

	Q2. What could be the possible guideline for solving the requirement prioritization challenges in distributed Scrum?	Q2. What aspects and techniques are applied during prioritization?	Q2. Who participates in prioritization tasks?
Agile Methodology on which Research Carried Out	Distributed Scrum.	Scrum.	Scrum and Kanban.
Methodology	Survey conducted through questionnaire, interviews and results validated through focus group.	Survey conducted through questionnaire.	Survey conducted through questionnaire.
No of Respondent for Survey	259 respondent for questionnaire and 5 respondents for interviews and focus group respectively.	30 respondent for questionnaire.	69 respondent for questionnaire.
Contribution	Development of practical guidelines and strategies to effectively address identified challenges in distributed Scrum for requirements prioritization.	The findings of this study can be used by practitioners to make decisions about RP activities for IT projects they participate in, and by researchers to plan more focused studies investigating the causes and contextual factors behind the practices declared by the respondents.	This paper provides an update on how RP is done in Scrum and Kanban practice on the basis of survey responses gathered from 69 IT industry practitioners.

Limitation	The proposed solutions are not properly validated with the industry yet, they need to go through a rigorous process in industry to be validated.	This survey study is associated with mono-method bias, as the only source of data are the answers of survey respondents. By making anonymous survey, the researchers tried to minimize threats of guessing answers and providing false, "better-looking" answers, but the researchers cannot totally exclude such possibilities.	Even when trying to reach people from various company types and application domains, it is hard to ensure that the sample represents the whole population and even if it is representative of the IT industry, there can be differences between different countries.
Future Work	Future research include the potential for conducting an industrial case study to assess the practical application and effectiveness of the proposed guidelines in real-world requirement prioritization in distributed Scrum environments.	The possible future work includes a more in-depth analysis of rationales behind RP practices used and their consequences; using other research methods like the case studies; and attracting more respondents from other countries to increase the accuracy of the results.	Using other research techniques, such as case studies, the potential future study entails a deeper examination of the justifications for the RP approaches employed and their outcomes. In particular, reported practices that potentially conflict with guidelines of Agile methods seem to be an interesting research direction.

4.7. Comparison of Findings with Existing Literature

Following table is a comprehensive comparison between the findings of research work and existing literature. Base paper 1 and 2 are the most relevant research papers for the comparison.

Table 4.11: Comparison of Findings with Existing Literature

Sr. #	Challenges	Proposed Solutions from thesis	Base paper 1	Base paper 2
1.	Stakeholders Participation	Use online methods for stakeholder participation. Centralize prioritization information. Ensure all stakeholders have a clear understanding of requirements.	Not directly mentioned of stakeholder participation methods, but emphasis on the involvement of PO, Scrum Master, and customer representatives.	Stakeholder participation is crucial and typically involves all relevant parties, including vendors and technical experts.
2.	Product Owner Technical Knowledge	Product owners should validate requirements with the development team. Gain in-depth technical knowledge of requirements. Product owners should have a technical background.	Product owner involvement is crucial, and their technical knowledge helps in better understanding and validation of requirements.	Product owner's technical understanding is important but not always emphasized in the practices studied.
3.	Developers Involvement	Developers should be part of the requirement-gathering process. Developers should get involved in the later stages for technical assessment. At least one developer should be involved in	Developer involvement in RP is necessary, especially in assessing the technical complexity of requirements. Their participation is mutual with customer representatives, enhancing the	Developers are typically involved in the requirement prioritization process but may not always participate from the start.

		requirement prioritization.	understanding of technical constraints.	
4.	Technical Limitations	Address technical limitations during requirement gathering and prioritization. Assign dependency and limitation scores to requirements. Involve technical experts in prioritization.	Technical constraints are considered in RP, especially concerning NFRs. Ignoring NFRs can lead to rework, increased costs, and time, hence they should be prioritized during RP.	Technical dependencies are considered but may not always be addressed systematically.
5.	Requirement Dependencies	Assign dependency scores to all requirements. Group requirements with similar scores. Address high-dependency requirements first	Dependency scores and technical constraints are essential in understanding backlog complexity. High-dependency requirements should be prioritized to reduce project risks and ensure smooth progression.	Dependencies are acknowledged; focus on business value and interdependencies, but scoring and grouping may vary.
6.	Time Constraint	Time constraints should be factored into prioritization. Include time buffers in requirements to handle unexpected challenges.	*Time constraint are not addressed as the study is based on Scrum requirement prioritization challenges*	*Time constraint are not addressed as the study is based on Scrum and Kanban requirement prioritization challenges*
7.	Communication, Coordination & Collaboration	Ensure at least 3-4 hours of overlap in distributed teams. Use	*Communication, coordination and collaboration*	*Communication, coordination and collaboration*

		async communication tools like Microsoft Teams and Slack. Minimize collaboration needs with effective documentation.	*challenges are of distributed Scrum so they are not addressed in this study*	*challenges are of distributed Scrum so they are not addressed in this study*
8.	Distributed Scrum Requirement Prioritization	Use comprehensive prioritization scores considering all constraints. All stakeholders should participate in RP. Use a prioritization framework for consistency.	*Distributed Scrum requirement prioritization challenge has been not addressed in the study.*	*Distributed Scrum requirement prioritization challenge has been not addressed in the study.*

4.8. Summary

In this chapter, all the challenges identified through LR are mentioned. After that analysis of the survey is done in order to get the accepted or rejected factors. Then guidelines are proposed through interviews against each accepted factor. Those guidelines are future evaluated through focus group members so that they can be implemented in the industry for the reduction of software failure.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1. Introduction

This chapter provides a summary of all the study's findings along with a brief analysis of the RQ1 and RQ2 contributions made in the thesis. The future work is also discussed in this chapter.

Through a thorough review of the literature, the most frequent issues that distributed Scrum teams faced at the time of requirement prioritization were determined. To confirm the difficulties found by LR, an industrial survey was carried out. Then, by combining the best strategies used by practitioners to address the challenges, guidelines were proposed to mitigate the challenges to overcome the software failure. Then expert review was conducted to validate the guidelines.

In this final chapter, a detailed study on how requirements are prioritized in distributed Scrum ended. The study carefully looked at how organizations handle prioritizing tasks when using Scrum with teams spread across different locations. As we think about the many insights we've gained and the new solutions we've suggested, this chapter is the highlight of our work. It brings together all our research findings, and their wider impacts, and sets the stage for future studies in this constantly changing field.

In this chapter, we focus on the worldwide importance of our research. We will not only summarize the key findings but also highlight the future of agile methods in teams spread across different locations. This is a crucial point where the emphasize was on how our work can lead to more innovation and better practices in this constantly changing field.

5.2. Reviewing Research Questions

RQ1: What are the current challenges regarding requirement prioritization in distributed Scrum that could lead to software failure?

Our first research question focused on the challenges of prioritizing requirements in distributed Scrum. To answer this, a detailed approach was used. The study started with a literature review (LR). This thorough process included creating a detailed plan. The LR method involved what studies to include or exclude.

Carefully the data was collected and organized key findings from these selected studies, putting them into tables for easier understanding. These findings were thoroughly documented, providing deep insights into the challenges faced. The literature review uncovered a wide range of challenges.

To validate and align these findings with real-world experiences, an industrial survey was conducted with practitioners actively engaged in distributed Scrum. This survey employed dual Likert scales for data collection, ensuring an accurate and reliable dataset. To gauge the survey's reliability and precision, Cronbach alpha's value was calculated. Further, a criterion was defined to accept or reject each challenge based on the average weighted response obtained through the survey. Out of the 12 challenges identified in the survey, eight were accepted.

RQ2: What could be the possible guidelines for solving the requirement prioritization challenges in distributed Scrum?

Our second research question sought to find potential guidelines and strategies to mitigate the challenges encountered in requirement prioritization within distributed Scrum that could lead to software failure. This question led to an extensive exploration of existing literature, focusing on mitigation strategies adopted by distributed Scrum teams. Our literature review (LR) served as the cornerstone for uncovering these strategies, gathered from a diverse array of studies conducted by the distributed agile community. The strategies identified were distilled and consolidated to create a framework that encapsulated the most effective practices, validated by both industry professionals and practitioners.

5.3. Research Contribution

The research contribution from this study:

- Identification of Challenges in Requirement Prioritization in Distributed Scrum: Thorough review of existing literature and real-world analysis to pinpoint challenges in distributed Scrum requirement prioritization which leads to software failure.
- Validation from Industry Experts on Identified Issues: Expert interviews and consultations to confirm identified challenges' authenticity and enrich our practical understanding.
- Proposed Solutions for Mitigating Challenges: Development of practical guidelines and strategies to effectively address identified challenges in distributed Scrum.
- Validation of the Proposed Guidelines with Industry Experts: Collaborative validation with experts to ensure the practicality and relevance of proposed solutions for real-world application.

5.4. Limitations

While our study adhered to rigorous research protocols, certain limitations should be acknowledged:

- The research is conducted with limited number of companies only operating only in certain field of work
- The proposed solutions are not properly validated with the industry yet, they need to go through a rigorous process in industry to be validated.
- The respondents to the survey were not equally divided based on their designation. More number of people from one profession could have influenced the results

5.5. Future Work

Future research include the potential for conducting an industrial case study to assess the practical application and effectiveness of the proposed guidelines in real-world requirement prioritization in distributed Scrum environments. This could further illuminate the path toward sustainable

Requirement Prioritization practices and inform the continuous improvement of agile methodologies in a distributed environment.

References

[1] T. Shah and S. v Patel, “A Review of Requirement Engineering Issues and Challenges in Various Software Development Methods,” International Journal of Computer Applications, vol. 99, no. 15, pp. 36–45, Aug. 2014.

[2] R. B. Svensson and R. Torkar, “Not All Requirements Prioritization Criteria Are Equal at All Times: A Quantitative Analysis,” Apr. 2021.

[3] Apoorva Srivastava, Sukriti Bhardwaj, Shipra Saraswat, “SCRUM model for agile methodology,” IEEE International Conference on Computing, Communication and Automation (ICCCA), 2017.

[4] Murat Dogus Kahya, Çağla Seneler, “Geographical Distance Challenges in Distributed Agile Software Development: Case Study of a Global Company,” IEEE 3rd International Conference on Computer Science and Engineering (UBMK), 2018.

[5] Kleophas Model, Carolin Mombrey, Georg Herzwurm,“Paving the Way to a Software-Supported Requirements Prioritization in Distributed Scrum Projects,” IEEE/ACM International Workshop on Software-Intensive Business (IWSiB), 2022.

[6] Ville T. Heikkilä, Daniela Damian, Casper Lassenius, Maria Paasivaara, “A Mapping Study on Requirements Engineering in Agile Software Development,” IEEE 41st Euromicro Conference on Software Engineering and Advanced Applications, 2015.

[7] Leonardo Sanches dos Santos, Alexandre L'Erario, Tiago Pagotto, Joao Ricardo Moreno Camilo, Fabricio Sousa Oliveira, Jose Augusto Fabri, “A SCRUM-Based Process to Distributed Projects in Multidisciplinary Teams: A Case Study,” IEEE/ACM 13th International Conference on Global Software Engineering (ICGSE), 2018.

[8] Victor Temitayo Faniran, Abdulbaqi Badru, Nurudeen Ajayi, “Adopting Scrum as an Agile approach in distributed software development: A review of literature,” IEEE 1st International Conference on Next Generation Computing Applications (NextComp), 2017.

[9] Ibrahim Seckin, Tolga Ovatman, “An Empirical Study on Scrum Application Patterns in Distributed Teams,” IEEE/ACM 13th International Conference on Global Software Engineering (ICGSE), 2018.

[10] F. Hujainah, R. B. A. Bakar, M. A. Abdulgabber, and K. Z. Zamli, “Software Requirements Prioritization: A Systematic Literature Review on Significance, Stakeholders, Techniques and Challenges,” IEEE Access, vol. 6, pp. 71497–71523, 2018.

[11] Juan Carlos B. Somohano-Murrieta, Jorge Octavio Ocharán-Hernández, Angel J. Sánchez-García, Maria de los Ángeles Arenas-Valdés, “Requirements Prioritization Techniques in the last decade: A Systematic Literature Review,” IEEE 8th International Conference in Software Engineering Research and Innovation (CONISOFT), 2020.

[12] Raneem Qaddoura, Alaa Abu-Srhan, Mais Haj Qasem, Amjad Hudaib, “Requirements Prioritization Techniques Review and Analysis,” IEEE International Conference on New Trends in Computing Sciences (ICTCS), 2017.

[13] Kaaenat Ali; Junaid Ali Khan, Farwah Aizaz, Mansoor Ahmed, “Software Requirements Prioritization in the context of Global Software Development,” International Conference on Frontiers of Information Technology (FIT), 2021.

[14] Heena Ahuja; Sujata, G. N. Purohit, “Understanding Requirement Prioritization Techniques,” IEEE International Conference on Computing, Communication and Automation (ICCCA), 2016.

[15] Shruti Sharma, Nitasha Hasteer, “A Comprehensive Study on State of Scrum Development,” IEEE International Conference on Computing, Communication and Automation (ICCCA), 2016.

[16] Youry Khmelevsky, Xitong Li, Stuart Madnick, “Software Development Using Agile and Scrum in Distributed Teams,” Annual IEEE International Systems Conference (SysCon), 2017.

[17] Pernille Lous, Marco Kuhrmann, Paolo Tell “Is Scrum Fit for Global Software Engineering?,’’ IEEE 12th International Conference on Global Software Engineering (ICGSE), 2017.

[18] Muhammad Younas, Dayang Norhayati Abang Jawawi, Muhammad Arif Shah, Ahmad Mustafa, Muhammad Awais, Muhammad Kamran Ishfaq, Karzan Wakil, “Elicitation of

Nonfunctional Requirements in Agile Development Using Cloud Computing Environment," IEEE Access, 2020.

[19] Saman Tariq, Ahmad Ibrahim, Ali Usama, M. Saad Shahbaz, " An Overview of Requirements Elicitation Techniques in Software Engineering with a focus on Agile Development" IEEE 4th International Conference on Computing & Information Sciences (ICCIS), 2021.

[20] Asma Batool, Yasir Hafeez Motla, Bushra Hamid, Sohail Asghar, Muhammad Riaz, Mehwish Mukhtar, Mehmood Ahmed, "Comparative Study of Traditional Requirement Engineering and Agile Requirement Engineering" IEEE 15th International Conference on Advanced Communications Technology (ICACT), 2013.

[21] Ahmed Mateen, Khizar Abbas, Muhammad Azeem Akbar, "Robust Approaches, Techniques and Tools for Requirement Engineering in Agile Development," IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI), 2017.

[22] Glasow, Priscilla A Fundamentals of survey research http://www.uky.edu/~kdbrad2/EPE619/Handouts/SurveyResearchReading.pdf

[23] Rana Muhammad Dilshad, Muhammad Ijaz Latif, "Focus Group Interview as a Tool for Qualitative Research: An Analysis," Pakistan Journal of Social Sciences (PJSS) Vol. 33, No. 1 (2013), pp. 191-198, 2013.

[24] R. Goldberg, "Software engineering: An emerging discipline," 1986.

[25] E. M. Lavrishcheva, "SOFTWARE ENGINEERING AS A SCIENTIFIC AND ENGINEERING DISCIPLINE," 2008.

[26] G. Booch, "The History of Software Engineering."

[27] T. Shah and S. v Patel, "A Review of Requirement Engineering Issues and Challenges in Various Software Development Methods," *International Journal of Computer Applications*, vol. 99, no. 15, pp. 36–45, Aug. 2014.

[28] I. Zafar, A. Shaheen, A. K. Nazir, B. Maqbool, W. H. Butt, and J. Zeb, "Why Pakistani Software Companies don't use Best Practices for Requirement Engineering Processes," in *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, Nov. 2018.

[29] B. Nuseibeh and S. Easterbrook, "Requirements Engineering: A Roadmap," 2000.

[30] R. B. Svensson and R. Torkar, “Not All Requirements Prioritization Criteria Are Equal at All Times: A Quantitative Analysis,” Apr. 2021.

[31] Aleksander Jarzębowicz*, Natalia Sitko Agile, “Requirements Prioritization in Practice: Results of an Industrial Survey” ELESVIER 24th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems, 2020.

[32] Noor Hazlini Borhan, Hazura Zuzalil, Sa’adah Hassan, Nor Hayati Mohd Ali “Requiremnets Prioritization in Agile Projects: From Expert’ Perspective,” Journal of Theoretical and Applied Information Technology 15th October 2022. Vol.100. No 19 © 2022 Little Lion Scientific, 2022.

[33] Noor Hazlini Borhan, Hazura Zulzalil, Sa’Adah Hassan, Norhayati Mohd Ali “Requirements Prioritization Techniques Focusing on Agile Software Development: A Systematic Literature Review” INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH, VOL 8, ISSUE 11, NOVEMBER 2019.

[34] Najia Saher, Fauziah Baharom and Rohaida Romli, “A Review of Requirement Prioritization Techniques in Agile Software Development,” Knowledge Management International Conference (KMICe) 2018, 25 –27 Julyt 2018, Miri Sarawak, Malaysia, 2018.

[35] Zornitza Racheva, Maya Daneva, Andrea Herrmann, Roel J. Wieringa, “A conceptual model and process for client-driven agile requirements prioritization,” IEEE Fourth International Conference on Research Challenges in Information Science (RCIS), 2010.

[36] Noor Hazlini Borhan, Hazura Zulzalil, Sa’Adah Hassan, Norhayati Mohd Ali “A Hybrid Prioritization Approach by integrating non-Functional and Functional User Stories in Agile-Scrum Software Development (i-USPA):A preliminary study,” IEEE International Conference on Computing (ICOCO), 2022.

[37] Ville T. Heikkilä, Daniela Damian, Casper Lassenius, Maria Paasivaara, “A Mapping Study on Requirements Engineering in Agile Software Development,” IEEE 41st Euromicro Conference on Software Engineering and Advanced Applications, 2015.

[38] Hamzah Alaidaros, Ahmed Bakodah, Salim F. Bamsaoud, “A Review on Requirements Prioritization Approaches of Software Project Management,” IEEE International Conference on Intelligent Technology, System and Service for Internet of Everything (ITSS-IoE), 2022.

[39] Woogon Shim, “An Agile Method of Representing, Organizing, and (Re) Prioritizing Requirements in a Large Enterprise,” IEEE 27th International Requirements Engineering Conference (RE), 2019.

[40] Ethan Hadar, Amin Hassanzadeh, “Big Data Analytics on Cyber Attack Graphs for Prioritizing Agile Security Requirements,” IEEE 27th International Requirements Engineering Conference (RE), 2019.

[41] Sanjaya Kumar Saxena, Rachna Chakraborty, “Decisively: Application of Quantitative Analysis and Decision Science in Agile Requirements Engineering,” IEEE 22nd International Requirements Engineering Conference (RE), 2014.

[42] Zornitza Racheva, Maya Daneva, Klaas Sikkel, Roel Wieringa, Andrea Herrmann, “Do we Know Enough about Requirements Prioritization in Agile Projects: Insights from a Case Study,” IEEE 27th International Requirements Engineering Conference (RE), 2010.

[43] Heera Sheemar, Gurpreet Kour, “Enhancing User-Stories Prioritization Process in Agile Environment,” IEEE International Conference on Innovations in Control, Communication and Information Systems (ICICCI), 2017.

[44] Najia Saher, Fauziah Baharom, Rohaida Romli, “Guideline for the Selection of Requirement Prioritization Techniques in Agile Software Development: An Empirical Research,” International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-8 Issue-5, January 2020.

[45] Nikhil Govil, Ashish Sharma, “Information Extraction on Requirement Prioritization Approaches in Agile Software Development Processes,” IEEE 5th International Conference on Computing Methodologies and Communication (ICCMC), 2021.

[46] Joseph Gillain, Ivan Jureta, Stephane Faulkner, “Planning Optimal Agile Releases via Requirements Optimization,” IEEE 24th International Requirements Engineering Conference Workshops (REW), 2016.

[47] Rashmi Popli, Naresh Chauhan, Hemant Sharma, “Prioritising User Stories In Agile Environment,” IEEE International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT), 2014.

[48] Md Shamsur Rahim, AZM Ehtesham Chowdhury, Shovra Das, “RIZE: A Proposed Requirements Prioritization Technique for Agile Development,” IEEE Region 10 Humanitarian Technology Conference (R10-HTC), 2017.

[49] Amjad Hudaib, Fatima Alhaj, “Self-Organizing Maps for Agile Requirements Prioritization,” IEE 2nd International Conference on new Trends in Computing Sciences (ICTCS), 2019.

[50] Disman; Mohammad Ali, M. Syaom Barliana, “THE USE OF QUANTITATIVE RESEARCH METHOD AND STATISTICAL DATA ANALYSIS IN DISSERTATION: AN EVALUATION STUDY” International Journal of Education Vol. 10 No. 1, pp. 46-52, August. 2017.

[51] Ashatu Hussein, “The use of Triangulation in Social Sciences Research: Can qualitative and quantitative methods be combined,” Journal of Comparative Social Work 2009/1, 2009.

[52] Nancy L. Leech, Amy B. Dellinger, Kim B. Brannagan, Hideyuki Tanaka , “Evaluating Mixed Research Studies:A Mixed Methods Approach,” Journal of Mixed Methods Research 4(1) 17–31 ªThe Author(s), 2010.

[53] André Queirós, Daniel Faria, Fernando Almeida, “STRENGTHS AND LIMITATIONS OF QUALITATIVE AND QUANTITATIVE RESEARCH METHODS” European Journal of Education Studies ISSN: 2501 - 1111

[54] M. Kasunic, “Designing an effective survey. Handbook. Carnegie Mellon University,” Softw. Eng. Inst., no. September, p. 140, 2005.

[55] Diane Strode, “Agile methods: a comparative analysis,” Annual Conference of the National Advisory Committee on Computing Qualifications (NACCQ 2006), 2006

[56] Ken Schwaber and Jeff Sutherland, “The Definitive Guide to Scrum: The Rules of the Game,” 2017.

[57] Ellie Fossey, Carol Harvey, Larry Davidson, "Understanding and Evaluating Qualitative Research" Volume 36, Issue 6.

[58] Hamed Taherdoost, "How to Conduct an Effective Interview; A Guide to Interview Design in Research Study Authors" International Journal of Academic Research in Management (IJARM), 11 (1), pp.39-51. hal-03741838ff, 2022.

[59] H. Taherdoost, "Sampling Methods in Research Methodology; How to Choose a Sampling Technique for Research," SSRN Electron. J., vol. 5, no. 2, pp. 18– 27, 2018.

[60] Rietz, T., Schneider, "We See We Disagree: Insights from Designing a Cooperative Requirements Prioritization System." Proceedings of the 28th European Conference on Information Systems (ECIS 2020). Marrakesh, Morocco, June 15th-17th, 2020.

Appendix A: Survey Questionnaire

Demographic Questions

1. What is your role in the organization?
- Product Owner
- Scrum Master
- Project Manager
- Developer
- Other:
2. Your experience in distributed scrum.
- 0-3 years
- 4-7 years
- 8-10 years
- More than 10 years
3. What is the size of your team?
- Less than 15

- 16-25
- 25-35
- More Than 35

Core Questions

4. Which stakeholders mostly take part in requirement prioritization?
    a. Check all that apply.
    b. Product Owner Client
    c. Scrum Master
    d. Development Team
    e. All of these

5. All stakeholders should participate in requirement prioritization for distributed scrum.
    a. Check all that apply.
    b. Product Owner Client
    c. Scrum Master
    d. Development Team
    e. All of these
6. The product owner does not have enough technical knowledge to prioritize requirements.
    a. Strongly Agree
    b. Agree
    c. Neutral Disagree
    d. Strongly Disagree
7. The development team has all the required technical knowledge. If they participate, will that help in requirement prioritization
    a. Strongly Agree
    b. Agree
    c. Neutral Disagree
    d. Strongly Disagree

8. The business value is often the only prioritization criterion in distributed scrum development
    a. Strongly Agree
    b. Agree
    c. Neutral Disagree
    d. Strongly Disagree.
9. The technical constraints are also considered while prioritizing requirements.
    a. Strongly Agree
    b. Agree
    c. Neutral Disagree
    d. Strongly Disagree
10. The dependency between different requirements is also considered during requirement prioritization. *
    a. Strongly Agree
    b. Agree
    c. Neutral Disagree
    d. Strongly Disagree
    e. Strongly Disagree
11. In the requirement prioritizing phase of product backlog the time constraints are also considered
    a. Strongly Agree
    b. Agree
    c. Neutral Disagree
    d. Strongly Disagree
12. In the requirement prioritizing phase of product backlog does the budget constraints are considered.
    a. Strongly Agree
    b. Agree
    c. Neutral
    d. Disagree
    e. Strongly Disagree

13. Does your organization use any online tool for the product backlog?
    a. Yes
    b. No
14. The geographical distance among members of the Scrum team leads to various challenges.
    a. Strongly Agree
    b. Agree
    c. Neutral
    d. Disagree
    e. Strongly Disagree
15. The temporal distance among members of the Scrum team leads to various challenges.
    a. Strongly Agree
    b. Agree
    c. Neutral
    d. Disagree
    e. Strongly Disagree
16. One of the biggest challenges in distributed scrum is communication and coordination.
    a. Strongly Agree
    b. Agree
    c. Neutral
    d. Disagree
    e. Strongly Disagree
17. One of the biggest challenges in distributed scrum is communication and coordination.
    a. Strongly Agree
    b. Agree
    c. Neutral
    d. Disagree
    e. Strongly Disagree
18. Lack of collaboration among distributed scrum teams causes significant issues while requirement engineering phases.
    a. Strongly Agree
    b. Agree

c. Neutral
d. Disagree
e. Strongly Disagree

19. Is requirement prioritization in distributed scrum a challenging task.
    a. Strongly Agree
    b. Agree
    c. Neutral
    d. Disagree
    e. Strongly Disagree
20. Requirement prioritization plays a vital role in the development of the desired product. Have your team ever faced requirement prioritization issues in a distributed scrum environment
    a. Strongly Agree
    b. Agree
    c. Neutral
    d. Disagree
    e. Strongly Disagree

Appendix B: Respondents responses

User	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16	Q17	Q18	Q19
U1	Growth Hacker	0-3 years	16-25	Product Owner	Agree	Strongly Agree	Agree	Disagree	Agree	Agree	Agree	Neutral	Yes	Strongly Agree	Strongly Agree	Agree	Agree	Neutral	Sometime
U2	Co-Founder	0-3 years	Less than 15	Product Owner; Client; Scrum Master; Development Team	Strongly Agree	Agree	Strongly Agree	Agree	Agree	Strongly Agree	Agree	Disagree	Yes	Neutral	Neutral	Agree	Neutral	Disagree	Seldom
U3	Developer	0-3 years	Less than 15	Product Owner; Client; Scrum Master	Agree	Disagree	Neutral	Agree	Agree	Disagree	Neutral	Strongly Agree	Yes	Disagree	Agree	Strongly Agree	Strongly Agree	Disagree	Sometime

U4	Project Manager	4-7 years	Less than 15	All of these	Agree	Neutral	Strongly Agree	Agree	Agree	Agree	Agree	Agree	Yes	Agree	Agree	Strongly Agree	Agree	Agree	Always
U5	Project Manager	4-7 years	Less than 15	Client	Strongly Agree	Agree	Agree	Disagree	Strongly Agree	Agree	Neutral	Disagree	Yes	Agree	Agree	Strongly Agree	Agree	Neutral	Seldom
U6	Product Owner	0-3 years	Less than 15	Product Owner; Client	Strongly Agree	Agree	Disagree	Neutral	Neutral	Agree	Strongly Agree	Strongly Agree	Yes	Disagree	Disagree	Agree	Disagree	Disagree	Always
U7	Product Owner	0-3 years	Less than 15	Client	Strongly Agree	Strongly Agree	Strongly Agree	Agree	Agree	Neutral	Strongly Agree	Strongly Agree	Yes	Agree	Agree	Strongly Agree	Strongly Agree	Strongly Agree	Always
U8	Project Manager	0-3 years	Less than 15	All of these	Strongly Agree	Neutral	Strongly Agree	Agree	Agree	Agree	Agree	Neutral	Yes	Disagree	Neutral	Agree	Agree	Agree	Sometime
U9	Project Manager	0-3 years	Less than 15	Client	Strongly Agree	Strongly Disagree	Strongly Agree	Strongly Agree	Agree	Agree	Strongly Agree	Agree	Yes	Strongly Agree	Agree	Strongly Agree	Disagree	Agree	Frequently
U10	Developer	0-3 years	Less than 15	Product Owner; Client	Strongly Agree	Agree	Agree	Neutral	Strongly Agree	Strongly Agree	Agree	Neutral	Yes	Strongly Disagree	Strongly Agree	Strongly Agree	Strongly Agree	Agree	Frequently
U11	Developer	0-3 years	16-25	Product Owner; Client; Scrum Master	Strongly Agree	Agree	Agree	Strongly Agree	Agree	Strongly Agree	Agree	Agree	Yes	Agree	Agree	Neutral	Neutral	Agree	Frequently
U12	System Engineer	4-7 years	More Than	Product Owner; Client; Development Team	Neutral	Agree	Strongly Agree	Agree	Strongly Agree	Neutral	Agree	Strongly Agree	Yes	Disagree	Disagree	Strongly Agree	Agree	Strongly Agree	Sometime

			35																
U13	Project Manager	4-7 years	Less than 15	Client; Development Team	Strongly Agree	Disagree	Strongly Agree	Neutral	Agree	Agree	Agree	Neutral	No	Agree	Strongly Agree	Strongly Agree	Strongly Agree	Agree	Sometime
U14	HR Officer	0-3 years	16-25	Product Owner; Client	Strongly Agree	Agree	Agree	Neutral	Agree	Agree	Agree	Agree	No	Agree	Agree	Strongly Agree	Agree	Agree	Seldom
U15	Developer	0-3 years	Less than 15	Client; Development Team	Strongly Agree	Agree	Agree	Agree	Strongly Agree	Agree	Agree	Neutral	Yes	Agree	Agree	Agree	Agree	Agree	Frequently
U16	Developer	0-3 years	Less than 15	Client; Development Team	Disagree	Agree	Agree	Disagree	Agree	Agree	Agree	Agree	Yes	Disagree	Neutral	Agree	Agree	Agree	Sometime
U17	Developer	4-7 years	More Than 35	Product Owner	Agree	Agree	Agree	Agree	Agree	Agree	Agree	Agree	Yes	Strongly Disagree	Strongly Disagree	Agree	Agree	Disagree	Never
U18	Developer	4-7 years	Less than 15	Client; Scrum Master; Development Team	Agree	Disagree	Strongly Agree	Neutral	Agree	Agree	Agree	Neutral	Yes	Disagree	Agree	Strongly Agree	Agree	Disagree	Sometime
U19	Developer	0-3 years	Less than 15	Development Team	Agree	Neutral	Neutral	Neutral	Neutral	Neutral	Agree	Strongly Agree	No	Agree	Agree	Strongly Agree	Agree	Neutral	Always
U20	Talent Acquisition Specialist	0-3 years	More Than 35	All of these	Strongly Agree	Strongly Agree	Strongly Agree	Strongly Agree	Strongly Agree	Strongly Agree	Strongly Agree	Strongly Agree	Yes	Strongly Agree	Strongly Agree	Strongly Agree	Strongly Agree	Strongly Agree	Always

U21	Developer	0-3 years	Less than 15	All of these	Neutral	Disagree	Agree	Agree	Strongly Agree	Strongly Agree	Agree	Agree	No	Neutral	Neutral	Agree	Agree	Agree	Frequently
U22	Data annotator	0-3 years	Less than 15	Product Owner	Agree	Agree	Agree	Neutral	Agree	Agree	Agree	Neutral	Yes	Agree	Agree	Agree	Agree	Neutral	Sometime
U23	SQA Engineer	4-7 years	16-25	Development Team	Strongly Agree	Agree	Agree	Neutral	Agree	Strongly Agree	Agree	Agree	Yes	Agree	Strongly Agree	Strongly Agree	Strongly Agree	Agree	Always
U24	Developer	0-3 years	Less than 15	Development Team	Agree	Neutral	Agree	Neutral	Agree	Agree	Disagree	Neutral	Yes	Disagree	Agree	Agree	Agree	Agree	Sometime
U25	Project Manager	0-3 years	Less than 15	Product Owner; Client	Neutral	Strongly Agree	Agree	Agree	Agree	Strongly Agree	Neutral	Agree	Yes	Agree	Agree	Strongly Agree	Agree	Disagree	Sometime
U26	Developer	0-3 years	Less than 15	Product Owner; Client	Strongly Agree	Strongly Agree	Strongly Agree	Disagree	Agree	Neutral	Agree	Strongly Agree	Yes	Agree	Agree	Neutral	Strongly Agree	Agree	Always
U27	Program Management	More than 10 years	25-35	Product Owner; Development Team	Agree	Agree	Strongly Agree	Disagree	Agree	Agree	Strongly Agree	Agree	Yes	Agree	Strongly Agree	Strongly Agree	Strongly Agree	Neutral	Sometime
U28	Scrum Master	0-3 years	16-25	Product Owner; Development Team	Neutral	Neutral	Strongly Agree	Neutral	Strongly Agree	Agree	Agree	Neutral	Yes	Agree	Agree	Strongly Agree	Strongly Agree	Disagree	Seldom
U29	Developer	4-7 years	16-25	Product Owner; Client; Develo	Disagree	Disagree	Strongly Agree	Agree	Strongly Agree	Strongly Agree	Strongly	Agree	Yes	Disagree	Disagree	Agree	Agree	Agree	Sometime

				pment Team							Agree								
U30	Scrum Master	0-3 years	Less than 15	Scrum Master	Neutral	Agree	Agree	Strongly Agree	Strongly Agree	Strongly Agree	Strongly Agree	Agree	Yes	Strongly Agree	Strongly Agree	Agree	Agree	Disagree	Sometime
U31	Scrum Master	8-10 years	More Than 35	Product Owner; Client; Development Team	Agree	Strongly Agree	Neutral	Strongly Agree	Neutral	Agree	Neutral	Disagree	Yes	Agree	Neutral	Agree	Neutral	Disagree	Always
U32	Scrum Master	4-7 years	Less than 15	Product Owner; Scrum Master; Development Team	Strongly Agree	Disagree	Agree	Strongly Agree	Strongly Agree	Strongly Agree	Strongly Agree	Strongly Agree	Yes	Strongly Agree	Agree	Strongly Agree	Strongly Agree	Agree	Frequently
U33	Delivery Manager	8-10 years	25-35	Product Owner; Client	Neutral	Neutral	Disagree	Disagree	Disagree	Disagree	Disagree	Disagree	Yes	Agree	Agree	Neutral	Agree	Neutral	Frequently
U34	Scrum Master	0-3 years	More Than 35	Product Owner; Development Team	Agree	Agree	Agree	Strongly Agree	Agree	Strongly Agree	Agree	Neutral	Yes	Neutral	Neutral	Agree	Agree	Strongly Agree	Frequently
U35	Product Owner	0-3 years	Less than 15	All of these	Strongly Agree	Strongly Agree	Strongly Agree	Neutral	Strongly Agree	Agree	Strongly Agree	Agree	Yes	Agree	Neutral	Strongly Agree	Agree	Disagree	Sometime
U36	Developer	0-3 years	Less than 15	Product Owner; Client; Scrum Master; Development Team; All of these	Agree	Neutral	Agree	Agree	Agree	Agree	Strongly Agree	Agree	Yes	Strongly Agree	Agree	Strongly Agree	Strongly Agree	Strongly Agree	Always
U37	Scrum Master	4-7 years	Less than 15	Product Owner; Development Team	Disagree	Strongly Agree	Strongly Agree	Strongly Disagree	Strongly Agree	Strongly Agree	Agree	Agree	Yes	Neutral	Neutral	Agree	Agree	Disagree	Sometime
U38	Scrum Master	4-7 years	Less	Product Owner; Client;	Disagree	Agree	Strongly	Strongly	Agree	Strongly	Agree	Strongly	Yes	Strongly	Strongly	Agree	Stron	Strongly	Always

			than 15	Scrum Master			Agree	Agree		Agree		Agree		Agree	Agree		gly Agree	Agree	
U39	Developer	0-3 years	Less than 15	All of these	Strongly Agree	Agree	Agree	Strongly Agree	Agree	Neutral	Agree	Strongly Agree	Yes	Strongly Agree	Neutral	Strongly Agree	Agree	Agree	Never
U40	Developer	0-3 years	More Than 35	Product Owner; Development Team	Strongly Agree	Neutral	Strongly Agree	Agree	Disagree	Disagree	Neutral	Agree	Yes	Agree	Strongly Agree	Strongly Agree	Stronngly Agree	Agree	Frequently
U41	Head of Product	0-3 years	16-25	Product Owner; Development Team	Disagree	Agree	Agree	Disagree	Strongly Agree	Agree	Agree	Agree	Yes	Agree	Disagree	Strongly Agree	Agree	Agree	Frequently
U42	Product Owner	8-10 years	Less than 15	Product Owner; Client; Scrum Master; Development Team; All of these	Agree	Agree	Strongly Agree	Neutral	Agree	Agree	Neutral	Neutral	Yes	Agree	Neutral	Agree	Neutral	Neutral	Sometime
U43	Scrum Master	0-3 years	Less than 15	Product Owner	Disagree	Agree	Agree	Disagree	Agree	Agree	Agree	Disagree	Yes	Disagree	Neutral	Agree	Disagree	Disagree	Sometime
U44	Scrum Master	4-7 years	25-35	Product Owner	Strongly Agree	Neutral	Agree	Strongly Agree	Neutral	Neutral	Agree	Agree	Yes	Disagree	Disagree	Agree	Agree	Agree	Frequently
U45	Scrum Master	0-3 years	25-35	Product Owner; Client; Scrum Master	Neutral	Disagree	Agree	Neutral	Agree	Agree	Agree	Agree	Yes	Agree	Agree	Agree	Agree	Neutral	Sometime
U46	Scrum Master	0-3 years	Less than 15	Product Owner; Client; Scrum Master	Strongly Agree	Strongly Agree	Strongly Agree	Disagree	Strongly Agree	Strongly Agree	Strongly Agree	Strongly Agree	Yes	Strongly Agree	Strongly Agree	Strongly Agree	Stronngly Agree	Strongly Agree	Frequently
U47	Scrum Master	0-3 years	Less th	Product Owner; Scrum Master;	Strongly	Agree	Strongly	Strongly	Neutral	Neutral	Disagree	Disagree	No	Disagree	Disagree	Agree	Disa	Neutral	Seldom

			an 15	Development Team	Agree		Agree	Agree									gree		
U48	Project Manager	4-7 years	16-25	Product Owner	Agree	Strongly Agree	Agree	Neutral	Agree	Agree	Agree	Agree	Yes	Agree	Agree	Agree	Agree	Disagree	Seldom
U49	Product Owner	0-3 years	Less than 15	All of these	Strongly Agree	Agree	Neutral	Neutral	Strongly Agree	Agree	Strongly Agree	Agree	Yes	Agree	Agree	Agree	Neutral	Strongly Agree	Sometime
U50	Project Manager	0-3 years	16-25	Product Owner	Disagree	Agree	Agree	Disagree	Disagree	Strongly Agree	Strongly Agree	Agree	Yes	Neutral	Neutral	Agree	Strongly Agree	Agree	Frequently
U51	Scrum Master	0-3 years	Less than 15	Product Owner; Scrum Master; Development Team	Strongly Agree	Strongly Agree	Agree	Agree	Strongly Agree	Strongly Agree	Strongly Agree	Agree	Yes	Disagree	Neutral	Disagree	Agree	Agree	Frequently
U52	Scrum Master	0-3 years	Less than 15	Product Owner; Scrum Master; Development Team	Neutral	Agree	Neutral	Disagree	Agree	Agree	Neutral	Neutral	Yes	Agree	Agree	Agree	Strongly Agree	Agree	Frequently
U53	Business Quality Analyst	0-3 years	16-25	Product Owner; Scrum Master; Development Team	Agree	Agree	Agree	Neutral	Agree	Agree	Agree	Neutral	Yes	Disagree	Strongly Disagree	Disagree	Agree	Disagree	Seldom
U54	Project Manager	0-3 years	16-25	Product Owner; Client; Scrum Master	Strongly Agree	Disagree	Agree	Agree	Agree	Agree	Agree	Agree	Yes	Disagree	Neutral	Agree	Agree	Agree	Always
U55	SQA Engineer	0-3 years	16-25	All of these	Agree	Disagree	Agree	Agree	Agree	Agree	Strongly Agree	Agree	Yes	Agree	Agree	Strongly Agree	Strongly Agree	Neutral	Always
U56	Scrum Master	0-3 years	16-25	Product Owner; Scrum Master	Agree	Agree	Agree	Agree	Disagree	Agree	Agree	Agree	No	Agree	Agree	Agree	Agree	Agree	Frequently
U57	Developer	0-3 years	16-25	Product Owner; Client	Agree	Strongly	Strongly	Agree	Agree	Agree	Strongly	Strongly Agree	Yes	Neutral	Agree	Strongly	Strongl	Agree	Sometime

						Agree	Agree				Agree					Agree	y Agree		
U58	Sr. QA Engineer	4-7 years	More Than 35	All of these	Strongly Agree	Strongly Agree	Strongly Agree	Agree	Agree	Agree	Agree	Neutral	Yes	Agree	Agree	Strongly Agree	Agree	Agree	Frequently
U59	Developer	0-3 years	16-25	Development Team	Strongly Disagree	Strongly Agree	Disagree	Agree	Neutral	Neutral	Disagree	Agree	Yes	Disagree	Agree	Neutral	Neutral	Strongly Agree	Sometime
U60	QA With product owner certification	0-3 years	Less than 15	Product Owner; Client; Scrum Master; Development Team; All of these	Disagree	Agree	Strongly Agree	Disagree	Agree	Agree	Agree	Agree	Yes	Neutral	Agree	Agree	Agree	Agree	Sometime
U61	Scrum Master	4-7 years	Less than 15	Product Owner	Strongly Agree	Strongly Agree	Agree	Disagree	Disagree	Strongly Agree	Agree	Disagree	Yes	Disagree	Disagree	Agree	Neutral	Neutral	Sometime
U62	Software Quality Engineer	0-3 years	Less than 15	Scrum Master; Development Team	Agree	Neutral	Agree	Agree	Agree	Agree	Neutral	Strongly Agree	Yes	Agree	Agree	Neutral	Agree	Neutral	Always
U63	Scrum Master	0-3 years	Less than 15	Product Owner	Agree	Strongly Agree	Agree	Agree	Agree	Agree	Agree	Neutral	Yes	Neutral	Neutral	Agree	Agree	Neutral	Sometime
U64	Scrum Master	0-3 years	16-25	All of these	Agree	Neutral	Strongly Agree	Disagree	Strongly Agree	Agree	Agree	Agree	No	Strongly Disagree	Strongly Agree	Neutral	Agree	Neutral	Always
U65	Project Manager	4-7 years	Less than 15	All of these	Neutral	Agree	Agree	Agree	Agree	Disagree	Agree	Neutral	Yes	Disagree	Neutral	Agree	Disagree	Agree	Sometime
U66	Scrum Master	8-10	16-	Product Owner	Disagree	Strongly	Agree	Agree	Agree	Agree	Agree	Neutral	Yes	Neutral	Disa	Disa	Agree	Disagree	Some

		years	25			Agree									gree	gree			time
U67	Scrum Master	4-7 years	Less than 15	All of these	Agree	Strongly Agree	Strongly Agree	Agree	Agree	Agree	Agree	Agree	Yes	Disagree	Disagree	Agree	Agree	Agree	Sometime
U68	QA	0-3 years	Less than 15	All of these	Agree	Agree	Agree	Neutral	Agree	Agree	Agree	Disagree	Yes	Disagree	Disagree	Disagree	Agree	Disagree	Seldom
U69	Project Manager	0-3 years	More Than 35	Product Owner; Client; Scrum Master	Strongly Agree	Neutral	Strongly Agree	Neutral	Agree	Agree	Agree	Agree	No	Strongly Agree	Strongly Agree	Strongly Agree	Strongly Agree	Strongly Agree	Sometime
U70	Scrum Master	0-3 years	More Than 35	Product Owner; Client	Agree	Agree	Agree	Agree	Strongly Agree	Strongly Agree	Agree	Agree	Yes	Agree	Agree	Neutral	Agree	Disagree	Sometime
U71	QA lead	4-7 years	16-25	Product Owner; Client	Neutral	Strongly Agree	Agree	Neutral	Agree	Agree	Agree	Strongly Disagree	Yes	Strongly Disagree	Agree	Agree	Agree	Neutral	Frequently
U72	Scrum Master	4-7 years	16-25	Product Owner	Agree	Agree	Strongly Agree	Disagree	Neutral	Strongly Agree	Strongly Disagree	Agree	Yes	Disagree	Agree	Agree	Disagree	Neutral	Sometime
U73	QA engineer	0-3 years	16-25	All of these	Neutral	Disagree	Agree	Disagree	Agree	Agree	Disagree	Strongly Agree	Yes	Agree	Agree	Agree	Agree	Agree	Seldom
U74	Recruitment consultant	0-3 years	Less than 15	Product Owner; Client; Scrum Master; Development Team; All of these	Agree	Disagree	Agree	Neutral	Agree	Agree	Agree	Disagree	No	Agree	Agree	Agree	Agree	Strongly Agree	Always
U75	Accountant	0-3 years	Less than	All of these	Agree	Strongly Agree	Agree	Agree	Agree	Agree	Agree	Agree	No	Agree	Agree	Agree	Agree	Agree	Sometime

			15																
U76	Scrum Master	4-7 years	Less than 15	Product Owner; Development Team	Neutral	Agree	Agree	Agree	Strongly Agree	Strongly Agree	Strongly Agree	Agree	Yes	Disagree	Disagree	Neutral	Agree	Disagree	Sometime
U77	Scrum Master	0-3 years	Less than 15	Development Team	Neutral	Neutral	Agree	Neutral	Agree	Agree	Agree	Neutral	Yes	Disagree	Neutral	Neutral	Agree	Neutral	Sometime
U78	Developer	0-3 years	16-25	Scrum Master	Neutral	Agree	Agree	Neutral	Disagree	Agree	Disagree	Strongly Agree	No	Agree	Neutral	Strongly Agree	Agree	Neutral	Frequently
U79	Developer	4-7 years	16-25	Product Owner; Client; Development Team	Agree	Disagree	Strongly Agree	Disagree	Strongly Agree	Strongly Agree	Neutral	Strongly Agree	Yes	Neutral	Neutral	Agree	Neutral	Agree	Sometime
U80	Scrum Master	0-3 years	Less than 15	Product Owner	Disagree	Agree	Neutral	Neutral	Agree	Agree	Agree	Agree	Yes	Neutral	Neutral	Agree	Neutral	Neutral	Sometime
U81	Project Manager	0-3 years	16-25	Product Owner; Client	Neutral	Strongly Agree	Neutral	Strongly Agree	Strongly Agree	Strongly Agree	Neutral	Agree	Yes	Strongly Agree	Strongly Agree	Strongly Agree	Strongly Agree	Neutral	Sometime
U82	Developer	0-3 years	Less than 15	Development Team	Agree	Agree	Agree	Agree	Agree	Agree	Agree	Neutral	Yes	Agree	Agree	Neutral	Agree	Neutral	Always
U83	Project Manager	0-3 years	16-25	Product Owner; Client	Strongly Agree	Strongly Agree	Strongly Agree	Agree	Strongly Agree	Strongly Agree	Agree	Neutral	Yes	Agree	Agree	Strongly Agree	Strongly Agree	Agree	Frequently
U84	Project Manager	0-3 years	Less than	All of these	Strongly Agree	Strongly Agree	Strongly Agree	Neutral	Strongly Agree	Strongly Agree	Strongly Agree	Agree	Yes	Agree	Agree	Strongly A	Strongly A	Agree	Always

			15													gree	gree		
U85	Scrum Master	4-7 years	Less than 15	Product Owner; Client	Agree	Strongly Agree	Agree	Neutral	Agree	Agree	Agree	Agree	Yes	Disagree	Neutral	Neutral	Strongly Agree	Disagree	Frequently
U86	Developer	0-3 years	More Than 35	Client; Development Team	Strongly Agree	Agree	Strongly Agree	Strongly Agree	Agree	Agree	Strongly Agree	Strongly Agree	No	Strongly Disagree	Neutral	Agree	Strongly Agree	Neutral	Sometime
U87	Project Manager	0-3 years	Less than 15	Product Owner; Client	Agree	Agree	Neutral	Neutral	Agree	Agree	Strongly Agree	Agree	Yes	Agree	Neutral	Neutral	Neutral	Disagree	Sometime
U88	Product Owner	0-3 years	Less than 15	Product Owner	Disagree	Strongly Agree	Agree	Agree	Agree	Strongly Agree	Agree	Neutral	Yes	Disagree	Disagree	Neutral	Disagree	Agree	Sometime
U89	Project Manager	4-7 years	16-25	Product Owner	Neutral	Agree	Agree	Neutral	Agree	Strongly Agree	Strongly Agree	Agree	Yes	Agree	Neutral	Agree	Agree	Neutral	Sometime
U90	Project Manager	More than 10 years	16-25	Product Owner	Agree	Agree	Agree	Agree	Agree	Agree	Sometimes	Disagree	Yes	Agree	Agree	Agree	Agree	Agree	Frequently
U91	SQA Engineer	0-3 years	16-25	Client; Scrum Master; Development Team	Agree	Agree	Agree	Agree	Agree	Agree	Agree	Strongly Agree	Yes	Neutral	Neutral	Neutral	Strongly Agree	Agree	Always
U92	Scrum Master	0-3 years	Less than 15	Product Owner; Development Team	Agree	Agree	Neutral	Neutral	Disagree	Strongly Agree	Agree	Disagree	Yes	Agree	Neutral	Neutral	Neutral	Disagree	Frequently
U93	Project Manager	0-3 years	Less tha	Product Owner; Client; Develo	Disagree	Disagree	Agree	Neutral	Agree	Agree	Agree	Neutral	Yes	Disagree	Disagree	Disagree	Agree	Disagree	Seldom

			n 15	pment Team																
U94	Scrum Master	More than 10 years	Less than 15	Product Owner; Client; Scrum Master; Development Team; All of these	Strongly Agree	Neutral	Agree	Agree	Strongly Agree	Agree	Agree	Agree	Yes	Neutral	Disagree	Neutral	Agree	Neutral	Seldom
U95	Tech Lead	0-3 years	Less than 15	Product Owner; Client; Development Team	Strongly Agree	Strongly Agree	Strongly Agree	Agree	Strongly Agree	Strongly Agree	Agree	Neutral	Yes	Neutral	Neutral	Agree	Agree	Agree	Seldom
U96	Project Manager	0-3 years	16-25	Product Owner; Client; Development Team	Strongly Agree	Agree	Agree	Disagree	Strongly Agree	Agree	Strongly Agree	Strongly Agree	Yes	Agree	Agree	Strongly Agree	Strongly Agree	Agree	Sometime
U97	Scrum Master	0-3 years	16-25	Product Owner; Client	Agree	Strongly Agree	Agree	Disagree	Agree	Agree	Agree	Disagree	Yes	Strongly Agree	Neutral	Agree	Agree	Neutral	Never
U98	Project Manager	4-7 years	More Than 35	Product Owner; Scrum Master	Agree	Agree	Agree	Agree	Strongly Agree	Strongly Agree	Agree	Agree	Yes	Disagree	Neutral	Agree	Agree	Agree	Sometime
U99	SQA Engineer	4-7 years	25-35	Product Owner; Client	Strongly Agree	Strongly Agree	Strongly Agree	Agree	Strongly Agree	Strongly Agree	Strongly Agree	Strongly Agree	Yes	Agree	Agree	Strongly Agree	Strongly Agree	Agree	Sometime
U100	Project Manager	4-7 years	16-25	All of these	Strongly Agree	Agree	Strongly Agree	Agree	Agree	Neutral	Strongly Agree	Agree	Yes	Agree	Agree	Agree	Agree	Neutral	Sometime
U101	QA Engineer	4-7 years	Less than 15	All of these	Agree	Neutral	Strongly Agree	Neutral	Agree	Agree	Agree	Disagree	Yes	Neutral	Neutral	Agree	Neutral	Disagree	Seldom
U102	Developer	0-3 years	Less th	Product Owner	Agree	Agree	Neutral	Agree	Agree	Agree	Agree	Agree	Yes	Agree	Agree	Agree	Disa	Neutral	Frequently

			an 15														gree		
U103	Developer	0-3 years	Less than 15	Development Team	Neutral	Agree	Agree	Agree	Agree	Strongly Agree	Agree	Agree	Yes	Agree	Agree	Strongly Agree	Agree	Agree	Always
U104	Internee	4-7 years	More Than 35	Scrum Master	Agree	Agree	Agree	Agree	Agree	Neutral	Neutral	Neutral	Yes	Agree	Agree	Agree	Agree	Agree	Frequently
U105	Developer	0-3 years	16-25	Product Owner; Development Team	Agree	Agree	Agree	Agree	Agree	Agree	Neutral	Agree	Yes	Agree	Strongly Agree	Agree	Strongly Agree	Disagree	Frequently
U106	Developer	0-3 years	Less than 15	All of these	Agree	Agree	Agree	Agree	Agree	Agree	Agree	Agree	No	Neutral	Agree	Agree	Agree	Agree	Always
U107	Developer	0-3 years	Less than 15	Product Owner; Client; Scrum Master; Development Team; All of these	Strongly Agree	Disagree	Disagree	Neutral	Neutral	Agree	Agree	Agree	No	Agree	Agree	Agree	Strongly Agree	Agree	Sometime
U108	Developer	0-3 years	Less than 15	Development Team	Agree	Disagree	Agree	Agree	Agree	Strongly Agree	Strongly Agree	Strongly Agree	No	Strongly Agree	Strongly Agree	Strongly Agree	Agree	Strongly Agree	Always
U109	Developer	0-3 years	Less than 15	All of these	Agree	Agree	Neutral	Neutral	Strongly Agree	Disagree	Agree	Neutral	Yes	Disagree	Neutral	Agree	Agree	Neutral	Always
U110	Developer	0-3 years	25-35	Development Team	Agree	Agree	Agree	Agree	Agree	Agree	Agree	Agree	Yes	Agree	Agree	Agree	Agree	Agree	Sometime

U111	Developer	0-3 years	Less than 15	All of these	Strongly Agree	Strongly Agree	Agree	Strongly Disagree	Agree	Agree	Strongly Agree	Agree	Yes	Agree	Agree	Agree	Agree	Agree	Always
U112	Project Manager	0-3 years	Less than 15	Client	Strongly Agree	Strongly Agree	Strongly Agree	Neutral	Agree	Strongly Agree	Strongly Agree	Strongly Agree	Yes	Strongly Agree	Strongly Agree	Agree	Strongly Agree	Agree	Frequently
U113	Developer	0-3 years	Less than 15	Development Team	Agree	Agree	Agree	Agree	Agree	Agree	Neutral	Disagree	Yes	Agree	Neutral	Strongly Agree	Agree	Agree	Sometime
U114	internee	0-3 years	16-25	All of these	Agree	Neutral	Agree	Agree	Neutral	Agree	Agree	Neutral	Yes	Agree	Neutral	Agree	Agree	Agree	Sometime
U115	Developer	0-3 years	Less than 15	Product Owner; Client; Development Team	Agree	Strongly Agree	Strongly Agree	Agree	Strongly Agree	Neutral	Neutral	Agree	Yes	Agree	Neutral	Strongly Agree	Agree	Neutral	Never
U116	Project Manager	0-3 years	16-25	Product Owner; Client	Agree	Agree	Neutral	Agree	Strongly Agree	Strongly Agree	Neutral	Neutral	No	Agree	Agree	Strongly Agree	Strongly Agree	Agree	Frequently
U117	Digital Marketer	0-3 years	Less than 15	Scrum Master	Agree	Disagree	Strongly Agree	Agree	Agree	Strongly Agree	Agree	Strongly Agree	No	Strongly Agree	Agree	Strongly Agree	Disagree	Strongly Disagree	Sometime
U118	Business Developer	0-3 years	Less than 15	Development Team	Agree	Agree	Agree	Neutral	Agree	Agree	Agree	Agree	Yes	Agree	Agree	Agree	Agree	Agree	Sometime
U119	UI/UX designer	0-3 years	16-25	Client; Development Team	Strongly Agree	Agree	Strongly Agree	Strongly Agree	Strongly Agree	Agree	Agree	Neutral	Yes	Strongly Agree	Agree	Strongly Agree	Disagree	Disagree	Sometime

U120	Developer	0-3 years	Less than 15	Development Team	Neutral	Neutral	Strongly Agree	Agree	Neutral	Agree	Strongly Agree	Neutral	Yes	Strongly Agree	Strongly Agree	Disagree	Strongly Agree	Strongly Agree	Frequently
U121	Developer	0-3 years	Less than 15	All of these	Strongly Agree	Neutral	Strongly Agree	Agree	Agree	Agree	Agree	Agree	Yes	Neutral	Neutral	Disagree	Neutral	Agree	Frequently
U122	SQA	0-3 years	Less than 15	Product Owner	Neutral	Agree	Strongly Agree	Disagree	Neutral	Agree	Disagree	Neutral	Yes	Agree	Neutral	Neutral	Neutral	Disagree	Always
U123	Developer	4-7 years	Less than 15	Scrum Master	Agree	Strongly Agree	Agree	Disagree	Agree	Agree	Agree	Disagree	Yes	Strongly Agree	Agree	Strongly Agree	Agree	Disagree	Frequently
U124	Delivery Operation Center Engineer	0-3 years	Less than 15	Product Owner; Client; Scrum Master; Development Team; All of these	Strongly Agree	Strongly Agree	Strongly Agree	Agree	Agree	Agree	Agree	Agree	Yes	Agree	Agree	Agree	Agree	Agree	Frequently
U125	Developer	0-3 years	Less than 15	Scrum Master; Development Team	Agree	Agree	Strongly Agree	Agree	Strongly Agree	Strongly Agree	Strongly Agree	Strongly Agree	Yes	Agree	Agree	Agree	Strongly Agree	Agree	Sometime
U126	Business manager	0-3 years	16-25	All of these	Agree	Agree	Strongly Agree	Agree	Strongly Agree	Strongly Agree	Agree	Agree	Yes	Agree	Neutral	Agree	Agree	Agree	Sometime
U127	SQA Manager	0-3 years	25-35	All of these	Agree	Strongly Agree	Agree	Agree	Strongly Agree	Strongly Agree	Neutral	Neutral	Yes	Strongly Agree	Disagree	Strongly Agree	Strongly Agree	Agree	Always
U128	Product Owner	8-10 years	16-25	Product Owner; Client; Development Team	Agree	Agree	Neutral	Agree	Strongly Agree	Strongly Agree	Neutral	Disagree	Yes	Strongly Disagree	Strongly Disa	Agree	Disagree	Agree	Seldom

															gree				
U129	Developer	4-7 years	More Than 35	Product Owner	Agree	Agree	Strongly Agree	Neutral	Agree	Agree	Agree	Agree	Yes	Strongly Agree	Strongly Agree	Strongly Agree	Agree	Agree	Always
U130	Developer	0-3 years	Less than 15	Product Owner; Client; Development Team	Agree	Agree	Agree	Agree	Agree	Agree	Agree	Agree	Yes	Strongly Agree	Agree	Strongly Agree	Strongly Agree	Disagree	Frequently
U131	Project Manager	4-7 years	Less than 15	Product Owner; Scrum Master	Disagree	Agree	Disagree	Strongly Agree	Agree	Agree	Disagree	Disagree	Yes	Disagree	Disagree	Agree	Agree	Disagree	Seldom
U132	Project Manager	0-3 years	Less than 15	Product Owner; Development Team	Disagree	Agree	Agree	Disagree	Agree	Strongly Agree	Strongly Agree	Neutral	Yes	Strongly Agree	Agree	Strongly Agree	Agree	Agree	Frequently
U133	Developer	8-10 years	Less than 15	All of these	Agree	Strongly Agree	Strongly Agree	Neutral	Strongly Agree	Strongly Agree	Agree	Strongly Agree	Yes	Agree	Agree	Agree	Neutral	Strongly Agree	Sometime
U134	Developer	0-3 years	Less than 15	Product Owner	Agree	Agree	Agree	Neutral	Agree	Agree	Strongly Agree	Neutral	Yes	Neutral	Neutral	Neutral	Neutral	Neutral	Sometime
U135	SQA Engineer	0-3 years	Less than 15	Scrum Master	Strongly Agree	Strongly Agree	Strongly Agree	Agree	Agree	Strongly Agree	Strongly Agree	Neutral	Yes	Neutral	Agree	Neutral	Strongly Agree	Strongly Agree	Always
U136	Scrum Master	4-7 years	16-25	Product Owner	Strongly Agree	Neutral	Agree	Agree	Neutral	Neutral	Neutral	Agree	Yes	Neutral	Neutral	Neutral	Strongly Agree	Agree	Frequently

U137	Scrum Master	0-3 years	Less than 15	Product Owner; Client	Disagree	Agree	Disagree	Agree	Agree	Strongly Agree	Disagree	Agree	Yes	Disagree	Neutral	Disagree	Agree	Disagree	Sometime
U138	Project Manager	0-3 years	Less than 15	Product Owner; Client; Scrum Master	Agree	Neutral	Strongly Agree	Disagree	Agree	Agree	Strongly Agree	Neutral	Yes	Neutral	Neutral	Strongly Agree	Agree	Neutral	Sometime
U139	Scrum Master	4-7 years	16-25	Product Owner	Disagree	Agree	Agree	Agree	Agree	Agree	Agree	Agree	Yes	Agree	Neutral	Agree	Agree	Disagree	Sometime
U140	Scrum Master	0-3 years	Less than 15	All of these	Strongly Agree	Neutral	Strongly Agree	Neutral	Neutral	Agree	Neutral	Neutral	Yes	Disagree	Neutral	Disagree	Agree	Disagree	Sometime
U141	Product Owner	0-3 years	25-35	Product Owner	Disagree	Agree	Agree	Strongly Agree	Strongly Agree	Strongly Agree	Disagree	Neutral	Yes	Agree	Disagree	Disagree	Strongly Agree	Agree	Sometime
U142	Developer	0-3 years	Less than 15	Product Owner; Client	Agree	Agree	Agree	Disagree	Agree	Agree	Strongly Agree	Strongly Agree	No	Agree	Agree	Strongly Agree	Strongly Agree	Strongly Agree	Always
U143	Scrum Master	4-7 years	Less than 15	Client; Scrum Master	Strongly Agree	Neutral	Strongly Agree	Agree	Strongly Agree	Strongly Agree	Neutral	Disagree	Yes	Neutral	Neutral	Agree	Agree	Strongly Agree	Sometime
U144	SQA, Scrum Master and PM	4-7 years	25-35	All of these	Strongly Agree	Strongly Agree	Strongly Agree	Agree	Strongly Agree	Strongly Agree	Strongly Agree	Disagree	Yes	Agree	Agree	Neutral	Strongly Agree	Agree	Sometime
U145	SQA engineer	4-7 years	Less than 15	All of these	Strongly Agree	Agree	Strongly Agree	Strongly Agree	Strongly Agree	Strongly Agree	Agree	Strongly Agree	Yes	Strongly Agree	Strongly Agree	Agree	Strongly Agree	Strongly Agree	Always

U146	Developer	0-3 years	Less than 15	Scrum Master	Neutral	Neutral	Agree	Neutral	Agree	Agree	Agree	Agree	Yes	Disagree	Disagree	Strongly Agree	Strongly Agree	Agree	Sometime
U147	Developer	0-3 years	Less than 15	All of these	Agree	Agree	Strongly Agree	Neutral	Agree	Strongly Agree	Neutral	Neutral	Yes	Neutral	Disagree	Disagree	Disagree	Disagree	Sometime
U148	Developer	0-3 years	16-25	Development Team	Agree	Agree	Agree	Agree	Agree	Strongly Agree	Neutral	Neutral	No	Agree	Neutral	Neutral	Neutral	Neutral	Frequently
U149	Scrum Master	0-3 years	Less than 15	Product Owner; Scrum Master; Development Team	Disagree	Disagree	Strongly Agree	Disagree	Agree	Agree	Agree	Agree	Yes	Disagree	Neutral	Neutral	Agree	Disagree	Seldom
U150	Developer	0-3 years	16-25	Development Team	Agree	Strongly Agree	Strongly Agree	Agree	Agree	Agree	Agree	Agree	No	Agree	Agree	Neutral	Agree	Agree	Frequently
U151	Project Manager/ Developer	0-3 years	16-25	Product Owner; Scrum Master; Development Team	Agree	Agree	Strongly Agree	Agree	Strongly Agree	Strongly Agree	Strongly Agree	Strongly Agree	Yes	Neutral	Agree	Agree	Agree	Agree	Sometime
U152	Project Manager	0-3 years	16-25	Product Owner; Client	Agree	Agree	Agree	Agree	Agree	Agree	Agree	Agree	Yes	Agree	Agree	Agree	Agree	Agree	Frequently
U153	Scrum Master	4-7 years	Less than 15	Product Owner	Agree	Strongly Agree	Strongly Agree	Agree	Agree	Agree	Agree	Neutral	Yes	Agree	Agree	Agree	Agree	Neutral	Seldom
U154	Scrum Master	4-7 years	25-35	Product Owner; Client; Scrum Master	Agree	Strongly Agree	Neutral	Agree	Strongly Agree	Strongly Agree	Disagree	Neutral	Yes	Strongly Disagree	Disagree	Agree	Agree	Neutral	Sometime
U155	Scrum Master	0-3 years	25-35	Product Owner	Agree	Strongly Agree	Agree	Neutral	Strongly Agree	Agree	Agree	Neutral	Yes	Disagree	Disagree	Strongly Agree	Agree	Neutral	Frequently
U156	Product Owner	8-10 years	Less th	Product Owner; Client	Neutral	Neutral	Agree	Disagree	Strongly	Agree	Agree	Neutral	Yes	Strongly Disagree	Disagree	Disagree	Disa	Disagree	Never

			an 15						Agree								gree		
U157	Scrum Master	0-3 years	16-25	Product Owner; Client	Disagree	Strongly Agree	Neutral	Disagree	Disagree	Strongly Agree	Strongly Agree	Disagree	Yes	Strongly Agree	Neutral	Strongly Agree	Strongly Agree	Strongly Agree	Always
U158	Scrum Master	4-7 years	Less than 15	Product Owner; Development Team	Agree	Strongly Agree	Agree	Agree	Agree	Agree	Agree	Disagree	Yes	Neutral	Neutral	Disagree	Neutral	Disagree	Seldom
U159	Project Manager	8-10 years	25-35	Product Owner	Disagree	Strongly Agree	Strongly Agree	Strongly Agree	Strongly Agree	Strongly Agree	Strongly Agree	Disagree	Yes	Disagree	Agree	Agree	Neutral	Neutral	Sometime
U160	Business Analyst	0-3 years	Less than 15	Product Owner	Agree	Strongly Agree	Agree	Neutral	Strongly Agree	Neutral	Agree	Disagree	Yes	Agree	Agree	Disagree	Agree	Agree	Sometime
U161	QA	4-7 years	Less than 15	All of these	Agree	Neutral	Agree	Agree	Strongly Agree	Strongly Agree	Agree	Agree	Yes	Agree	Neutral	Agree	Agree	Agree	Frequently
U162	QA	4-7 years	More Than 35	Product Owner	Agree	Neutral	Agree	Agree	Agree	Strongly Agree	Agree	Agree	Yes	Neutral	Neutral	Neutral	Agree	Neutral	Sometime
U163	QA	More than 10 years	Less than 15	Product Owner; Development Team	Strongly Disagree	Agree	Agree	Disagree	Strongly Agree	Strongly Agree	Strongly Agree	Strongly Agree	Yes	Disagree	Disagree	Disagree	Agree	Disagree	Seldom
U164	Developer	0-3 years	Less than 15	Product Owner; Client	Strongly Agree	Agree	Strongly Agree	Neutral	Agree	Neutral	Agree	Disagree	Yes	Disagree	Agree	Strongly Agree	Strongly Agree	Agree	Sometime

U165	Developer	8-10 years	Less than 15	Product Owner; Client; Development Team	Agree	Strongly Agree	Neutral	Agree	Strongly Agree	Agree	Neutral	Agree	Yes	Agree	Neutral	Agree	Strongly Agree	Agree	Sometime
U166	Scrum Master	4-7 years	More Than 35	Product Owner; Scrum Master	Agree	Neutral	Strongly Agree	Agree	Strongly Agree	Agree	Agree	Agree	Yes	Strongly Agree	Strongly Agree	Strongly Agree	Strongly Agree	Strongly Agree	Always
U167	Product Owner	0-3 years	Less than 15	All of these	Agree	Agree	Strongly Agree	Disagree	Strongly Agree	Strongly Agree	Strongly Agree	Strongly Agree	Yes	Disagree	Agree	Strongly Agree	Strongly Agree	Neutral	Sometime
U168	Project Manager	4-7 years	16-25	Product Owner; Scrum Master; Development Team	Neutral	Disagree	Agree	Agree	Strongly Agree	Strongly Agree	Strongly Agree	Agree	Yes	Strongly Disagree	Disagree	Agree	Strongly Agree	Agree	Sometime
U169	Project Manager	4-7 years	More Than 35	Client; Scrum Master; Development Team	Neutral	Disagree	Agree	Neutral	Strongly Agree	Strongly Agree	Agree	Strongly Disagree	Yes	Strongly Disagree	Strongly Disagree	Disagree	Neutral	Disagree	Sometime
U170	Business Analyst	4-7 years	Less than 15	Product Owner; Client; Development Team	Strongly Agree	Neutral	Strongly Agree	Strongly Agree	Strongly Agree	Strongly Agree	Strongly Agree	Neutral	Yes	Agree	Agree	Agree	Neutral	Neutral	Sometime
U171	Developer	4-7 years	Less than 15	All of these	Neutral	Agree	Agree	Strongly Agree	Neutral	Agree	Neutral	Agree	Yes	Disagree	Disagree	Strongly Agree	Agree	Agree	Sometime
U172	Scrum Master	4-7 years	More Than 35	Product Owner; Scrum Master; Development Team	Strongly Disagree	Strongly Agree	Strongly Agree	Disagree	Agree	Agree	Agree	Disagree	Yes	Strongly Agree	Agree	Strongly Agree	Disagree	Disagree	Seldom

U173	Project Manager	0-3 years	Less than 15	Product Owner; Scrum Master; Development Team	Neutral	Agree	Agree	Neutral	Agree	Strongly Agree	Agree	Agree	Yes	Neutral	Agree	Agree	Strongly Agree	Neutral	Sometime
U174	Software QA Engineer	4-7 years	Less than 15	All of these	Agree	Agree	Agree	Neutral	Agree	Strongly Agree	Agree	Agree	Yes	Agree	Agree	Strongly Agree	Strongly Agree	Agree	Frequently
U175	Product Owner	4-7 years	16-25	Product Owner; Scrum Master; Development Team	Agree	Agree	Agree	Neutral	Agree	Strongly Agree	Disagree	Neutral	Yes	Neutral	Strongly Agree	Agree	Agree	Neutral	Frequently
U176	Scrum Master	4-7 years	Less than 15	Product Owner; Client	Disagree	Strongly Agree	Neutral	Neutral	Agree	Agree	Agree	Neutral	Yes	Disagree	Disagree	Disagree	Agree	Disagree	Sometime
U177	Sr QA Engineer	4-7 years	More Than 35	Product Owner	Strongly Agree	Strongly Agree	Strongly Agree	Agree	Strongly Agree	Agree	Neutral	Strongly Agree	Yes	Strongly Agree	Strongly Agree	Strongly Agree	Strongly Agree	Strongly Agree	Always
U178	Scrum Master	4-7 years	Less than 15	Product Owner	Strongly Disagree	Strongly Agree	Neutral	Agree	Agree	Neutral	Agree	Disagree	Yes	Strongly Disagree	Strongly Disagree	Disagree	Strongly Disagree	Strongly Disagree	Sometime
U179	Head of Engineering	8-10 years	16-25	Product Owner; Client; Scrum Master	Agree	Disagree	Agree	Agree	Agree	Strongly Agree	Disagree	Disagree	Yes	Agree	Agree	Agree	Strongly Agree	Agree	Frequently
U180	SQA Engineer	4-7 years	16-25	All of these	Strongly Agree	Disagree	Neutral	Agree	Agree	Agree	Agree	Neutral	Yes	Neutral	Strongly Agree	Agree	Strongly Agree	Strongly Agree	Always
U1	QA	0-3 years	Less	Client; Scrum Master	Neutral	Strongly	Agree	Neutral	Strongly	Agree	Agree	Strongly	No	Neutral	Neutral	Disa	Agree	Strongly	Always

81			than 15			Agree			Agree			Disagree				gree		Disagree	
U182	Scrum Master	4-7 years	Less than 15	Product Owner; Scrum Master	Neutral	Strongly Agree	Disagree	Strongly Agree	Strongly Agree	Strongly Agree	Strongly Agree	Strongly Agree	Yes	Strongly Disagree	Strongly Disagree	Strongly Disagree	Strongly Disagree	Strongly Disagree	Never
U183	Project Manager	0-3 years	More Than 35	Product Owner	Strongly Agree	Strongly Agree	Agree	Neutral	Agree	Agree	Strongly Agree	Agree	Yes	Strongly Agree	Strongly Agree	Strongly Agree	Strongly Agree	Neutral	Sometime
U184	Developer	0-3 years	Less than 15	All of these	Strongly Agree	Strongly Agree	Agree	Agree	Agree	Agree	Agree	Agree	Yes	Agree	Agree	Agree	Agree	Neutral	Sometime
U185	Product Owner	0-3 years	16-25	All of these	Neutral	Agree	Agree	Neutral	Strongly Agree	Strongly Agree	Agree	Agree	Yes	Disagree	Neutral	Agree	Agree	Neutral	Seldom
U186	Director	4-7 years	Less than 15	Product Owner; Client	Neutral	Disagree	Agree	Agree	Agree	Agree	Agree	Neutral	Yes	Agree	Agree	Agree	Agree	Disagree	Frequently
U187	Scrum master for projects running on scrum & Project Manager role for other projects I am doing both	0-3 years	Less than 15	Product Owner	Disagree	Strongly Agree	Disagree	Agree	Disagree	Disagree	Agree	Disagree	Yes	Agree	Strongly Agree	Agree	Agree	Strongly Disagree	Never
U188	Project Manager	0-3 years	Less than	All of these	Agree	Agree	Strongly Agree	Neutral	Agree	Strongly Agree	Agree	Agree	Yes	Agree	Agree	Strongly A	Agree	Agree	Sometime

			15													gree			
U189	SQA Engineer	0-3 years	16-25	Product Owner; Client; Scrum Master	Agree	Agree	Agree	Neutral	Strongly Agree	Strongly Agree	Strongly Agree	Strongly Agree	Yes	Strongly Agree	Agree	Strongly Agree	Strongly Agree	Agree	Sometime
U190	Scrum Master	0-3 years	Less than 15	Product Owner	Agree	Agree	Agree	Agree	Strongly Agree	Strongly Agree	Strongly Agree	Strongly Agree	Yes	Disagree	Neutral	Strongly Agree	Strongly Agree	Agree	Sometime
U191	Project Manager	0-3 years	16-25	All of these	Strongly Disagree	Agree	Strongly Agree	Neutral	Agree	Agree	Strongly Agree	Strongly Agree	Yes	Strongly Agree	Agree	Strongly Agree	Strongly Agree	Agree	Sometime
U192	Developer	4-7 years	Less than 15	Product Owner	Agree	Agree	Neutral	Disagree	Strongly Agree	Neutral	Strongly Disagree	Agree	Yes	Agree	Disagree	Agree	Agree	Neutral	Frequently
U193	Project Manager	4-7 years	16-25	Development Team	Agree	Neutral	Strongly Agree	Disagree	Agree	Agree	Agree	Neutral	Yes	Disagree	Agree	Agree	Agree	Agree	Frequently
U194	Scrum Master	4-7 years	Less than 15	Product Owner; Client	Agree	Agree	Neutral	Strongly Agree	Agree	Strongly Agree	Agree	Disagree	Yes	Strongly Agree	Neutral	Disagree	Agree	Agree	Sometime
U195	Product Owner	4-7 years	Less than 15	Product Owner; Scrum Master	Strongly Agree	Disagree	Agree	Neutral	Strongly Disagree	Strongly Agree	Agree	Agree	No	Disagree	Neutral	Strongly Agree	Strongly Disagree	Agree	Seldom
U196	Product Owner	4-7 years	Less than 15	Development Team	Strongly Disagree	Agree	Neutral	Agree	Agree	Disagree	Strongly Disagree	Agree	No	Neutral	Agree	Agree	Agree	Disagree	Never
U197	Product Owner	More than	16-25	Client	Strongly Di	Strongly	Strongly Dis	Agree	Disagree	Agree	Strongly	Strongly Disagree	No	Strongly Agree	Strongly Di	Neutral	Agree	Agree	Sometime

		10 years			sagree	Agree	agree				Disagree				sagree					
U198	Scrum Master	8-10 years	16-25	All of these	Disagree	Agree	Strongly Agree	Disagree	Agree	Disagree	Agree	Disagree	No	Strongly Disagree	Agree	Agree	Disagree	Strongly Disagree	Sometime
U199	Scrum Master	8-10 years	Less than 15	Product Owner; Client	Neutral	Neutral	Agree	Agree	Strongly Agree	Disagree	Strongly Disagree	Agree	Yes	Agree	Neutral	Neutral	Disagree	Disagree	Frequently
U200	Scrum Master	4-7 years	Less than 15	Product Owner; Client	Agree	Agree	Agree	Neutral	Disagree	Disagree	Disagree	Agree	Yes	Agree	Neutral	Agree	Agree	Neutral	Frequently
U201	Scrum Master	4-7 years	More Than 35	Product Owner; Client; Scrum Master	Agree	Agree	Disagree	Strongly Agree	Disagree	Strongly Agree	Disagree	Strongly Agree	Yes	Disagree	Agree	Agree	Disagree	Disagree	Frequently
U202	Developer	0-3 years	Less than 15	All of these	Agree	Neutral	Strongly Agree	Strongly Disagree	Neutral	Disagree	Agree	Agree	Yes	Agree	Agree	Agree	Agree	Neutral	Sometime
U203	Product Owner	4-7 years	Less than 15	All of these	Disagree	Agree	Agree	Neutral	Agree	Neutral	Agree	Neutral	Yes	Agree	Agree	Agree	Agree	Neutral	Always
U204	Project Manager	8-10 years	16-25	Development Team	Strongly Agree	Strongly Agree	Agree	Agree	Agree	Disagree	Strongly Disagree	Agree	Yes	Disagree	Agree	Strongly Agree	Strongly Disagree	Agree	Sometime
U205	Scrum Master	8-10 years	25-35	Scrum Master	Agree	Agree	Disagree	Agree	Strongly Agree	Disagree	Neutral	Agree	Yes	Disagree	Neutral	Agree	Agree	Disagree	Seldom
U2	Project Manager	8-10	Less	Client	Agree	Agree	Neutral	Agree	Disagree	Agree	Neutral	Agree	No	Disagree	Disa	Agree	Agree	Disagree	Frequ

06		years	than 15												gree				ently
U207	Product Owner	4-7 years	Less than 15	Product Owner; Client; Development Team	Agree	Agree	Agree	Agree	Agree	Agree	Agree	Neutral	Yes	Agree	Agree	Strongly Agree	Agree	Strongly Agree	Sometime
U208	Project Manager	0-3 years	Less than 15	Product Owner; Client; Scrum Master	Disagree	Neutral	Neutral	Strongly Agree	Neutral	Agree	Agree	Disagree	No	Agree	Agree	Agree	Agree	Agree	Always
U209	Project Manager	0-3 years	Less than 15	Product Owner; Client; Scrum Master; Development Team	Agree	Agree	Agree	Agree	Agree	Neutral	Neutral	Agree	Yes	Strongly Agree	Strongly Agree	Strongly Agree	Neutral	Agree	Always
U210	Scrum Master	8-10 years	Less than 15	Product Owner; Client	Agree	Neutral	Agree	Agree	Disagree	Disagree	Strongly Disagree	Agree	No	Agree	Agree	Strongly Agree	Disagree	Disagree	Frequently
U211	Project Manager	4-7 years	More Than 35	All of these	Agree	Disagree	Agree	Agree	Disagree	Strongly Agree	Neutral	Disagree	No	Agree	Strongly Agree	Strongly Disagree	Agree	Neutral	Frequently
U212	Developer	4-7 years	More Than 35	Product Owner; Client	Neutral	Agree	Disagree	Agree	Disagree	Agree	Disagree	Agree	Yes	Disagree	Agree	Agree	Disagree	Disagree	Frequently
U213	Developer	8-10 years	More Than 35	Product Owner; Client; Scrum Master; Development Team	Agree	Disagree	Agree	Strongly Disagree	Strongly Agree	Strongly Disagree	Strongly Disagree	Agree	Yes	Neutral	Disagree	Agree	Agree	Disagree	Frequently
U214	Developer	0-3 years	More Th	Client	Disagree	Agree	Disagree	Strongly Agree	Neutral	Disagree	Agree	Disagree	Yes	Strongly Agree	Disagree	Disagree	Agree	Strongly Agree	Always

			an 35																
U215	Scrum Master	4-7 years	25-35	Client; Development Team	Disagree	Agree	Strongly Agree	Neutral	Agree	Disagree	Neutral	Agree	Yes	Disagree	Strongly Disagree	Strongly Agree	Agree	Neutral	Sometime
U216	Product Owner	4-7 years	Less than 15	All of these	Strongly Agree	Disagree	Agree	Neutral	Agree	Neutral	Agree	Strongly Agree	Yes	Strongly Agree	Strongly Agree	Strongly Agree	Strongly Agree	Agree	Always
U217	Product Owner	More than 10 years	More Than 35	Product Owner; Client	Strongly Disagree	Disagree	Agree	Strongly Agree	Strongly Agree	Agree	Agree	Agree	Yes	Disagree	Disagree	Agree	Strongly Agree	Agree	Frequently
U218	Project Manager	8-10 years	16-25	Scrum Master	Neutral	Disagree	Agree	Strongly Agree	Agree	Neutral	Disagree	Agree	No	Agree	Neutral	Disagree	Strongly Disagree	Agree	Seldom
U219	Project Manager	0-3 years	Less than 15	Product Owner; Client; Development Team	Neutral	Agree	Neutral	Agree	Neutral	Agree	Agree	Agree	Yes	Agree	Agree	Neutral	Agree	Neutral	Frequently
U220	Scrum Master	0-3 years	Less than 15	Product Owner; Client	Agree	Agree	Agree	Agree	Agree	Neutral	Neutral	Agree	Yes	Strongly Agree	Strongly Agree	Strongly Agree	Strongly Agree	Strongly Agree	Sometime
U221	Product Owner	0-3 years	Less than 15	Product Owner; Client; Scrum Master	Agree	Agree	Agree	Agree	Agree	Agree	Neutral	Agree	Yes	Agree	Agree	Agree	Agree	Agree	Always
U222	Product Owner	4-7 years	Less than	Product Owner	Agree	Agree	Agree	Agree	Agree	Agree	Neutral	Neutral	Yes	Agree	Agree	Agree	Neutral	Agree	Frequently

			15																
U223	Product Owner	0-3 years	Less than 15	Product Owner; Client	Agree	Agree	Neutral	Strongly Agree	Neutral	Agree	Disagree	Disagree	Yes	Agree	Agree	Agree	Agree	Agree	Always
U224	Project Manager	0-3 years	Less than 15	Client; Scrum Master	Agree	Neutral	Agree	Strongly Agree	Agree	Neutral	Agree	Agree	Yes	Agree	Agree	Agree	Agree	Agree	Sometime
U225	Project Manager	4-7 years	25-35	Client	Neutral	Disagree	Strongly Agree	Strongly Disagree	Disagree	Disagree	Disagree	Disagree	Yes	Disagree	Disagree	Neutral	Agree	Strongly Disagree	Sometime
U226	Product Owner	8-10 years	25-35	Client	Disagree	Agree	Agree	Disagree	Disagree	Strongly Disagree	Strongly Disagree	Strongly Disagree	No	Neutral	Disagree	Disagree	Agree	Disagree	Seldom
U227	Project Manager	0-3 years	Less than 15	All of these	Disagree	Agree	Neutral	Agree	Agree	Agree	Neutral	Strongly Agree	No	Neutral	Neutral	Agree	Agree	Agree	Sometime
U228	Project Manager	8-10 years	16-25	Client; Scrum Master; Development Team	Disagree	Disagree	Disagree	Disagree	Disagree	Disagree	Strongly Disagree	Disagree	Yes	Disagree	Disagree	Neutral	Agree	Strongly Disagree	Seldom
U229	Scrum Master	8-10 years	16-25	Client	Neutral	Disagree	Disagree	Strongly Disagree	Neutral	Disagree	Strongly Disagree	Neutral	Yes	Strongly Disagree	Disagree	Strongly Disagree	Strongly Disagree	Disagree	Sometime
U230	Project Manager	4-7 years	Less than 15	Product Owner; Client; Scrum Master	Disagree	Neutral	Agree	Neutral	Neutral	Agree	Neutral	Agree	Yes	Agree	Agree	Agree	Agree	Agree	Frequently
U2	Developer	4-7 years	16-	Client; Scrum Master	Strongly	Strongly	Strongly	Disagree	Neutral	Disagree	Strongl	Neutral	No	Neutral	Disa	Disa	Disa	Disagree	Some

31			25		Disagree	Disagree	Disagree				y Disagree				gree	gree	gree		time
U232	Developer	4-7 years	16-25	Client; Scrum Master	Strongly Disagree	Strongly Disagree	Strongly Disagree	Disagree	Neutral	Disagree	Strongly Disagree	Neutral	No	Neutral	Disagree	Disagree	Disagree	Disagree	Sometime
U233	Scrum Master	8-10 years	25-35	Product Owner; Client; Scrum Master; Development Team; All of these	Strongly Agree	Strongly Disagree	Strongly Disagree	Disagree	Disagree	Disagree	Disagree	Disagree	No	Disagree	Disagree	Disagree	Neutral	Disagree	Sometime
U234	Scrum Master	More than 10 years	25-35	Client; Scrum Master; Development Team	Disagree	Disagree	Strongly Disagree	Agree	Neutral	Agree	Agree	Neutral	No	Strongly Disagree	Disagree	Disagree	Disagree	Strongly Disagree	Frequently
U235	Product Owner	8-10 years	25-35	Client; Scrum Master	Strongly Agree	Strongly Agree	Strongly Disagree	Strongly Disagree	Strongly Disagree	Disagree	Disagree	Neutral	No	Disagree	Neutral	Agree	Neutral	Neutral	Frequently
U236	Developer	8-10 years	16-25	Product Owner; Client; Scrum Master; Development Team; All of these	Disagree	Neutral	Disagree	Strongly Disagree	Disagree	Neutral	Disagree	Disagree	No	Disagree	Neutral	Disagree	Strongly Disagree	Strongly Disagree	Seldom
U237	Product Owner	4-7 years	Less than 15	Product Owner; Client	Agree	Agree	Agree	Neutral	Neutral	Disagree	Disagree	Neutral	Yes	Neutral	Neutral	Agree	Neutral	Agree	Sometime
U238	Scrum Master	8-10 years	16-25	Scrum Master; Development Team	Disagree	Disagree	Agree	Neutral	Disagree	Disagree	Strongly Disagree	Strongly Disagree	Yes	Neutral	Neutral	Disagree	Disagree	Disagree	Frequently
U239	Project Manager	More than 10 years	25-35	All of these	Agree	Disagree	Neutral	Disagree	Disagree	Disagree	Disagree	Disagree	No	Disagree	Strongly Disagree	Neutral	Agree	Agree	Seldom

U240	Scrum Master	4-7 years	Less than 15	Product Owner; Client; Development Team	Neutral	Agree	Neutral	Agree	Neutral	Neutral	Agree	Agree	Yes	Agree	Agree	Agree	Agree	Neutral	Frequently
U241	Project Manager	8-10 years	25-35	Client; Scrum Master; Development Team	Disagree	Disagree	Agree	Disagree	Neutral	Agree	Neutral	Disagree	No	Disagree	Disagree	Neutral	Disagree	Disagree	Always
U242	Developer	More than 10 years	25-35	Product Owner; Client; Scrum Master	Disagree	Disagree	Agree	Agree	Strongly Agree	Strongly Agree	Disagree	Agree	Yes	Neutral	Agree	Disagree	Disagree	Neutral	Frequently
U243	Project Manager	4-7 years	Less than 15	All of these	Neutral	Neutral	Agree	Neutral	Neutral	Neutral	Agree	Neutral	Yes	Neutral	Agree	Neutral	Agree	Agree	Frequently
U244	Developer	More than 10 years	16-25	Scrum Master; Development Team	Disagree	Agree	Disagree	Neutral	Neutral	Disagree	Neutral	Agree	Yes	Neutral	Neutral	Strongly Agree	Agree	Disagree	Frequently
U245	Product Owner	8-10 years	More Than 35	Client; Scrum Master	Disagree	Disagree	Strongly Disagree	Strongly Disagree	Disagree	Disagree	Neutral	Neutral	No	Disagree	Disagree	Neutral	Neutral	Neutral	Sometime
U246	Scrum Master	8-10 years	More Than 35	Client; Scrum Master	Agree	Disagree	Disagree	Disagree	Neutral	Disagree	Neutral	Disagree	No	Disagree	Neutral	Disagree	Neutral	Disagree	Never
U247	Scrum Master	4-7 years	Less than 15	Product Owner	Disagree	Disagree	Neutral	Agree	Neutral	Agree	Agree	Neutral	Yes	Agree	Neutral	Neutral	Agree	Agree	Sometime
U248	Project Manager	8-10 years	25-35	Scrum Master	Agree	Neutral	Strongly Disagree	Strongly Disagree	Strongly Disagree	Neutral	Disagree	Disagree	Yes	Disagree	Neutral	Disagree	Disagree	Disagree	Frequently

U249	Product Owner	4-7 years	Less than 15	Product Owner; Client	Agree	Agree	Neutral	Agree	Agree	Neutral	Neutral	Neutral	Yes	Agree	Agree	Agree	Neutral	Neutral	Frequently
U250	Project Manager	4-7 years	25-35	Client; Scrum Master	Neutral	Neutral	Strongly Disagree	Strongly Disagree	Disagree	Disagree	Disagree	Disagree	Yes	Neutral	Disagree	Disagree	Neutral	Disagree	Frequently
U251	Scrum Master	8-10 years	More Than 35	Client; Scrum Master	Agree	Disagree	Disagree	Strongly Disagree	Strongly Disagree	Neutral	Strongly Disagree	Disagree	Yes	Neutral	Disagree	Agree	Agree	Disagree	Seldom
U252	Developer	0-3 years	Less than 15	Product Owner; Client; Development Team	Agree	Neutral	Agree	Agree	Agree	Agree	Strongly Agree	Agree	Yes	Strongly Agree	Strongly Agree	Strongly Agree	Strongly Agree	Agree	Frequently
U253	Scrum Master	4-7 years	Less than 15	Product Owner; Client	Neutral	Neutral	Agree	Neutral	Neutral	Neutral	Agree	Agree	Yes	Disagree	Disagree	Disagree	Neutral	Neutral	Always
U254	Project Manager	4-7 years	More Than 35	Product Owner	Agree	Agree	Neutral	Neutral	Disagree	Neutral	Agree	Strongly Agree	Yes	Strongly Agree	Agree	Disagree	Agree	Agree	Always
U255	Software Quality Assurance Engineer	0-3 years	16-25	Product Owner; Client	Agree	Disagree	Neutral	Strongly Agree	Agree	Agree	Agree	Agree	Yes	Strongly Agree	Agree	Neutral	Agree	Agree	Frequently
U256	Project Manager	0-3 years	16-25	All of these	Agree	Agree	Neutral	Disagree	Agree	Strongly Agree	Agree	Neutral	Yes	Agree	Neutral	Agree	Neutral	Neutral	Sometime
U257	Scrum Master	0-3 years	Less than 15	Product Owner	Agree	Agree	Agree	Agree	Strongly Agree	Agree	Agree	Agree	Yes	Neutral	Agree	Agree	Strongly Agree	Agree	Sometime

U258	Software Architect	8-10 years	16-25	Product Owner; Scrum Master; Development Team	Strongly Agree	Agree	Strongly Agree	Agree	Strongly Agree	Agree	Strongly Agree	Strongly Agree	Yes	Strongly Agree	Agree	Strongly Agree	Strongly Agree	Strongly Agree	Frequently
U259	Scrum Master	More than 10 years	More Than 35	Product Owner	Agree	Strongly Agree	Agree	Agree	Strongly Agree	Strongly Agree	Agree	Neutral	Yes	Agree	Agree	Agree	Agree	Agree	Sometime

Appendix C: Question asked from focus group

Serial Number	Question
1	What is your opinion there should be any online tool where all stakeholders can participate at the time of requirement prioritization.
2	Do you agree that there should be one common person who should gather all prioritization information from team so they can be considered together when actual prioritization happens?
3	Do you agree that product owner should validate requirements with development team to have their side of the story as well.
4	Do you think product owner should have some technical background.
5	Do you agree developers should be the part of requirement prioritization at least one nominated person from the development team should be present.
6	What do you think technical limitations should be accessed while doing the requirement gathering and requirement prioritization to reduce software failure.
7	In your opinion assigning dependency score to everything in the product backlog is important.
8	In your opinion having some time buffer for each requirements to make sure we have some extra time to solve any unexpected challenge in the sprint will help to cater time constraints. .

9	What do you think using async communication tool like Microsoft Team and Slack could help in better communication coordination.
10	What do you think prioritize requirements scores should be really comprehensive. They should include all technical, non-technical, and geographical constraints.
11	What do you think by using a requirement prioritization framework so everyone can follow a guideline.

Appendix D: Calculation of weightage value

NO	Factors	Strongly Agree (2)	Agree (1)	Neutral (0)	Disagree (-1)	Strongly Disagree (-2)	Total
	Stakeholders participation						
1	**Use some online method where all stakeholders can participate.**	1*2=2	4*1=4	0*0=0	0	0	6
2	**Stakeholder's participation should happen in a sync environment.**	0	0	0	2*-1=2	3*-2=-6	-8
3	**One common person can gather all prioritization information from team so they can be considered together when actual prioritization happens**	4*2=8	1*1=1	0	0	0	9
4	**Focus on having all information about requirements with everyone so that everyone has a clear understanding of each requirement. That can help in requirement prioritization**	4*2=8	1*1=1	0	0	0	9
	Product owner technical knowledge constraints						
5	**Product owner should validate requirements with development team to have their side of the story as well**	3*2=6	2*1=2	0	0	0	8
6	**Rather having only overview of the requirements, product owner should explore in-depth details for better technical understanding**	2*2=4	3*1=3	0	0	0	7
7	**Product owner should have some technical background**	5*2=10	0	0	0	0	10
	Developers Involvement						

8	**Developers should be part of requirement gather process to access the complexity and technical constraints**	3*2=6	2*1=2	0	0	0	8
9	**Developers should be the part of requirement prioritization at least one nominated person from the development team should be present**	3*2=6	2*1=2	0	0	0	8
10	**Developers should get involved at very last stages when most of the requirements are clear to have a technical assessment.**	0	5*1=5	0	0		5
11	**Developers should participate in business requirement discussion to help identify technical constrains**				3*-1=-3	2*-2=-4	-7
	Technical limitations						
12	**Technical limitations should be accessed while doing the requirement gathering and requirement prioritization to reduce software failure.**	3*2=6	2*1=2	0	0	0	8
13	**Technical dependencies should be addressed as well**	2*2=4	3*1=3	0	0	0	7
14	**All requirements should have a dependency score and limitation score to make them easily identifiable.**	3*2=6	2*1=2	0	0	0	8
15	**A technical person should be always present in the process of requirement prioritization.**	4*2=8	1*1=1	0	0	0	9
	Requirement dependencies						
16	**Assign dependency score to everything in the product backlog**	4*2=8	1*1=1	0	0	0	9
17	**Involve technical experts to access technical dependencies which will ensuresoftware success.**	4*2=8	1*1=1	0	0	0	9
18	**Similar dependency score requirements should be grouped to understand product backlog complexity.**	0	0		3*-1=-3	2*-2=-4	-7
19	**Highest dependency score requirements should be addressed first from product backlog**	2*2=4	3*1=3	0	0	0	7
	Time constraint						
20	**All stakeholder should do time estimations of requirements**	0	1	0	3*-1=-3	1*-2=-2	-4

21	**Time constraint should also be given some value while prioritizing requirements for a sprint.**	2*2=4	3*1=3	0	0	0	7
22	**Have some time buffer in each requirements to make sure we have some extra time to solve any unexpected challenge in the sprint.**	1*2=2	4	0	0	0	6
	Communication, Coordination & Collaboration						
23	**There should be at least 3-4 hours of time overlap in the distributed team.**	1*2=2	4	0	0	0	6
24	**All team members should not diverge more than 3 hours of time in time zone for easier communication.**	0	0	0	4*-1=4	1*-2=-2	-6
25	**Consider using async communication tool like Microsoft Team and Slack.**	3*2=6	2*1=2	0	0	0	8
26	**Minimize the collaboration required. Build processes that uses documentation and lesser human involvement all the time.**	3*2=6	2*1=2	0	0	0	8
	Distributed Scrum Requirement Prioritization Constraints						
27	**Prioritize requirements scores should be really comprehensive. They should include all technical, non-technical, and geographical constraints.**	3*2=6	2*1=2	0	0	0	8
28	**Every stakeholder should participate in the requirement prioritization process in distributed scrum**	1*2=2	4*1=4	0*0=0	0	0	6
29	**Use a requirement prioritization framework so everyone can follow a guideline**	1*2=2	4*1=4	0*0=0	0	0	6

Appendix E: Accepted and rejected guidelines through focus group

NO	Factors	Weightage Value	Avg. Weightage Response	Results
	Stakeholders participation			
1	**Use some online method where all stakeholders can participate.**	6	6/5=1.2	Accepted
2	**Stakeholder's participation should happen in a sync environment.**	-8	-8/5=-1.6	Rejected
3	**One common person can gather all prioritization information from team so they can be considered together when actual prioritization happens**	9	9/5=1.8	Accepted
4	**Focus on having all information about requirements with everyone so that everyone has a clear understanding of each requirement. That can help in requirement prioritization**	9	9/5=1.8	Accepted
	Product owner technical knowledge constraints			
5	**Product owner should validate requirements with development team to have their side of the story as well**	8	8/5=1.6	Accepted
6	**Rather having only overview of the requirements, product owner should explore in-depth details for better technical understanding**	7	7/5=1.4	Accepted
7	**Product owner should have some technical background**	10	10/5=2	Accepted
	Developers Involvement			
8	**Developers should be part of requirement gather process to access the complexity and technical constraints**	8	8/5=1.6	Accepted
9	**Developers should be the part of requirement prioritization at least one nominated person from the development team should be present**	8	8/5=1.6	Accepted
10	**Developers should get involved at very last stages when most of the requirements are clear to have a technical assessment.**	5	5/5=1	Accepted
11	**Developers should participate in business requirement discussion to help identify technical constrains**	-7	-7/5=-1.4	Rejected
	Technical limitations			

12	**Technical limitations should be accessed while doing the requirement gathering and requirement prioritization to reduce software failure.**	8	8/5=1.6	Accepted
13	**Technical dependencies should be addressed as well**	7	7/5=1.4	Accepted
14	**All requirements should have a dependency score and limitation score to make them easily identifiable.**	8	8/5=1.6	Accepted
15	**A technical person should be always present in the process of requirement prioritization.**	9	9/5=1.8	Accepted
	Requirement dependencies			
16	**Assign dependency score to everything in the product backlog**	9	9/5=1.8	Accepted
17	**Involve technical experts to access technical dependencies which will ensure software success.**	9	9/5=1.8	Accepted
18	**Similar dependency score requirements should be grouped to understand product backlog complexity.**	-7	-7/5=-1.4	Rejected
19	**Highest dependency score requirements should be addressed first from product backlog**	7	7/5=1.4	Accepted
	Time constraint			
20	**All stakeholder should do time estimations of requirements**	-4	-4/5=-0.8	Rejected
21	**Time constraint should also be given some value while prioritizing requirements for a sprint.**	7	7/5=1.4	Accepted
22	**Have some time buffer in each requirements to make sure we have some extra time to solve any unexpected challenge in the sprint.**	6	6/5=1.2	Accepted
	Communication, Coordination & Collaboration			

23	**There should be at least 3-4 hours of time overlap in the distributed team.**	6	6/5=1.2	Accepted
24	**All team members should not diverge more than 3 hours of time in time zone for easier communication.**	-6	-6/5=-1.2	Rejected
25	**Consider using async communication tool like Microsoft Team and Slack.**	8	8/5=1.6	Accepted
26	**Minimize the collaboration required. Build processes that uses documentation and lesser human involvement all the time.**	8	8/5=1.6	Accepted
	Distributed Scrum Requirement Prioritization Constraints			
27	**Prioritize requirements scores should be really comprehensive. They should include all technical, non-technical, and geographical constraints.**	8	8/5=1.6	Accepted
28	**Every stakeholder should participate in the requirement prioritization process in distributed scrum**	6	6/5=1.2	Accepted
29	**Use a requirement prioritization framework so everyone can follow a guideline**	6	6/5=1.2	Accepted