AN INDUSTRY SURVEY OF DEMOTIVATORS FOR SCALING UP AGILE METHODOLOGY

By

OSAMA TARIQ

NATIONAL UNIVERSITY OF MODERN LANGUAGES

ISLAMABAD

MAY, 2022

An Industry Survey of Demotivators for Scaling Up Agile Methodology

By

OSAMA TARIQ

BSCS, University of Arid Agriculture, Rawalpindi, 2018

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

MASTER OF SCIENCE

In **Software Engineering**

To

FACULTY OF ENGINEERING & COMPUTER SCIENCES

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 Computer Sciences for acceptance.

Thesis Title: **An Industry Survey of Demotivators for Scaling Up Agile Methodology**

Submitted by: Osama Tariq **Registration #:** MSSE/Ibd/S19

Master of Science in Software Engineering
Degree name in full

Software Engineering
Name of Discipline

Dr. Muzafar Khan	________________________
Name of Research Supervisor	Signature of Research Supervisor
Dr. Basit Shahzad	________________________
Name of Dean (FE&CS)	Signature of Dean (FE&CS)
Prof. Dr. Muhammad Safeer Awan	________________________
Name of Pro-Rector Academics	Signature of Pro-Rector Academics

May 18th, 2022
Date

AUTHOR'S DECLARATION

I Osama Tariq

Son of Tariq Mahmood Khokhar

Registration # MSSE/Ibd/S19

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 **An Industry Survey of Demotivators For Scaling Up Agile Methodology** 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 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 cancelled and the degree revoked.

Signature of Candidate

Osama Tariq

Name of Candidate

May 18th, 2022

Date

ABSTRACT

Traditional software development approaches advocate heavy upfront planning, extensive documentation and reluctance to change adoption. These characteristics attributed to the failure of many software development projects in the past. Eventually, agile software development approach evolved that changed many of the aspects of traditional software development such as flexible planning, light documentation, change embracing approach. These approaches yielded better results when applied to the small-scale software projects but challenges were encountered when agile approaches were applied to large scale software projects. This research study aims to seek the opinion of the industry practitioners regarding the demotivators faced while scaling agile methodologies as mentioned in the literature. Questionnaire survey has been adopted as the research methodology due to its aptness in this research study. 143 survey respondents have contributed their valuable opinions for data collection in this research study. To map the industry survey findings with the literature survey, a comparison has been made between the top ranked demotivators from literature and industry survey. Statistical data analysis reveals a high degree of consistency between the findings of literature review and the opinion of large-scale agile software practitioners. Moreover, the best practices to address the demotivators have also been discussed at length.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

DEDICATION

I would like to dedicate this effort to the altruistic support and motivation of my beloved parents who prayed for me without my knowledge and my best teacher respected Dr. Muzafar Khan, whose guidance always accompanied me throughout the journey of my life.

CHAPTER 1

INTRODUCTION

1.1 Overview

This chapter basically briefs out the motivation for conducting this research study, problem statement, research questions, aim of the research, research objectives, scope of this research work and then the thesis organization defines the structure of the remaining chapters.

1.2 Context

Software development is a process consisting of several steps undertaken in a specific sequence to produce a software system [1]. The software development process uses different phases, all logically linked with each other with the output of one phase being the input of another [1]. Different software development models were developed in the past that tended to represent the software development process from different perspectives, although the purpose of each model was to produce a finished software product in the end [2]. The proposed models only presented the possible flow of activities in software development process [3]. The first ever model of software development practically presented in 1970 by Willian W. Royce was the Waterfall Model [4]. The waterfall model was defined to be such a model where all the activities are performed one after the other in a particular sequence [5]. The sequence of the activities mostly remains fixed and resembles the waterfall where all the activities cascade from top to bottom in a linear fashion [6]. When applied in the context of software development, the waterfall method dictated that all the phases in software development would be strictly followed in a linear method with the former stage precisely preceding the next [7].

The waterfall was the earliest proposed software engineering design model that gained wide acceptance in the software development community [5]. The software engineers started building software systems based on the waterfall model and that era witnessed great success of waterfall model [8]. Based on the statistics of researchers of Standish Group, the adoption of waterfall models experienced 44% successful small scale software projects with just 11% failure rate which unambiguously indicated its high suitability in small scale software development projects [9]. Waterfall model, also called as plan-driven model or represented by the umbrella term "traditional software development methodology" [6], was actually based on a prefixed series of phases most commonly referred in literature as requirements elicitation, software design, implementation, testing and maintenance [10]. The concept behind the waterfall model was strict adherence to the defined phases and the next stage could only be initiated once the previous stages were complete in all aspects [11]. The entire roadmap was to be planned in advance, the milestones, cost of production, time and effort involved along with heavy documentations and artefacts associated with every important aspect of software development [12]. Once the previous phases were crossed, there was no concept of backtracking and reworking after the identification of improvements and corrections in the later stages, everything had to planned again and started from scratch [13]. But due to the dynamic and flexible nature of software projects where the gathered requirements were by no means complete and the changing market trends and customer requirements proved the collected requirements to be incomplete in many aspects, the traditional software development approaches increased the quantity of rework required which in turn pushed up the cost of production and time involved, causing most of the software projects to lose their initial calculations [10].

The hard and fast upfront planning and calculation was not exhaustive, the progressive software development brought many of the changes in the initial blueprint that in turn necessitated the process of backtracking and introduced rework in the completed tasks ranging from requirements set to software design to coding and testing [12]. The initial planning worked somewhat unchanged for small scale software projects and the traditional software development methodologies became famous for being associated with small to somewhat medium scale projects [14] because of smaller requirements set in such projects, less involvement of stakeholders and less time and associated cost. The large-scale software projects being dynamic and multi-featured required the input of lot of stakeholders and hence a large requirement set which was very difficult to predict in advance [15]. The changing needs of business environment made it impossible for developers to forestall all the unforeseen events of software development in the form of rigid project plan artefacts [15]. But the traditional approaches when applied to large scale software projects caused multi-dimensional problems in terms of cost overrun, time overrun, excessive rework, frequent slippages and decreasing trust of the stakeholders [16]. Resultantly, the traditional approaches slowly got sidelined and the developers started searching out ways to deal with the inherent problems of traditional software development methodologies [12].

1.3 Related Work

The traditional software approaches remained in practice for small and medium scale software projects due to their suitability, but the significant failure of large-scale projects made the applicability of traditional software approaches less useful [12]. According to the report from Standish Group International [9], 42% of the large-scale software projects employing traditional methodology failed and 55% of such projects got challenged which indicates a lack of suitability / adaptiveness of traditional approaches in large scale software projects. The decreasing success rate of traditional methodologies in large-scale software development opened the ways for another software development methodology referred to in literature as incremental or iterative software development methodology [17]. Iterative methodology, as the name suggests, develops the software in different iterations [18]. Initially the set of requirements is collected from all the stakeholders and the gathered requirements are then prioritized based on the input from different stakeholders [19]. The related requirements are then taken from the prioritized list, implemented and released in the form of small increments with each successive increments building up the software system towards completion [18]. Here, the estimation is done for each iteration instead of complete software development process which in turn consumes less resources and proves more flexible than heavy upfront planning which undergoes significant changes over the course of software development process [20].

The start of 21st century witnessed many significant changes in the world of software engineering. The researchers came up with an entirely different lightweight software development technique called as agile methodology [21]. The agile methodology, in contrast to the traditional plan driven approaches, was based on agile manifesto in which individuals and interactions are valued over processed and tools, working software is valued over comprehensive documentation, customer collaboration over contract negotiation and responding to changes over following a plan [22]. The agile methodologies focused on the minimal documentation and essential planning that was subject to alteration upon recognition of changes due to changing business needs, stakeholders' requirements, government regulations etc. [13].

However, the introduction of change management attributes in agile methodologies made it quite suitable and favorable for those software projects that typically face the problem of requirement uncertainty resulting in significant shuffling of requirements set even in later stages of software development [23]. The flexibility of agile methods favored their applicability on small to medium scale software projects because of less requirement modifications, few stakeholders and relatively short development cycle [6]. According to a study conducted by Dan Schilling [24], there have been 72% of the software projects that adopted agile methodologies and successfully completed within time and budget constraints up to the stakeholders' expectations. In contrast to small to medium scale software projects, large scale projects have numerous complexities in terms of budget, time constraints, number of stakeholders involved, evolving business needs, ever changing large requirements set with associated changes in software architecture, design, coding and integration among others [25]. Due to the varied nature of large-scale projects, the application of agile methodologies proved to be less favorable and encountered several problems over the course of their application [26].

Several articles in the literature have referred to the problems associated with implementation of agile methodologies at large scale as challenges / challenging factors [27], demotivators etc. [28]. A few studies found in the literature have tried to explore the challenges and demotivators faced up while scaling agile methodologies. Authors in [29] have explored and presented the literature findings of challenges and success factors for scaling up agile methodologies. In another article [30], researchers have worked on the challenges that organizations faced and success factors that helped organizations while adopting and scaling agile methodologies. Yet in another article, the authors have detailed out the demotivators for adoption of agile methodologies at large scale agile projects purely from literature's perspective [28].

1.4 Motivation

While scaling up agile methods to tailor them for large scale software projects, the demotivators are faced which hinder their successful implementation and in turn make the software project less successful in different qualitative and quantitative aspects [31]. The demotivators faced while scaling up agile methodologies have been mentioned from literature's perspective only [28] [29] [32]. Alongside the demotivators, few studies have highlighted the suggested practices that can help addressing and mitigating the effects of those demotivators [30]. Hence, there was a need to seek the opinion of industry practitioners regarding demotivators faced while scaling up agile methodologies along with the appropriateness and practical effectiveness of the suggested practices to address the specified demotivators as mentioned in the literature [28] [29]. This research study aims to identify such demotivators from literature and conduct an industry survey to seek the opinion of industry practitioners regarding the demotivators faced while scaling up agile methodologies. This became the basis and motivation for pursuing this research study.

1.5 Problem Statement

The demotivators faced while scaling up agile methodologies at large scale software development projects have been pointed out in the literature [28] [29] [30]. However, there is a need to investigate the viewpoint of industry practitioners about such demotivators like communication challenges in multi-team environments, lack of proper planning for large scale agile projects, scarcity of large-scale agile experts, complexity of large-scale projects, difficult implementation of agile at large scale etc.

are some of the commonly mentioned demotivators while dealing with large scale agile development projects [28] [29]. Furthermore, the opinion of industry practitioners needs to be sought regarding the suggested practices mentioned by authors in the literature to address the demotivators encountered while scaling up agile methodologies [28].

1.6 Research Questions

This research aims to provide the answers to the following research questions by following a systematic research strategy.

i. What are the demotivators faced by the industry practitioners during the large-scale agile development projects?

ii. What are the best practices to address the identified demotivators for large scale agile development projects?

1.7 Aim of the Research

This study aims to seek the opinion of the industry practitioners working on large scale agile development projects regarding the demotivators faced while adopting agile methodologies for large scale software development projects. It also aims to establish the concurrence / relevance between the findings of the literature and the opinion of the industry practitioners regarding the demotivators faced while scaling up agile methodologies. Moreover, it this study presents the set of best practices to address each demotivator in an effective manner based on the opinion of software practitioners and the findings of literature survey.

1.8 Research Objectives

The following objectives are intended to be achieved by conducting this research study.

i. To list out the demotivators for scaling agile methodologies for large scale software development projects as found in the literature and verified by the industry practitioners.

ii. To present the set of best practices in the opinion of industry practitioners to address the identified demotivators.

1.9 Scope of Research Work

This research work will be explicitly dealing with the demotivators faced while scaling up agile methodologies on large scale software projects by presenting the opinion of industry practitioners regarding the demotivators identified from literature survey. The priorities of the identified demotivators, based on their relative criticalities, will be presented and the set of best practices to deal with each of the demotivator will be listed out to help address them in the most efficient manner.

1.10 Thesis Organization

The rest of thesis is organized as follows; Chapter 2 provides a brief background on literature review, Chapter 3 presents research methodology adopted in this research study, Chapter 4 presents survey data collection, Chapter 5 discusses the data analysis and reporting of results, Chapter 6 provides the overall conclusion and future research directions.

CHAPTER 2

LITERATURE REVIEW

2.1 Overview

Literature review is the first and foremost step undertaken while conducting research in any discipline wherein all the existing publications including journal articles and conference proceedings from renowned digital libraries on a particular topic are explored to gauge the extent of research conducted till now in a particular field [33]. Thereafter, the research gaps are identified based on the contemporary issues being discussed actively among the researchers to direct the future research based on the identified gaps [34]. In this chapter, a brief description of the published articles collected and studied as part of literature survey of this research study along with the identified research gaps has been comprehensively presented.

2.2 History of Software Development

The history of software engineering dates back to as early as 1969 when the term software engineering was coined in NATO conference [35]. A number of software developers gathered to discuss the surging software development crisis associated with large number of failed and challenged projects, lack of standardized practices and guidelines for software development process and frequent collapse of the existing software systems during operational use [36]. The conference aimed to address the aforesaid problems by recognizing the software development field as a separate discipline and coining the term “software engineering”, designing the set of principles for the software development discipline with the purpose to develop standardized practices to achieve uniformity in the work practices of developers around the globe, which will, in turn, lead to the production of higher quality durable software products, both in terms of time and cost, up to the satisfaction of the existing and prospective stakeholders [36]. The proceedings of the conference gave rise to the pursuit of development and standardization of the software development practices in the developers community [37], which, in turn, led to the proposition of the first ever development model in the software engineering history commonly known as "Waterfall Model" [3]. The software development model aimed to simplify as well as standardize the different work practices followed by the developers in software development process, resulting in the evolution of structured software development life cycle, commonly referred to as SDLC in literature [38].

2.3 Proposition of Waterfall Model

The first software development model, namely, the waterfall model was proposed by Winston W. Royce in 1970 [39] which aimed to address the emerging software crisis by providing an initial blueprint to the software engineers to streamline the software development process, albeit in a very rudimentary manner [2]. The waterfall model typically imitated the water stream falling off the rock, strictly in the downward manner with no chance of reverse motion [7]. The different steps of software development process, also referred to as phases of waterfall model, started from requirements gathering and definition, proceeding to second phase of system design, which, in turn led to the third phase of implementation, followed by the software testing phase which almost ended the software development lifecycle with the last step, operational maintenance associated with the software system for the lifetime [40]. The development of waterfall model practically aided the software engineers by providing pragmatic guidelines for building software systems with comparative ease [6]. According to a survey conducted by Johnson [9], approximately 69% of the projects employing waterfall model were declared successful in meeting their originally stated objectives. The percentage of challenged and unsuccessful software projects developed using waterfall methodology remained close to 41% and 29% respectively, speaking of the high success rate of waterfall methodologies [5]. When the success rate of waterfall methodologies is discussed with reference to the size of the software projects, 67% small scale projects, 38% medium scale projects and 21% large scale projects could make their way to success [8]. The waterfall model perceived the software development process as a rigid and fixed sequence of activities that was to be undertaken in a strict linear fashion [40]. The successive phase could only be initiated once the preceding phase was complete in all respects [41], and once the next phase was entered into, there was no concept of backtracking and reverting to the previous phases to fix something discovered at a later stage of SDLC [10]. Software systems, unlike other physical systems in the real world, are quite different in nature and other attributes, the most important of which is their intangible nature, which makes them highly flexible [1]. The development of software systems is based on several different parameters viz inputs from different stakeholders, the needs of ever changing and evolving business environment, government laws and regulations, adoption to the changing market trends to name a few [7].

Quite frequently, the software development process entailed the procedure of backtracking to the previously completed phases of SDLC due to a multitude of reasons viz identification of vital functional / non-functional requirements at later stage of SDLC for inclusion in the system, incorrect inclusion of unnecessary requirements leading up to their inclusion in design and implementation phases, logical errors in software coding detected during the testing phase causing rework in coding stage etc. [42]. The waterfall methodologies adopted the notion of planning everything well in advance to continue with the same roadmap for the entire SDLC, which is never possible even in small scale software projects due to myriad of unforeseen factors outside the natural control of software engineers and project managers [43]. The dynamic nature of software systems, when coupled up with the rigid and sequential practices of waterfall methodologies, caused frequent problems for developers to start the whole project from scratch even in the finishing stages which caused cost and time overrun unnecessarily [44]. The rework, being directly proportional to the size and hence, the complexity of the software projects, was somewhat affordable in small to medium scale projects but became unbearable for large scale software projects [45], evident from the diminishing success rate of waterfall methodology by moving from small to large scale projects, ultimately leading the researchers to come up with an alternate, yet workable strategy to counter the inherent and stringent flaws of waterfall methodology [45].

2.4 Transition to Prototyping Methodology

The shortcomings of waterfall model paved the way for another software development model namely prototyping model [46]. This methodology is one step ahead of the rigid waterfall model and suggests a rather flexible approach to software development [47]. Prototyping model typically works on gathering a set of functional requirements from stakeholders and building up a minimal working model viz prototype based on the essential requirements for the feedback of the stakeholders [46]. The feedback of the stakeholders is incorporated in the next version of prototype which is placed before the stakeholders for enhancements [48]. Each feedback cycle refines the prototype till the time it matches the stakeholders needs and demands in totality [48].

The main idea behind prototyping model is to initially gather all the requirements from the stakeholders in the start, identify the critical functional requirements and build up a crude working system by implementing those necessary requirements [46]. Once the prototype has been designed, the same is passed repeatedly through feedback loops after consultation with the stakeholders to refine the system design until all the requirements have been implemented satisfactorily [49]. The prototyping methodology, in contrast to waterfall methodology, is flexible in nature that relies on development of software systems by incorporating changes in the software development process [47]. The studies reported in [14] suggest a success rate of 71% for software projects employing prototyping methodology. The prototyping method brought an improvement in software engineering domain compared with the traditional software methodology viz the waterfall model, but the large-scale software systems could not be successfully developed using prototyping model based on the large requirement set of such systems, ever evolving requirements, large time-spans of large scale projects, involvement of numerous stakeholders including government entities challenged the application of prototyping models for large scale projects [50]. Eventually, prototyping methodology phased out in favor of a better software development methodology.

2.5 Introduction of Agile Methods

The increased requirements and business environment change was necessitating the proposition of a unique software development methodology to address the problems of requirements and business environment change during software development lifecycle, hitherto left unaddressed by the plan driven / traditional software development approaches [45]. It was not until 2001 when researchers gathered to consider an altogether different approach to software development to counter the challenges posed by the traditional development methodologies [10]. The researchers came up with a mindset [22], governed by four key principles that redefined the future of software engineering. The key principles aka “Agile Manifesto” were carefully designed keeping in view the inherent flaws in the previous software development paradigms, that caused numerous software projects to fail, causing huge losses [22].

2.5.1 Agile Manifesto

The usage of software development processes and tools was strictly defined in plan driven approaches whereas agile took a different stance by stating that individuals and interactions should be preferred over processes and tools because software is developed by individuals, not by processes and tools [51]. The processes aid the developers in developing the software, whereas the verification and validation of the developed content comes from the customers which clearly states the importance of individuals over the processes and tools [21]. Agile manifesto gave preference to working software over comprehensive documentation to increase the importance of the purpose of software development [22]. Earlier approaches focused heavily on the production of formal documents aka artefacts, associated with each and every phase of software development life cycle to document all the changes and the associated working [52]. These heavily documented artefacts were of little importance during and after the software development process as the development team documented all the minor and major events associated with each phase and these artefacts were usually not shared with the customer [10]. As agile methodologies claimed to be customer centric, they dictated the usage of minimal documentation necessary to document the system design and most important changes associated as the artefacts in a changing business environment become outdated very quickly [45]. As the customer is interested in the rightful use of the software product, it should be given preference upon production and maintenance of heavy software artefacts [52]. Agile approaches favored extensive customer involvement and collaboration throughout the software development process by considering customer as a necessary stakeholder and member of the development team [22]. The plan driven approaches mostly worked without the customer involvement except in the initial requirements development phase, proceeding through the successive stages without customer involvement which resulted in incorrect system architectural and interface design, necessitating major rework in the later stages of SDLC, running out the project of its initial budget and time estimates [53].

Agile advocates the concept of customer collaboration over the process of contract negotiation [53] which means that terms and conditions of the contract between the customer and developers are quite formal which are, most of the time, not followed [54]. The customer involvement makes the development process very flexible by seeking the feedback of customer in each phase which in turn leads to the design and production of a high-quality software product, fulfilling the purpose of its development [21]. For continuous improvement and upkeep of the reputation of the developers in particular and software engineering discipline in general, some of the terms of the contract may be dispensed with to yield a working software developed within specified constraints, fulfilling the stakeholders' requirements [22]. Another inherent flaw in the plan driven / traditional software development approaches was the strict adherence to the project management plan once designed in the start [55]. Any restructuring or redefinition of the plan during any stage of the development process was not considered and the managers and developers preferred to ignore any change in the existing course of action once selected [6]. This approach didn't work due to the dynamic nature of software where everything seems to change mainly requirements causing changes in software design which in turn introduced changes in software coding, subsequently necessitating rework in the testing phase [56]. Agile approaches proposed a flexible change embracing strategy to incorporate the suggested changes during any phase of the software development process by making amendments in the software development process and the project management plans accordingly [57]. Because the software development is a dynamic and retractable flow of activities, the project plans should be built keeping in consideration the possible imminent changes in future. In other words, change embracing strategy should be preferred over following a fixed rigid plan [16].

2.5.2 Outcome of Application of Agile Methodologies

The agile methodology based on the agile manifesto when put into practice, yielded more efficient results than the traditional / plan driven software development approaches [8]. Jorgensen & Magne, in their research study [58], stated that agile methodologies witnessed a remarkable software project success rate when applied to small to medium scale projects as compared to large scale software development projects. Because the agile methodologies were originally perceived to be developed for small to medium scale software projects, hence the application yielded much higher results [58].

The decreasing success rate of agile methodologies in the context of large-scale software projects led researchers to explore ways and means to make the agile methodologies applicable for large scale projects to increase their usefulness [59]. The multi-team, multi-site and multi-customer nature of the large-scale projects posed a natural challenge to the applicability and success of the agile methodologies [60].

2.6 Conception of Agile Scaling Frameworks

Based on the inherent flaws of agile methodologies when viewed in the context of large-scale software systems, researchers came up with the logical collection of principles that aim to provide a pathway for easy implementation of agile methodologies at large scale software systems commonly referred to as Agile Scaling Frameworks in the literature [61]. These frameworks tend to provide a body of knowledge consisting of the practices and procedures that can be adopted to scale agile successfully, gaining the promised success of agile methodologies in large scale software projects [59]. The first such framework to be introduced in the software engineering discipline by Dean Leffingwell is Scaled Agile Framework (SAFe) [60] which incorporates a wide range of software development practices from different variants of agile methodologies namely Scrum, Extreme Programming (XP), Dynamic Systems Development Method (DSDM), Crystal Clear Development (CCD) etc. [60]. SAFe is commonly the most popular scaling framework in large enterprises, according to a report published by VersionOne in 12th state of agile survey [62], although Large Scale Scrum (LeSS), Spotify, Nexus frameworks are also preferred by industry practitioners while implementing agile at large scale. Gruver & Mouser, in their study [63], have highlighted that the novice introduction of scaling agile frameworks in the market has not yet passed a considerable time and there is a need to investigate the adoptability of these frameworks in the software industry as well as the manner in which the organizations are adopting these scaling frameworks.

2.7 Outcome of Application of Scaling Agile Frameworks

Owing to the increasing popularity of the agile scaling frameworks, the researchers viz Paasivaara & Lassenius, in their research article [64], have tried exploring the benefits and challenges faced while adopting Scaled Agile Framework (SAFe) in large-scale software enterprises. The authors adopted the multi-vocal literature review methodology to explore the published material viz conference proceedings and journal articles as well as the grey literature / non-published material viz blogs, websites etc. The authors gathered 7 case studies from peer-reviewed sources and 47 case studies from non-peer reviewed sources regarding the implementation of SAFe framework in large-scale software enterprises. The organizations from different domains like financial, software, manufacturing as well as telecommunication have been reported to adopt the SAFe framework.

The adoption of SAFe framework reports the benefits of enhanced transparency in business practices and communications, alignment in the expectation of software development teams and stakeholders, alignment of organizational goals and policies with the work practices of SAFe framework, improved product and process quality, reduced time to market based on the adoption of SAFe framework, greater predictability in the deliverance of the quality software product, reduction in the cost of production and quality of software products, enhanced responsiveness towards the changing market trends and the stakeholders requirements etc. Among the reported challenges faced by the large-scale enterprises while adopting SAFe framework include the resistance to acceptance of change in the organizational prevailing practices, strong change resistance from development teams and project managers having no or little knowledge of SAFe framework, difficult implementation and scaling of agile methodologies for large-scale projects, using hybrid approach to software development i.e., blending plan driven approaches with agile approaches, lack of autonomy and decision making power, lack of experience staff members with varied expertise, challenges in project management viz planning releases and change management, challenges in backlog prioritization and maintenance, loss of agility using SAFe framework i.e., moving away from agile methods, unsuitability of SAFe framework in certain environments, challenges of achieving integration and coordination in multi-team environment of Global Software Development (GSD) projects etc. are the most commonly reported challenges. The authors, in their findings, reported that apart from the inherent challenges of scaling agile methodologies, SAFe framework introduces its own challenges that need to be explored. The practical usefulness of SAFe framework is limited because of the limited research conducted in this domain. Further, research can be conducted to explore the solutions to the reported challenges to help organizations address them appropriately.

Another study conducted by Conboy & Carroll [65] conducted a literature review to identify the challenges for using agile scaling frameworks and associated recommendations for each challenge to mitigate it, as reported in the literature. The authors have explored 13 research publications and extracted the 9 most commonly reported challenges. The challenges include the resistance to change readiness and adoption, balancing organizational structure with the implementation of scaling frameworks at large-scale enterprise level, loss of essence of agile methodology by over adherence to the scaling frameworks, uncertainty about the outcome of implementation of agile scaling frameworks, lack of specific metrics and criteria for the selection of agile scaling frameworks, lack of flexibility in the procedures of agile scaling frameworks, lack of autonomy of development teams, lack of experts for implementing scaling frameworks, difficult transition from plan-driven software development approaches to large-scale agile software development approaches. The authors have, however, suggested that more research needs to be conducted in the domain of scaling agile methodologies for large scale software projects to thoroughly identify the challenges to help organizations incorporate necessary procedures to address them using nimble project management approaches.

2.8 Exploration of Challenges for Scaling Up Agile Methodologies

To address the concerns of decreasing success rate of agile methodologies for large-scale software projects, the authors Hobbs & Petit in their research article [66] have adopted systematic literature review (SLR) to explore the challenges that are encountered while scaling up agile methodologies to large multi-team projects. The authors have tried to categorize the challenges for scaling up agile in two categories with the first category containing challenges inherent to the agile methods themselves and the second category containing challenges specific to the large-scale enterprise environment. The former category includes such challenges as the size of the development team, distributed existence of the software development teams, communication and coordination challenges, lack of tacit knowledge sharing between the distributed teams etc. The latter category groups the challenges like strict adherence to the formal software development procedures in large scale enterprises, following rigid project management guidelines, management unwilling to adopt the modern agile approaches, blending plan driven approaches with agile methodologies, global presence of large-scale enterprises, non-uniformity in the business practices and workflows of distributed software development teams to name a few. The authors narrated that despite the increasing trend of adoption of agile methodologies for large-scale software projects, there are basic contradictions between the nature of large-scale software projects and the agile methodologies due to which agile methods require significant transformation at project as well as organizational level to make them scalable. Here, the authors have explored the literature aspect of the challenges faced while scaling up agile methodologies, however, the industry opinion about the literature findings is still missing at large.

Moe & Mikalsen, in their research article, explored the case study of implementation of agile methodologies at a large-scale organization viz a maritime services provider company [67]. The main aim of conducting the case study was to closely monitor the process of scaling agile at large-scale with the focus to identify the challenges faced over the course of transformation. The authors adopted a mixed research methodology by collecting the data relevant to their research study through interviews, detailed analysis of business artefacts, observation of the collaboration meetings with stakeholders and development team. The authors quoted change resistance from management and development team, inter-team coordination challenges, lack of effective collaboration among diverse stakeholders, lack of large-scale agile experts and lack of knowledge while implementing agile methods as the important challenges while scaling up agile methodologies. Here, the authors have selected a single case study design which is very specific and the results so obtained cannot be generalized and relied upon completely. The authors suggested that future research can be based on exploring further challenges over the course of scaling and addressing them in an appropriate manner. Further, multiple case studies can be studied about the transformation of agile methodologies at large-scale to get a mature view of the challenges faced while scaling agile at large.

A systematic literature review conducted by Muhammad Faisal Abrar et al [28] pointed out the demotivators that hinder the scaling of agile methodologies at large scale development projects. The authors adopted systematic literature review (SLR) approach followed by contrived search criteria through which the research articles were filtered and extracted 15 demotivators from 58 relevant papers. The authors have tried to compare the identified demotivators from different perspectives like the existence of demotivators in different continents, citing of such challenges in different digital libraries and occurrence of the demotivators in different decades. The authors have provided a brief overview of the demotivators that are quite frequently faced while scaling up agile methodologies, however, the authors have only relied upon the findings from literature’s perspective only. The practical opinion of industry practitioners needs to be taken into consideration to equate the results of the literature survey with that of the practitioners’ view.

Kim et al. [27] extended the work of [28] by exploring the challenges as well as the success factors for the transformation of agile methodologies at large scale by following a systematic literature review approach. By conducting a comprehensive systematic literature review (SLR), the authors have identified 30 challenges and 25 success factors for scaling up agile methodologies for large-scale software projects. The challenges and success factors have been grouped into 9 and 7 categories respectively based on their similarity. The authors have not considered the industry practitioners' view regarding the challenges and success factors faced during the process of agile methodologies transformation. In this study, the reported challenges and success factors presented from literature have been classified into relevant categories to enable the project managers and developers to focus on specific categories of demotivators and success factors as a consolidated group instead of considering them individually, which may not prove to be an effective approach.

The authors in [68] adopted the similar strategy as that of [28] but instead focused on the motivating factors that aid in the adoption of agile practices for large scale software projects. An extensive systematic literature review has been conducted by the authors where 21 motivators have been identified from a total of 58 research papers. The authors have adopted a criterion of frequency of occurrence in literature to rank the motivating factors based on their criticality and importance. The more a demotivator is cited by the authors in literature, the more is its relative criticality. When the motivators are weighted against the defined criterion, some of the motivators have been marked as critical based on their frequency of occurrence / citation in the literature. Moreover, the motivating factors have been compared based on their existence in different regions / continents and frequency of citation in digital libraries. However, an equally important antithesis of motivating factors viz the demotivating factors that hinder the successful implementation of agile methodologies on large scale software projects have been largely skipped by the authors in the research study. However, the motivators for large scale agile development have been presented from authentic and renowned journals.

Shahbaz et al. [68] conducted a questionnaire survey from the agile practitioners of Pakistani software industry to explore the impact of challenges and success factors on the agile software development. The survey has been conducted in 23 software companies in Pakistan involving 90 industry practitioners, out of which the opinion of

67 practitioners has been finally selected. The opinion of industry practitioners has been sought through a questionnaire survey by presenting 36 motivating factors and 24 demotivating factors that motivated & demotivated them to adopt agile methodology instead of traditional software development approaches. Industry survey, employed as the research methodology in this study, is the de facto standard to authenticate the literature findings. Here, the prime focus of the research study is to elicit the motivating and demotivating factors for adoption of agile methodologies in place of traditional software development approaches in the software companies of Pakistan. But the demotivators faced while scaling up agile methodologies for large scale software development projects have not been explored.

Another study conducted by Martin & Kalenda et al [30] went one step ahead of Kim et al. [29] and adopted systematic literature review approach to explore the agile scaling practices, challenges and success factors for large scale software development and applied the theoretical literature findings to one software development company as a case study. The authors have listed the main scaling practices adopted in the transformation of agile methodologies, the core challenges faced and the main motivation factors during the entire case study. The conducted study is very specific and hence, cannot be generalized but the case study approach practically validates the theoretical findings.

The authors in [69] enhanced the contribution of Kim et al. [27] by exploring success factors and risk factors in adopting agile methods for large scale software development projects by conducting a systematic literature review. The authors aimed to utilize the theoretical output of literature review process i.e., the extracted success factors and risk factors from literature as input to propose “Large Scale Agile Adoption Model” (LSAAM) that will aid the management in successfully implementing the agile methodologies in large scale software development projects by minimizing the transformation risks. The proposed model has not been practically implemented by the authors in any industry project and hence, the practical validity of the model is not proven. However, large scale agile adoption model has been based on extensive literature survey findings and hence, can prove to be quite useful if implemented with standard practices in a controller manner.

2.9 Description of Demotivators Extracted from Literature

After a careful survey of the relevant existing research material published in renowned digital libraries, the conducted systematic literature survey has extracted the list of demotivators from different articles in the literature. A list of 64 demotivators has been complied after extraction of demotivators from the journal articles. The demotivators extracted from each article are presented in tabular format for easy comprehension.

Table 2.1: Demotivators presented by Faisal & Sohail [28]

Sr. #	**Identifier**	**Demotivator**
1	D1	Traditional Organizational Culture
2	D2	Lack of Management and Commitment Support
3	D3	Lack of agile experts
4	D4	Reluctance to adopt
5	D5	Bad customer relationship
6	D6	Problem in requirement elicitation
7	D7	Lack of customer knowledge
8	D8	Problem of team feedback
9	D9	Reduced productivity due to delay
10	D10	Lack of customer presence
11	D11	Exhaustive pair programming
12	D12	Lack of team training
13	D13	Lack of effective communication
14	D14	Lack of team orientation
15	D15	Continuous testing and integrations

Table 2.2: Demotivators presented by Dikert [28]

Sr. #	**Identifier**	**Demotivator**
1	D1	General resistance to change
2	D2	Skepticism towards the new way of working

3	D3	Top-down mandate creates resistance
4	D4	Management unwilling to change
5	D5	Lack of coaching
6	D6	Lack of training
7	D7	Too high workload
8	D8	Old commitments kept
9	D9	Challenges in rearranging physical spaces
10	D10	Misunderstanding agile concepts
11	D11	Lack of guidance from literature
12	D12	Agile customized poorly
13	D13	Reverting to the old way of working
14	D14	Excessive enthusiasm
15	D15	Interfacing between teams difficult
16	D16	Autonomous team model challenging
17	D17	Global distribution challenges
18	D18	Achieving technical consistency
19	D19	Interpretation of agile differs between users
20	D20	Using old and new approaches side by side
21	D21	Middle managers role in agile unclear
22	D22	Management in waterfall model
23	D23	Keeping the old bureaucracy
24	D24	Internal silos kept
25	D25	High-level requirement management large missing in agile
26	D26	Requirement refinement challenging
27	D27	Creating and estimating user stories hard
28	D28	Gap between long- and short-term planning
29	D29	Accommodating non-functional testing
30	D30	Lack of automated testing
31	D31	Requirement's ambiguity affects quality assurance
32	D32	Other functions unwilling to change
33	D33	Challenges in adjusting to incremental delivery pace

34	D34	Challenges in adjusting product launch activities
35	D35	Rewarding model not teamwork centric

Table 2.3: Demotivators presented by Kalenda & Hyna [27]

Sr. #	**Identifier**	**Demotivator**
1	D1	Resistance to change
2	D2	Distributed environment
3	D3	Quality assurance issues
4	D4	Integration with non-agile parts of organization
5	D5	Lack of commitment and teamwork
6	D6	Too much pressure and workload
7	D7	Lack of knowledge, coaching and training
8	D8	Requirements management hierarchy
9	D9	Measuring progress

Table 2.4: Demotivators cited by Moe & Mikalsen [67]

Sr. #	**Identifier**	**Demotivator**
1	D1	Chance resistance from management and development team
2	D2	Inter team coordination challenges
3	D3	Lack of effective coordination among diverse stakeholders
4	D4	Lack of large-scale agile experts
5	D5	Lack of knowledge while implementing agile methods

After carefully analyzing the 64 extracted demotivators for duplicates and irrelevant ones, the list of 24 demotivators has been extracted which is detailed out in Table 2.5.

Table 2.5: Filtered List of Demotivators

Sr. #	Identifier	Demotivator
1	D1	Traditional organizational culture
2	D2	General resistance to change
3	D3	Lack of management and commitment support
4	D4	Lack of agile experts
5	D5	Reluctance to adopt
6	D6	Bad customer relationship
7	D7	Problem in requirement elicitation
8	D8	Lack of knowledge
9	D9	Problem of team feedback / interfacing between teams difficult / lack of teamwork
10	D10	Reduced productivity due to delay
11	D11	Lack of customer presence
12	D12	Lack of team training
13	D13	Lack of effective communication / distributed environment / global distribution challenges
14	D14	Lack of team orientation
15	D15	Management unwilling to change
16	D16	Too high workload and pressure
17	D17	Misunderstanding agile concepts
18	D18	Agile customized poorly / misinterpretation of agile concepts
19	D19	Reverting to the old way of working / management in waterfall model
20	D20	Using old and new approaches side by side
21	D21	Creating and estimating user stories hard
22	D22	Requirement ambiguity affects quality assurance
23	D23	Lack of proper planning for large scale agile projects
24	D24	Complexity of large-scale projects

2.10 Practices to Address Demotivators from Literature

Xin Nan et al [65] conducted a Systematic Literature Review as part of his research study to explore the recommendations and guidelines that can help to address the faced challenges for scaling up agile methodologies. The research findings of Xin Nan are listed out in Table 2.6.

Table 2.6: Recommended Practices to Address Demotivators by Xin Nan [65]

Sr. #	**Identifier**	**Recommended Practices**
1	P1	Make team coordination top priority
2	P2	Self-autonomous teams
3	P3	Flexible development approach
4	P4	Investment in human resource
5	P5	Consider customer as a necessary stakeholder
6	P6	Complete and correct identification of real stakeholders
7	P7	Induct experienced team members
8	P8	Appointment of project facilitator
9	P9	Leadership change
10	P10	Routine progress feedback to point out slacks
11	P11	Outsource agile projects
12	P12	Maintain consistency in work practices

The findings of the research study regarding recommendations for mitigating the challenges for scaling up agile as extracted from a comprehensive systematic literature review conducted by Wright [32] are discussed in Table 2.7.

Table 2.7: Recommended Practices to Address Demotivators by Wright [32]

Sr. #	**Identifier**	**Recommended Practices**
1	P1	Manage conflicting requirements
2	P2	Integrate quality assurance activities in each phase
3	P3	Limited customer involvement
4	P4	Lack of face-to-face meetings
5	P5	Knowledge sharing between distributed teams
6	P6	Impart on-job training
7	P7	Formulate realistic timelines
8	P8	Conduction of team training sessions
9	P9	Maintain flexible timelines
10	P10	Develop forward advancing attitude

2.11 Summary

In this chapter, the articles relevant to the research topic have been explored as part of literature review phase and a brief description of each article comprising of its aim, research methodology, findings, limitations have been presented. Finally, the list of extracted and filtered demotivators, recommended practices from literature has been presented in a tabular and easy to understand format. Literature review sets the path for the research methodology which is briefly discussed in the next chapter.

CHAPTER 3

METHODOLOGY

3.1 Overview

This chapter covers the research methodology adopted in this study. The whole process entailing the selected research method, justification for selection of the research method, composition and conduction of questionnaire survey, demographics of the survey respondents has been discussed at length.

3.2 Research Strategy

Research is the process of exploring solutions to the new emerging problems in a particular domain [70]. It is not a haphazard process rather a highly organized and systematic process that follows a specific methodology using which the proposed problem is addressed in a step-by-step manner [71]. The methodology specifies the procedure that will be adopted for exploring answers to the research questions set by the researcher [72]. The research methodology adopted can be an interview, observation [73], survey [74], simulation [75] etc. based on the type of research study being conducted.

Research strategy describes the methods used in conducting research. It is an overall approach adopted by the researcher executed in a step by step manner while conducting research [70]. It entails different methods and procedures that are utilized over the course of research. The research methods further fall into two categories viz quantitative and qualitative research based on the nature of the research study undertaken [76]. The research strategy adopted in this study is better illustrated with the help of the following diagram.

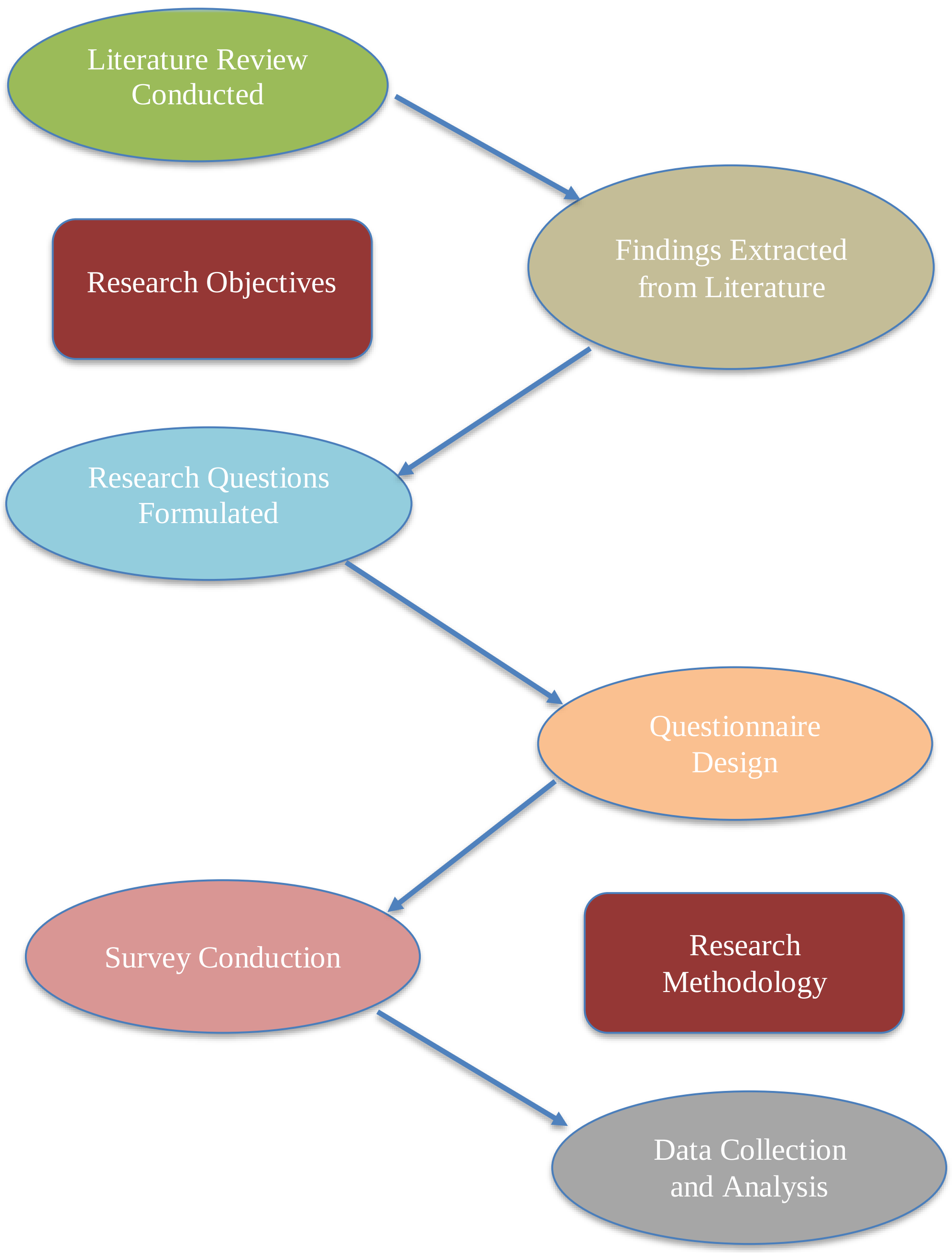
Literature Review Conducted
Research Objectives
Findings Extracted from Literature
Research Questions Formulated
Questionnaire Design
Survey Conduction
Research Methodology
Data Collection and Analysis

Figure 3.1: Research Strategy

Research methodology can either be qualitative, quantitative or a combination of both commonly referred to as mixed method research [77]. Quantitative research, as the name suggests, involves numeric data or data that can be converted to or represented through numerals [71]. The data under quantitative research is mostly gathered through experimentation and simulation [78] and the collected data undergoes statistical analysis tests to yield the desired results. Quantitative data collection yields numeric data that is easy to process using statistical tests [79]. The results of data analysis are easy to analyze and describe because of numeric nature of data [80]. However, not all studies employ quantitative data collection. In this study, quantitative research methodology has been adopted because the data to be collected and analyzed has been converted into numeric format before statistical analysis phase.

Qualitative research involves collecting and analyzing data that is non-numeric, descriptive and narrative in nature [81]. Qualitative research employs qualitative data collection methods which generate non-numeric data [82]. The qualitative data analysis involves extensive use of artificial intelligence and data mining techniques for extraction of various patterns from qualitative data because of its descriptive and open-ended nature [83]. This contrasts with the quantitative research that deals exclusively with numeric or nearly numeric data [71]. The qualitative data collection also results in greater influx of junk data because it is mostly collected through open ended questionnaires, interviews and observations [73].

Mixed research methods or multi-research method, as the name suggests, employs a combination of research methods and data collection methods [76]. These methods are mostly used in situations where it is not possible to utilize a single research methodology alone. Mixed method research may adopt a combination of quantitative and qualitative data collection methods [84], a mixture of different research methodologies [77] or even a combination of different data collection and research methodologies [77].

3.3 Survey Medium

Survey medium refers to the method or source used to collect the data for research [85]. The medium selected should be feasible, pragmatic and aid in easy collection of requisite data [86]. Various survey media are used which include telephonic survey where the data is collected from the respondents through telephonic interviews [87], email surveys where the questionnaires and responses are sent and received via email [88], face to face surveys where the respondents and researcher are physically present for participation in the research study [89], online surveys where the questionnaires are floated to intended recipients through social networking sites for effective dissemination [90], paper based survey where the survey questions are mailed / physically distributed to the participants and their responses are collected back in the similar manner [91]. The survey medium selected in this research study is the online survey method (URL of Questionnaire: https://forms.gle/DtVLBfX3KxP2hS9A6) due to its cost effectiveness in questionnaire distribution, time saving in data collection, effective inclusion of the intended recipients in the research study, and easy outreach to the global large scale agile software developers community. The responses of the respondents have been received as downloadable Microsoft Excel workbook with each row containing the responses of a single respondent.

3.4 Survey Instrument

Survey instrument refers to the technique / medium used for data collection out of which questionnaire is a very commonly used survey instrument based on the type of research being conducted [86]. Here, a combination of close ended and open-ended questionnaire has been chosen for this research study to reap the benefits of both approaches. Close ended questionnaires allow the respondents to select the most suitable options using radio buttons, check boxes, drop down menus etc. [92]. The brevity of the responses is increased in this approach, but the respondents are not able to communicate their viewpoint in a descriptive manner which suppresses the true purpose of carrying out the research study to a great extent [93]. On the flip side, open ended questionnaires give more control to the respondents to allow them to answer the questions at length which results in a greater influx of useful data along with junk data which needs to be filtered out before applying statistical tests [94]. The questionnaire has been designed in Google Forms due to its user-friendly interface which greatly aids in building up questionnaire within relatively less time.

The questionnaire has been designed in accordance with the data collection instrument guidelines presented by the authors Kitchenham & Pfleeger [95]. According to the authors in [95], survey research is initiated with the definition of the objectives of the said survey wherein the purpose of conducting the survey is stated along with its outcomes. Once the objectives are stated, then the relevant literature is explored to find out the extent of existing work explored by researchers in a particular domain as well as to find out if a similar sort of survey has already been conducted by the researchers. Based on the data explored from literature, either a survey instrument can be constructed from scratch or re-used. For the instruments to be designed from scratch, question types are decided (open, closed or a combination of both), questions are designed carefully using unambiguous language, are mutually exclusive and the answers to those questions can be numerical, yes/no answers, or based on ordinal scales.

The length of the questionnaire has been kept reasonable and the relevant questions have been grouped together in subsections. Survey instrument design can attract researcher bias based on the way the questionnaire is designed to support a particular hypothesis or premise. The instrument reliability evaluation and validation is undertaken to ascertain whether the instrument produces similar results when distributed to different respondents as well as if the instrument actually measures what it intends to measure [95]. To eliminate bias, focus groups and / or pilot studies are commonly employed to evaluate and assess the validity of the designed instrument before actually administering it which helps to improve the instrument in several ways based on the feedback from the participants of the review process. Once the survey instrument has passed content and construct validity, it is ready to be administered to the target sample for data collection.

3.5 Questionnaire Design Process

Based on the supposed guidelines of Kitchenham & Pfleeger [95], the objective of conducting the questionnaire is stated in the introduction of questionnaire to seek the opinion of the industry practitioners regarding the demotivators faced while scaling up agile methodologies, along with the best practices to address those demotivators. Once the literature review has been conducted to extract the necessary material to aid in the conduction of survey, the questionnaire has to be designed by either reusing an existing questionnaire from literature or developing the questionnaire from scratch if such instrument is not available in the literature. For instruments to be reused, the existing questionnaire is assessed for the construct and content modification to allow it to adapt to the research under study. For this research study, no such existing questionnaire has been found in the literature, leaving the only option to design the questionnaire up from scratch. To initiate the questionnaire design process from scratch, the list of demotivators and the list of suggested practices to address them have been extracted from different journal articles to constitute the substance of the questionnaire (complete list of demotivators and suggested practices presented in Chapter 2). The demotivating factors have been filtered out to remove the duplicates and the ambiguous ones so as to make the questions mutually exclusive and non-overlapping. The questions have been designed in natural and simple language without the use of technical jargon. The questionnaire has been designed as a combination of both open and close ended questions to enable the respondent to express his / her opinion in an unequivocal manner.

The first section contains the demographic information of respondents. The demographic section contains only those questions that are quite relevant to the basic information of respondents like name, designation of the respondent while working in software industry, organization name they work / previously worked for, software development experience and particularly large-scale software development experience in years and the size of the organization (small / medium / large / very large scale). Sensitive & irrelevant questions like age, gender etc., have not been included being out of context. While there might be a debate among the researchers for the inclusion / exclusion of the “Don’t Know” option while designing questionnaire surveys, this controversial option has not been included in this questionnaire as its presence can lead the respondents not to express their true responses, which can result in high influx of spurious responses. The questions in instrument have been appropriately formatted and the font has been selected carefully to maximize the visual clarity.

The second section of questionnaire contains the proforma for the respondents to select the appropriate frequency level of demotivators encountered while scaling up agile methodologies. Likert Scale has been used to rank the frequency levels as Rare, Unlikely, Possible, Likely, Almost Certain [96]. These labels are given numeric values where 1 = Rare, 2 = Unlikely, 3 = Possible, 4 = Likely, 5 = Almost Certain. Similarly, the impact of those demotivators has also been represented by Likert Scale having labels as Incidental = 1, Minor = 2, Moderate = 3, Major = 4, Extreme = 5 [97]. The Likert scale has been balanced by having the equal and opposite options on the two ends of the scale and the interval between the options has been kept equal.

The third section of the questionnaire contains the demotivators and a list of suggested practices corresponding to each demotivator in the form of checkboxes whereby the respondent can select the best practice(s) s/he thinks is best suited to address the respective demotivator. Here, the open-ended option is given to the respondents to allow them to add any other additional guideline(s) / practice(s), not already available in the list, to address the demotivators in their opinion (for complete list of filtered demotivators and suggested practices, please consult Chapter 4). Each section contains the relevant questions logically grouped together and the length of the questionnaire has specifically been considered to allow the respondents to complete it within reasonable time length i.e., 10 to 15 minutes at normal pace. The conduct of the questionnaire also assures the respondents of the complete protection and confidentiality of their identity and their responses in all the circumstances. The responses of questionnaire will only be used for the intended purpose and will not be shared with third parties, either for profit or non-profit. This declaration has shown to enhance the trust and motivation of the survey respondents, enabling them to answer the questions with reasonable judgement and objectivity, thus increasing the positive response rate.

To avoid the willful inclusion of bias in the questionnaire, the questions have been developed in a neutral manner to prevent the inclination towards any specific point of view. To address the concerns of researcher bias and to ensure the observance of content and construct validity, the questionnaire has been passed through a focused group session to identify the inherent faults and drawbacks likely to affect the effectiveness of the proposed questionnaire and purpose of its conduction. Few industry practitioners have contributed their valuable feedback while critically analyzing the questionnaire in the process of content and construct validity of the questionnaire which have led to significant improvements in the designed instrument, making it ready for administration. To measure the internal consistency of the data items of the questionnaire, Cronbach's Alpha Reliability Test has been executed and an overall value of 0.928 of Cronbach's Alpha, based on the recommendations of George & Mallery [98] suggests that the data items are highly correlated and together contribute as a single construct for measuring the responses (for detailed discussion about internal consistency of the questionnaire, please consult Chapter 4).

3.6 Sampling Techniques

The sampling technique refers to the method / approach used for selecting samples from a population [99]. As it is not possible to consider the whole population in the research study, the researcher almost always extracts samples from the population based on different techniques where the sample is said to be the representation of the complete population [100].

Probability sampling is a type of sampling technique where each member of the population has an equal chance of selection in a sample [101]. It is used when the research entails the collection of generic data from general population where inclusion / exclusion criterion is not defined [102]. Probability sampling reduces the researcher's bias in sample selection and also produces more accurate results. However, it cannot be used in domain specific situations where the nature of respondents and inclusion / exclusion criterion needs to be defined [103].

Non-probability sampling techniques reduce the chances of equal selection of various members of the population [101]. The members to be included in the study are selected based on various inclusion / exclusion criteria that are applied to the population to filter out the samples [104]. Non-probability sampling has several types like

- Convenience sampling where the researcher selects samples from the population based on his convenience, ease and judgement [105]. There is no defined criterion in the selection of respondents, it is purely based on the researcher's convenience and ease. This includes researcher bias and the samples selected in most cases do not accurately represent population [106].

- Quota Sampling in which the population is divided into mutually exclusive, distinct groups or categories based on some criteria and then samples are selected from each group based on some inclusion / exclusion criteria [107].

- Judgmental or purposive sampling is a type of non-probability sampling where the researcher intentionally selects members of a population for participation in study based on their ability that fulfil certain criteria [108]. The researcher first filters out the population based on inclusion / exclusion criteria and then selects samples from the selected population for inclusion in research study [101].

- Snowball sampling or referral sampling initially employs few members out of a population fulfilling certain criteria based on their ability to participate efficiently in the research study [109]. The selected members further include other members in the research study which in turn include other members in the study to form a pool of participants [110].

In this study, a combination of purposive and snowball sampling has been employed wherein those respondents have been targeted who have worked / are working on large scale agile software development projects. The selected respondents have recruited other prospective members in the questionnaire survey who possess prior or current experience of working on large-scale agile projects.

3.7 Sample Size

The sample, as the name suggests, is the representation of population [111]. In this study, the population refers to the industry practitioners working on large scale agile software development projects. As it is not possible to consider the whole population for data collection [112], various sampling techniques based on different parameters are utilized to extract samples out of a population [113]. The population size can never be estimated precisely, it is always fixed arbitrarily / hypothetically large enough commensurate with the estimated population [100]. In this study, the population size has been arbitrarily selected as 100,000 industry practitioners who have worked / are working on large scale agile software development projects. The sample size has been calculated through an online website SurveyMonkey [114] which came out to be 138. The confidence interval, usually expressed as a percentage, is a measure of the similarity between the results of repeated data collection procedure from a certain population [115]. Based on the guidelines of U.S. Census Bureau [116], the confidence interval in this study has been taken as 90% which means that if the questionnaire survey is repeating by selecting different samples from the same population, the results obtained would be similar 90% of the time.

3.8 Data Collection Method

The data collection method employed in this study is the Questionnaire Survey which seems to be the most suitable method in this context. Because the data has to be collected from individuals who have prior or current experience of working on large scale agile software development projects, online questionnaire survey has been conducted via Google Forms. The questionnaire has been shared with prospective respondents through internet using snowball approach.

3.9 Respondents’ Profile for Survey

The respondents have been carefully selected who have worked or are currently working on large scale agile software development projects. The responses have been received from respondents with a diverse range of industry experience and have worked in IT industry in different capacities. Moreover, the respondents have been dispersed round the globe and have contributed their valuable opinions while filling out the questionnaire. The demographics of the respondents based on their large-scale agile work experience, designation and their organization size are graphically illustrated by these pie charts.

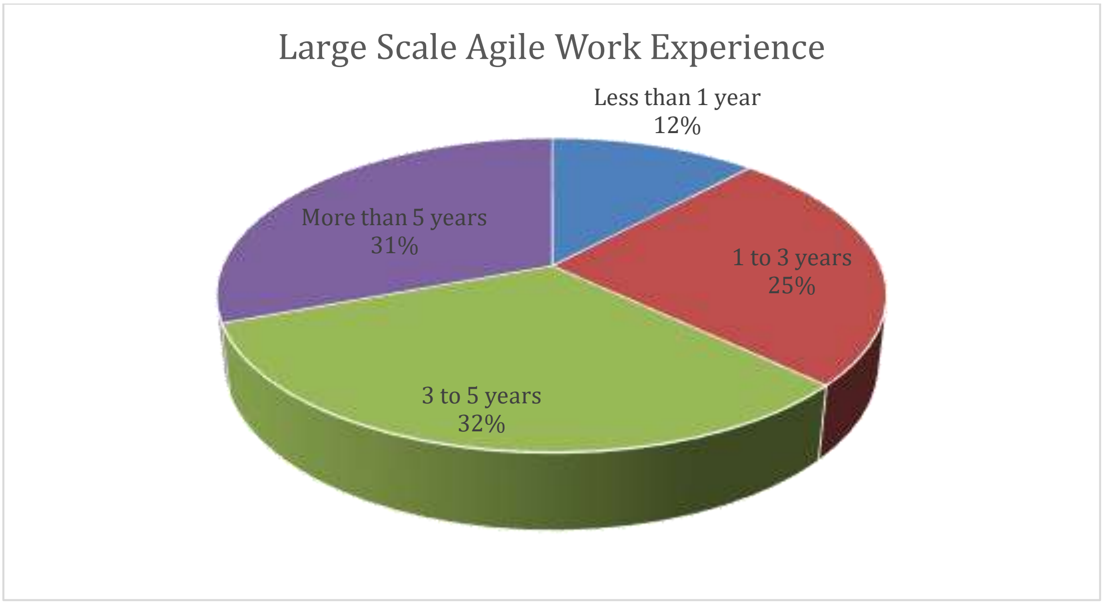

Figure 3.2: Large Scale Agile Work Experience of Survey Respondents

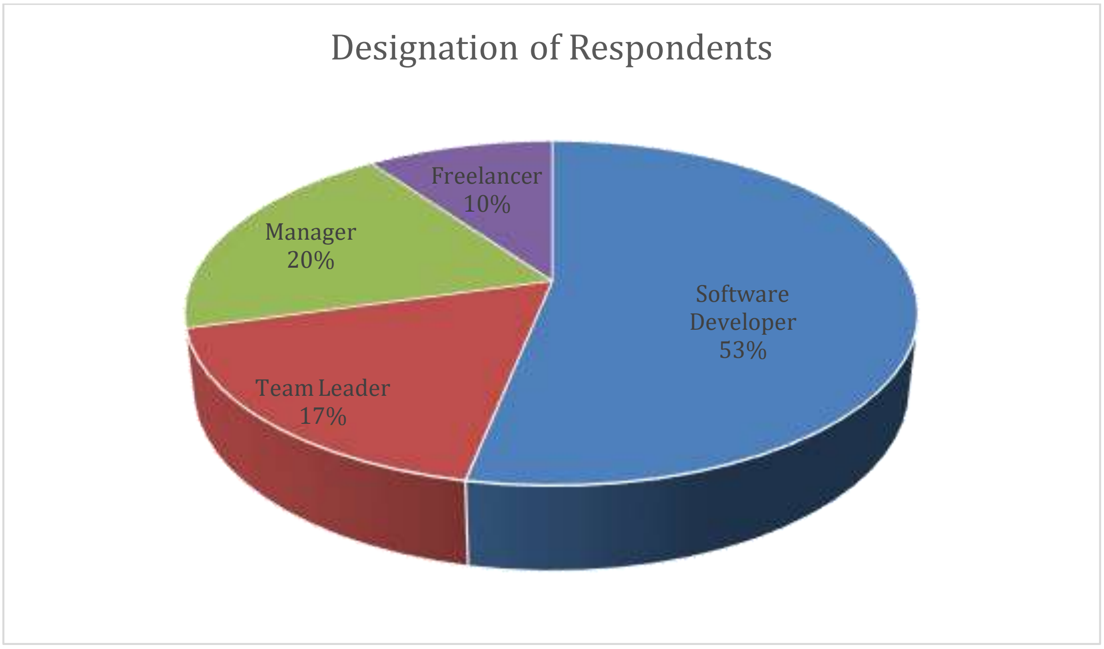

Figure 3.3: Designation of Survey Respondents

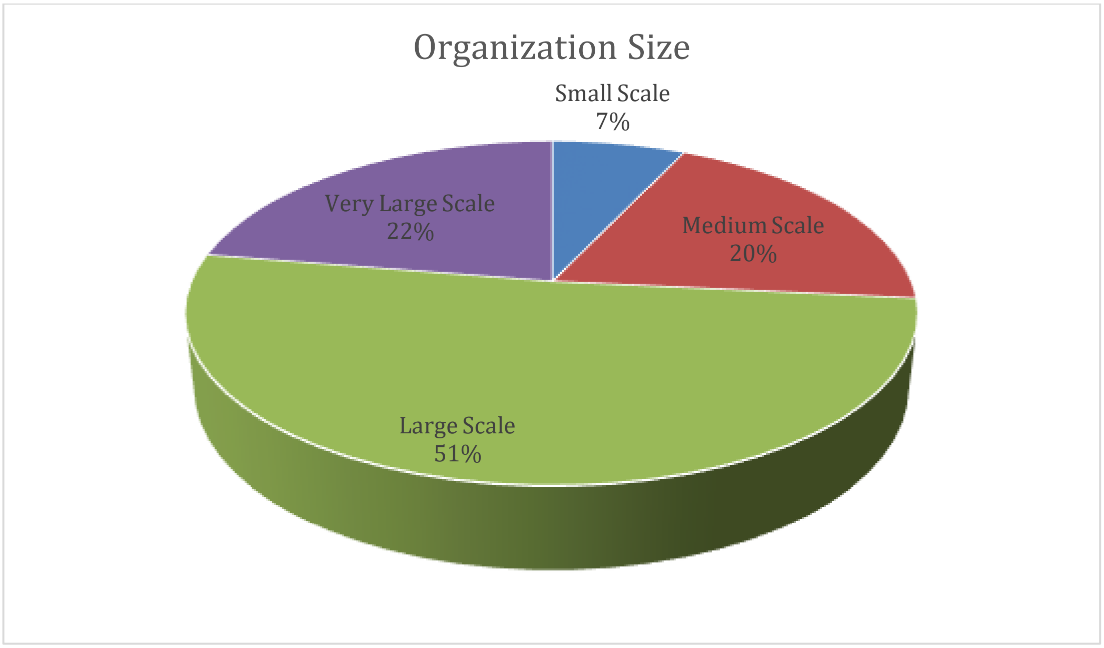

Figure 3.4: Organization Size of Survey Respondents

3.10 Data Analysis

Once the data relevant to the research is collected, it is cleansed to remove the junk data, filtered for irrelevant responses and then the valid responses are subject to statistical treatment to reveal useful information that helps in answering the research questions by drawing conclusions from analyzed data [75]. Data analysis phase employs a complete set of statistical tools, techniques and processes to process and analyze the data [117]. As part of this research study, different statistical methods shall be used to process and analyze the data. Since the data used in this research is numeric in nature, quantitative data analysis shall be performed. A brief description of each statistical tool, to be used in this research, is, however, provided for quick review of the readers.

- Arithmetic mean / average is a statistical tool that computes the average value of a set of numeric data items. It is denoted by $\overline{X}$ and represents the mean / central value of a data set. In this research study, arithmetic mean shall be used for the computation of the average priority of each demotivator as represented by all the survey respondents.

- Cronbach’s Alpha Reliability Test is a statistical tool that is used to evaluate the internal consistency of the data items of a questionnaire. In other words, it computes the numerical value of the reliability of a data collection instrument that denotes the extent to which the data items of a data collection instrument together measure a single construct. Cronbach’s alpha test shall be applied on the questionnaire to assess its overall reliability and internal consistency.

- Pearson’s correlation coefficient measures the degree of strength and direction of association between two data sets. It returns a numeric value ranging from -1 to +1 where transition from -1 to +1 indicates the strong positive relationship between the data sets and transition from -1 to +1 indicates strong negative relationship between the data sets.

3.11 Summary

This chapter has explained in detail the research methodology adopted in this research study. The next chapter explains the results of data analysis phase and discusses the results obtained from data analysis phase with respect to the research questions and objectives.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Overview

In this chapter, the data collected through questionnaire survey has undergone statistical analysis and the results obtained have been explained in detail along with the tests performed. The results of data analysis have been discussed in the light of the research questions and objectives.

4.2 Data Analysis Tools

The data analysis entails the use of various statistical methods for filtering, cleansing, ordering, analyzing, processing, interpreting and presenting data and obtained results. The statistical methods can be quite complex, requiring the use of software tools that provide easy to use interface for performing all steps of data analysis. Software tools provide a pre-built set of mathematical and statistical methods that can be applied to the data to yield the required results. A brief overview of the software tools used for data analysis as part of this research study is given below.

- Microsoft Excel is a leading spreadsheet program by Microsoft Corporation that provides built in tools for data manipulation, analysis and visualization for easy interpretation and presentation. The real strength of Excel lies in the availability of several callable mathematical functions that return the value after computation.

- Statistical Package for Social Sciences (SPSS) is a powerful statistical software suite produced and marketed by IBM®. It provides a powerful yet user friendly environment for performing complex statistical data analysis procedures, data mining algorithms, business data intelligence, data forecasting to support business decision support systems.

In this study, Microsoft Excel Professional Plus 2019 and IBM SPSS Statistics Version 23 software suite have been used for statistical data analysis.

4.3 Preliminary Data Processing

The data obtained from the questionnaire survey has been received in the form of downloadable Microsoft Excel Worksheet which contains the responses of all the respondents where each row represents the responses of a single respondent. The worksheet contains the frequency of occurrence of each demotivator along with its impact as selected by each respondent and the list of best practices corresponding to each demotivator which, in the opinion of the respondent, can address that particular demotivator in an effective manner.

The frequency and impact of each demotivator is represented by its corresponding label and each label in turn denotes a numeric rating as given in the following table.

Table 4.1: Labels of Frequency and Impact of Demotivators

Sr.	Frequency Label	Numeric Rating	Impact Label	Numeric Rating
1	Rare	1	Incidental	1
2	Unlikely	2	Minor	2
3	Possible	3	Moderate	3
4	Likely	4	Major	4
5	Almost Certain	5	Extreme	5

The corresponding frequency and impact labels have been replaced by their corresponding numeric ratings in the Excel response sheet using Find and Replace feature. This step converts the entire data in the Excel sheet into numeric format to enable the application of the mathematical and statistical tests. This sets the stage for the calculation of priority of the demotivators.

4.4 Ranking of Demotivators

In this research study, the priority of the demotivator is defined as the product of frequency of a demotivator and its impact [118]. The higher the priority value of a demotivator, the higher its criticality. The priority of each demotivator is calculated by using the formula **P = FD * ID** {where **P** = Priority, **FD** = Frequency of Demotivator, **ID** = Impact of Demotivator} [118]. The following table provides the theoretical illustration of calculation of priority of demotivators.

Table 4.2: Illustration of Calculation of Priority of Demotivators

Respondent	**FD1**	**FD2**	**ID1**	**ID2**	**P1 = FD1 * ID1**	**P2 = FD2 * ID2**
R1	A	C	B	D	A * B	C * D
R2	P	R	Q	S	P * Q	R * S
RN	W	Y	X	Z	W * X	Y * Z
Sum of Priorities					$\sum_{1}^{N} P1$	$\sum_{1}^{N} P2$
Arithmetic Mean of Priorities of Demotivators					$\bar{X} = \frac{\sum P1}{N}$	$\bar{X} = \frac{\sum P2}{N}$

As depicted in the above table, the priority of a demotivator is computed by calculating the product of its frequency and impact as assigned by each respondent. The sum of the priority of each demotivator, assigned by all the respondents is calculated and the arithmetic mean ($\bar{X}$) is then calculated to gauge the priority of a particular demotivator as represented by all the respondents. In other words, the arithmetic mean ($\bar{X}$) of priority of each demotivator represents the criticality of that particular demotivator according to all the survey respondents in a collective manner. Based on the average priority ($\bar{X}$), the demotivators are listed in the order of decreasing priority whereby the demotivator carrying the highest priority is ranked first. The following table presents the mean average priorities ($\bar{X}$) of the demotivators.

Table 4.3: Priorities of Demotivators from Industry Survey

Sr. No.	**Identifier**	**Demotivators**	**Priority ($\bar{X}$)**
1	D1	Traditional organizational culture	9.84
2	D2	General resistance to change	10
3	D3	Lack of management and commitment support	9.83
4	D4	Lack of agile experts	13.71
5	D5	Reluctance to adopt	10.48
6	D6	Bad customer relationship	9.23
7	D7	Problem in requirement elicitation	10.77
8	D8	Lack of knowledge	12.32
9	D9	Problem of team feedback / interfacing between teams difficult / lack of teamwork	11.83
10	D10	Reduced productivity due to delay	12.43
11	D11	Lack of customer presence	9.18
12	D12	Lack of team training	13.38
13	D13	Lack of effective communication / distributed environment / global distribution challenges	14.08
14	D14	Lack of team orientation	13.55
15	D15	Management unwilling to change	10.36
16	D16	Too high workload and pressure	15.46
17	D17	Misunderstanding agile concepts	13.43

18	D18	Agile customized poorly / misinterpretation of agile concepts	15.34
19	D19	Reverting to the old way of working / management in waterfall model	10.41
20	D20	Using old and new approaches side by side	10.41
21	D21	Creating and estimating user stories hard	9.41
22	D22	Requirement ambiguity affects quality assurance	9.93
23	D23	Lack of proper planning for large scale agile projects	16.24
24	D24	Complexity of large-scale projects	17.68

4.5 Selection of Top 10 Demotivators

It is a common practice in research studies to highlight and emphasize the list of top 10 items that are relatively more important than others [33]. Maruf & Ghazia discuss the list of top 10 software risk factors that occur in the software development process [119], Sommerville & Sawyer highlight the top 10 suggested practices for the requirements engineering phase [120], Taherdoost has explored and presented the top 10 most common causes of project failure in global marketplace [121], Xindong et al, in their study, presented the top 10 algorithms most commonly used in data mining field [122] which shows that the trend of focusing on the top 10 items is quite common. Based on the practice of highlighting top 10 items in research studies, top 10 demotivators have been selected based on their priorities for further discussion and analysis out of the 24 demotivators in this research study.

The following table presents the list of top 10 demotivators extracted from literature that hinder the successful implementation of agile methodologies on large scale projects in the order of decreasing priority / frequency of occurrence in literature studies.

Table 4.4: Top 10 Demotivators Selected from Literature

Sr #	Top 10 Demotivators	Frequency of Occurrence in Literature
1.	Agile Difficult to Implement / Poor Customization of Agile Methodologies	48%
2.	Resistance to Change	38%
3.	Requirements Engineering Challenges	38%
4.	Lack of agile experts	36%
5.	Lack of training / too high workload	31%
6.	Coordination challenges in multi-team environment	31%
7.	Lack of management and commitment support	27%
8.	Traditional organizational culture	22%
9.	Using old and new approaches side by side	21%
10.	Lack of effective communication	21%

4.6 Priorities of Demotivators from Industry Survey

The demotivators obtained from literature survey presented in Table 4.4 have been sorted in descending order in Microsoft Excel based on their priorities. The top 10 demotivators having the highest priorities are listed below, according to the opinion of industry practitioners. The table also presents the ranking of the corresponding demotivators according to the literature citation.

Table 4.5: Top 10 Demotivators Selected from Industry Survey Based on Priorities

Sr #	Identifier	Demotivator	Average Priority	Ranking in Literature
1.	D24	Complexity of Large-Scale Projects	17.68	12
2.	D23	Lack of Proper Planning for Large Scale Agile Projects	16.24	14
3.	D16	Too High Workload and Pressure	15.46	5
4.	D18	Agile Customized Poorly	15.34	1

5.	D13	Lack of Effective Communication	14.08	10
6.	D4	Lack of Agile Experts	13.71	4
7.	D14	Lack of Team Orientation	13.55	6
8.	D17	Misunderstanding Agile Concepts	13.43	11
9.	D12	Lack of Team Training	13.38	5
10.	D10	Reduced Productivity due to Delay	12.43	16

Based on the findings presented in Table 4.5, the results of literature survey are mapped with the results obtained from industry survey. The column "Sr. No." indicates the priority of a demotivator according to the opinion of industry practitioners and the column "Ranking in Literature" presents the corresponding priority of that particular demotivator as found in literature. Here, according to industry survey, complexity of large-scale projects is the most important demotivator carrying the highest priority and is thus the most critical demotivator (occupying 1st Serial No.) whereas in the literature, the same demotivator is ranked at Serial No.12. Similarly, poor customization of agile is ranked as the most critical demotivator (occupying 1st Serial No.) in literature whereas according to the results of survey, it is ranked at Serial No.4. In this way, the relative ranking of demotivators as cited in literature is mapped with the opinion of the industry practitioners. Out of the top 10 demotivators highlighted by both, literature and industry survey, there are 6 common demotivators that have been equally stressed and emphasized by both literature results and software practitioners viz too high workload and pressure, agile customized poorly, lack of effective communication, lack of agile experts, lack of team orientation, lack of team training. These 6 demotivators are even more important due to their presence / criticality in the literature as well as industry survey results.

However, there exist some differences too between the results of literature review and industry survey. Some of the demotivators not labelled as critical by the literature survey are ranked critical by the industry survey. In other words, industry practitioners have ranked a particular demotivator as important / most frequently occurring as opposed to its citation frequency in the literature e.g., complexity of large-scale software projects has been ranked 1st in the list of top 10 demotivators by the industry practitioners whereas the same demotivator is not found in the list of top 10 demotivators extracted from literature survey, rather it is ranked at 12th position. Similarly, resistance to change is the 2nd highest cited demotivator in the literature for scaling agile methodologies whereas it is not reported in the list of top 10 demotivators obtained from industry survey.

4.7 Measurement of Internal Consistency of Questionnaire Instrument

The internal consistency defines the reliability / correlation between the items of a data set in a data collection instrument [123]. The most commonly used and accurate measure of calculation of internal consistency & reliability of data collection instrument is Cronbach's Alpha reliability test whose main aim is to compute the value of internal consistency between the items of a data collection instrument with the higher value of Cronbach's Alpha representing higher reliability of data collection instrument [124]. The results of running Cronbach's Alpha reliability test for measuring the degree of internal consistency and reliability of the designed questionnaire are presented in the following table.

Table 4.6: Results of Cronbach's Alpha Reliability Test

Statistical Test	**Measure of Internal Consistency of Question Set of Frequency of Occurrence of Demotivators (First Subsection)**	**Measure of Internal Consistency of Question Set of Impact of Demotivators (Second Subsection)**	**Combined Measure of Internal Consistency of Both Subsections**
Cronbach's Alpha Reliability Test	0.880	0.875	0.928
No. of Items	24	24	48

The interpretation of results of Table 4.6 based on the reference values range of Cronbach's Alpha Reliability Test (excellent for > 0.9 value, good for > 0.8 value, acceptable for > 0.7 value, questionable for > 0.6 value, poor for > 0.5 value and unacceptable for < 0.5 value) provided by George & Mallery [98] is given below.

- A value of 0.880 measures good correlation and reliability between the data items of first subsection of questionnaire containing the question set of frequency of occurrence of demotivators.

- A value of 0.875 measures good correlation and consistency between the data items of second subsection of questionnaire containing the question set of impact of demotivators.

- The reliability score of 0.928 measures excellent combined correlation and internal consistency between the data items of first and second subsections of the questionnaire.

4.8 Correlation Analysis

Among the various statistical analysis tools, one of them is the correlation analysis. Through correlational analysis, the measure of relationship between two data sets is quantitatively expressed using a correlation coefficient [125]. It is used to determine the strength and direction of the linear association of two data sets [126]. Based on the recommendations of Schober & Patrick [127], the value of correlational coefficient ranges from -1 to +1 which indicates that as the value of correlation coefficient moves towards -1, the strength of the relationship between two variables increases in the negative direction i.e. one variable increases as the other one decreases and as the value of correlation coefficient moves towards +1, the strength of the relationship between two variables increases in the positive direction i.e. one variable increases as the other one increases.

Different types of coefficients are used for the correlational analysis in SPSS out of which Pearson's Coefficient is most commonly used [128]. It quantitatively expresses the direction and relationship of linear association between two data sets [126]. In this study, the correlational analysis test is executed on the two data sets to find out the strength and direction of the relationship between the results obtained from literature survey and the ones obtained from the industry survey. The frequency of occurrence of top 10 demotivators extracted from literature and the top 10 demotivators from industry survey are taken as two data sets for calculation of Pearson's Correlational Coefficient. The results are presented in the following table.

Table 4.7: Results of Pearson's Correlation Test

		Frequency of Occurrence in Literature	Average Priority of Demotivators from Industry Survey
Frequency of Occurrence in Literature	Pearson Correlation	1	0.956
	No. of Items	10	10
Average Priority of Demotivators from Industry Survey	Pearson Correlation	0.956	1
	No. of Items	10	10

By interpreting the results mentioned in table 4.7 according to the guidelines of Schober & Patrick [127], the value of Pearson's Correlation Coefficient of +0.956 suggests a very strong positive correlation between the results of literature survey and the opinion of industry practitioners. There is a high degree of association between the list of top 10 demotivators faced while scaling up agile methodologies as extracted from the literature and the list of top 10 demotivators computed from the statistical analysis of the data obtained from the industry survey of the large-scale agile practitioners.

4.9 Best Practices to Address Demotivators

The frequency of selection of each practice corresponding to each demotivator is calculated by using countif("range of cells", "criteria") function in Microsoft Excel. This function returns the count of "criteria" parameter in the specified "range of cells". The countif() function is executed for all the recommended practices of each demotivator and the practice having the highest frequency / count is attributed to that particular demotivator as the best practice to address it. The percentage of selection of the particular best practice is simply another form of its representation. It is calculated using the following equation.

$$Percentage = \frac{Frequency\ of\ Best\ Practice}{Total\ No.of\ Respondents}\ x\ 100\%$$

The following table shows the demotivator followed by the best practice to address it, the frequency of selection followed by its percentage of respondents who selected this practice as the best practice to address that particular demotivator.

Table 4.8: List of Best Practices to Address Demotivators

Sr #	Identifier	Demotivator	Best Practice	Frequency	Percentage
1.	D24	Complexity of Large-Scale Projects	Formulation of Realistic Work Breakdown Structure	64	43%
2.	D23	Lack of Proper Planning for Large Scale Agile Projects	Follow up Meetings to Discuss Progress	61	40%
3.	D16	Too High Workload and Pressure	Enable Team Members to Follow Their Own Schedule	53	36%

4.	D18	Agile Customized Poorly	Outsource Agile Projects	63	43%
5.	D13	Lack of Effective Communication	Appoint a Project Facilitator to Ensure Coordination Between Teams	55	37%
6.	D4	Lack of Agile Experts	Retention of Seasoned Experts	58	40%
7.	D14	Lack of Team Orientation	Make Team Coordination Top Priority	61	41%
8.	D17	Misunderstanding Agile Concepts	Outsource Agile Projects	63	42%
9.	D12	Lack of Team Training	Induct Experienced Team Members	60	39%
10.	D10	Reduced productivity due to delay	Formulate Realistic Timelines	67	44%

4.10 Discussion

After the statistical processing of the data is completed, the results are obtained which are then discussed and explained to the readers. In this section, the results of the statistical analysis of the data are elaborated in a simpler manner for easy comprehension. The interpreted results are then mapped with the research questions and research findings are justified.

The internal consistency / reliability of the data items of the questionnaire has been evaluated with Cronbach’s Alpha Reliability Test which approximately calculates the value of internal consistency between the items of a data collection instrument, the higher value of Cronbach’s Alpha indicates high reliability of data collection instrument. Based on the recommendations of George & Mallery, the Cronbach’s Alpha for first and second subsections of the questionnaire used in this research study comes out to be 0.880 and 0.875 respectively which indicates good correlation and consistency between the data items of the respective subsections. The overall reliability score of both subsections of the questionnaire is 0.928 which indicates excellent internal consistency of the questionnaire instrument designed in this research study. In other words, the internal consistency score of 0.928 indicates that the different data items of the questionnaire together measure a single construct quite reliably.

After the statistical treatment of data obtained from industrial survey, the top 10 demotivators cited in literature have been juxtaposed with top 10 demotivators in the opinion of industry practitioners and their relative rankings based on their priorities have been compared. Out of the top 10 demotivators list from literature and industry survey, 6 demotivators are reported as crucial by both literature and industry, which establishes strong concurrence between results of the literature survey and industry survey.

The correlational analysis of the two data sets of top 10 demotivators extracted from literature and obtained from industry survey, presented in Table 4.7, has been carried out using the most commonly used correlation coefficient viz Pearson’s Coefficient. Correlational analysis measures the strength of the relationship between two data sets which can be positive or negative. In this research study, the result of the application of Pearson’s Coefficient on the data set of top 10 demotivators extracted from literature and the top 10 demotivators obtained from industry survey, yields a value of +0.956, which when interpreted according to the guidelines of Schober & Patrick, suggests a very strong positive relationship between the results of literature survey and the industry survey. In other words, the opinion of the industry practitioners validates the findings of the literature survey to a great extent.

Table 4.8 displays the list of the top 10 demotivators along with the best practice to address that particular demotivator. The "Demotivator" column contains the demotivator followed by the "Best Practice" which contains the best practice to address that particular demotivator, followed by the "Frequency" column which contains a numeric value that indicates the frequency of selection of the particular best practice in the preceding column by the survey respondents e.g., according to 61 survey respondents, follow up meetings to discuss progress can best address the problem of proper planning for large scale agile projects. Likewise, 55 respondents are of the view that appointing a project facilitator to ensure coordination between teams is the best measure to address and resolve the problem of lack of effective communication.

The last column "Percentage" contains the percentage of the survey respondents who selected that particular best practice the highest number of times. The frequency is converted into percentage form and represented in this column as an alternate form of representation. In this way, each of the top 10 demotivators have their associated best practices along with the frequency of its selection and percentage of the respondents selecting that particular suggested practice.

The following table consolidates the answers to the research questions **R1** viz the demotivators faced by industry practitioners during the large-scale agile development projects and **R2** viz the recommendations to address the identified demotivators for large scale agile development projects. For the sake of simplicity, the top 10 demotivators which are commonly occurring and more important than the other ones, have been selected and presented in the following table. The column "Demotivators" presents the top 10 leading demotivators faced by industry practitioners during large scale agile development projects and the column "Best Practice" presents the recommendation corresponding to that particular demotivator to help address it in the most efficient manner.

Table 4.9: Mapping of Literature Survey Results with Industry Survey

Sr #	Identifier	Demotivator	Best Practice

1.	D24	Complexity of Large-Scale Projects	Formulation of Realistic Work Breakdown Structure
2.	D23	Lack of Proper Planning for Large Scale Agile Projects	Follow up Meetings to Discuss Progress
3.	D16	Too High Workload and Pressure	Enable Team Members to Follow Their Own Schedule
4.	D18	Agile Customized Poorly	Outsource Agile Projects
5.	D13	Lack of Effective Communication	Appoint a Project Facilitator to Ensure Coordination Between Teams
6.	D4	Lack of Agile Experts	Retention of Seasoned Experts
7.	D14	Lack of Team Orientation	Make Team Coordination Top Priority
8.	D17	Misunderstanding Agile Concepts	Outsource Agile Projects
9.	D12	Lack of Team Training	Induct Experienced Team Members
10.	D10	Reduced Productivity due to Delay	Formulate Realistic Timelines

4.11 Summary

In this chapter, the different phases of statistical data analysis have been described along with the detailed discussion of results of the data analysis phase. The next chapter provides the future research directions and some concluding remarks.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Overview

This chapter presents the overall summary of the research contributions made by this study and the future research directions that can be pursued to extend the existing research domain.

5.2 Conclusions

This research study has mainly focused on the initial extraction of demotivators faced while scaling agile methodologies at large scale from literature. The questionnaire survey has been designed and conducted to gather the viewpoint of the industry practitioners regarding the demotivators highlighted in literature. Thereafter, the priority of those demotivators is calculated by considering the frequency of occurrence of those demotivators and their relative impact. The demotivators have been listed in the order of decreasing priority and top 10 demotivators have been filtered out for further discussion/comparison & evaluation. The best practices to address the demotivators have been selected based on the frequency of selection by the respondents with the practice selected the most by the respondents has been attributed to the corresponding demotivator as the best practice to address it. The list of key demotivators and their best practices highlighted by this research study will help out the managers in dealing with the demotivators faced while scaling up agile methodologies which will yield higher success rate of agile methodologies when applied to large scale software development projects.

5.2 Limitations

Limitations are invariably an inherent part of any research work which are to be minimized / suppressed. This research study aims to conduct a questionnaire survey to validate the findings of literature regarding the demotivators faced while scaling up agile methodology. However, certain potential limitations exist in this research study such as

- The responses of the respondents participating in the questionnaire survey might not depict their true opinions.

- There is a need to explore more demotivators faced by industry practitioners while scaling up agile methodology. For this purpose, case study approach can be adopted for deep understanding.

- The literature explored as part of this research study is not exhaustive as it contains the published material in journal articles and conference proceedings in English Language. However, it is quite possible that published material relevant to the domain of this research study is available in other languages too which has been left unexplored due to language constraints.

5.3 Future Work

This research study has specifically considered the demotivators faced while scaling up agile methodologies as found in the literature and conducted a questionnaire survey to gather the opinion of industry practitioners regarding those demotivators. However, there is a need to consider the motivators mentioned in literature for scaling up agile methodologies and conduct a similar survey to consider the viewpoint of industry practitioners regarding the mentioned motivators [129]. The motivators can also be prioritized based on their criticality to help management focus on the most critical ones to successfully scale agile methodologies on large scale [130]. This can provide an ample research direction for researchers in future to extend the existing effort to cover another similar aspect of this study.

5.4 Summary

This chapter has discussed the salient contributions made by this research study. Moreover, the future research directions to extend the existing study have also been indicated for the interested researchers to explore.

REFERENCES

[1] L. M. Maruping and S. Matook, “The evolution of software development orchestration: current state and an agenda for future research,” *Eur. J. Inf. Syst.*, vol. 29, no. 5, pp. 443–457, 2020, doi: 10.1080/0960085X.2020.1831834.

[2] A. Kakar and A. Kakar, “A Brief History of Software Development and Manufacturing,” *SAIS 2020 Proc.*, no. November, 2020, [Online]. Available: https://aisel.aisnet.org/sais2020/4.

[3] J. Yu, “Research Process on Software Development Model,” *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 394, no. 3, 2018, doi: 10.1088/1757-899X/394/3/032045.

[4] R. Sherman, *Project Management. Business Intelligence Guidebook*. 2015.

[5] W. Van Casteren, “The Waterfall Model And The Agile Methodologies : A Comparison By Project Characteristics-Short The Waterfall Model and Agile Methodologies,” *Acad. Competences Bachelor*, no. February, pp. 10–13, 2017, doi: 10.13140/RG.2.2.36825.72805.

[6] M. STOICA, M. MIRCEA, and B. GHILIC-MICU, “Software Development: Agile vs. Traditional,” *Inform. Econ.*, vol. 17, no. 4/2013, pp. 64–76, 2013, doi: 10.12948/issn14531305/17.4.2013.06.

[7] S. T. ind, Karambir, “A Simulation Model for the Spiral Software Development Life Cycle,” *Int. J. Innov. Res. Comput. Commun. Eng.*, vol. 03, no. 05, pp. 3823–3830, 2015, doi: 10.15680/ijircce.2015.0305013.

[8] L. Khoza and C. Marnewick, “Waterfall and agile information system project success rates-a South African perspective,” *South African Comput. J.*, vol. 32, no. 1, pp. 43–73, 2020, doi: 10.18489/sacj.v32i1.683.

[9] J. Johnson and H. Mulder, “MONEY PIT : The True Cost of a Project,” no. January 2015, pp. 1–10, 2015, doi: 10.13140/RG.2.2.16556.62087.

[10] T. Sekgweleo, “Understanding Traditional Systems Development Methodologies,” vol. 4, no. 3, pp. 51–58, 2015.

[11] M. Sameen Mirza and S. Datta, “Strengths and Weakness of Traditional and Agile Processes - A Systematic Review,” *J. Softw.*, vol. 14, no. 5, pp. 209–219, 2019, doi: 10.17706/jsw.14.5.209-219.

[12] G. Papadopoulos, “Moving from Traditional to Agile Software Development Methodologies Also on Large, Distributed Projects.,” *Procedia - Soc. Behav. Sci.*, vol. 175, pp. 455–463, 2015, doi: 10.1016/j.sbspro.2015.01.1223.

[13] A. Aitken and V. Ilango, “A comparative analysis of traditional software engineering and agile software development,” *Proc. Annu. Hawaii Int. Conf. Syst. Sci.*, pp. 4751–4760, 2013, doi: 10.1109/HICSS.2013.31.

[14] P. Gerlero, “Successes and failures in software development project management: A systematic literature review,” *CEUR Workshop Proc.*, vol. 2992, pp. 131–145, 2021.

[15] Y. Lu, L. Luo, H. Wang, Y. Le, and Q. Shi, “Measurement model of project complexity for large-scale projects from task and organization perspective,” *Int. J. Proj. Manag.*, vol. 33, no. 3, pp. 610–622, 2015, doi: 10.1016/j.ijproman.2014.12.005.

[16] J. R. San Cristóbal, L. Carral, E. Diaz, J. A. Fraguela, and G. Iglesias, “Complexity and project management: A general overview,” *Complexity*, vol. 2018, 2018, doi: 10.1155/2018/4891286.

[17] B. Y. Tsai, S. Stobart, N. Parrington, and B. Thompson, “Iterative design and testing within the software development life cycle,” *Softw. Qual. J.*, vol. 6, no. 4, pp. 295–310, 1997, doi: 10.1023/a:1018528506161.

[18] C. Larman and V. R. Basili, “Iterative and incremental development: A brief history,” *Computer (Long. Beach. Calif).*, vol. 36, no. 6, pp. 47–56, 2003, doi: 10.1109/MC.2003.1204375.

[19] Q. Wang and X. Lai, “Requirements management for the incremental development model,” *Proc. - 2nd Asia-Pacific Conf. Qual. Software, APAQS 2001*, pp. 295–301, 2001, doi: 10.1109/APAQS.2001.990034.

[20] B. K. Bittner and I. Spence, *Managing Iterative Software Development Projects Publisher : Addison Wesley Professional Pub Date : June 27 , 2006 Print ISBN-10 : 0-321-26889-X Print ISBN-13 : 978-0-321-26889-1 Pages : 672*. 2006.

[21] P. Hohl *et al.*, “Back to the future: origins and directions of the ‘Agile Manifesto’ – views of the originators,” *J. Softw. Eng. Res. Dev.*, vol. 6, no. 1, 2018, doi: 10.1186/s40411-018-0059-z.

[22] E. M. Schön, M. Escalona, and J. Thomaschewski, “Agile Values and Their Implementation in Practice,” *Int. J. Interact. Multimed. Artif. Intell.*, vol. 3, no. 5, p. 61, 2015, doi: 10.9781/ijimai.2015.358.

[23] R. Kumar, A. Gupta, and H. Singh, “Agile Methodologies: Working Mechanism with Pros and Cons,” *Gian Jyoti E-Journal*, vol. 4, no. 2, pp. 18–27, 2014, [Online]. Available: www.en.wikipedia.org/wiki/File:SDLC_Phases_Related_to_Management_Controls.jpg.

[24] D. S. Nguyen, “Success Factors That Influence Agile Software Development Project Success,” *Am. Sci. Res. J. Eng. Technol. Sci.*, vol. 17, no. 1, pp. 172–222, 2016.

[25] A. M. H. Al-Said Ahmad, “Agile Large-Scale Software Development: Success Factors, Challenges and Solutions,” *i-manager’s J. Softw. Eng.*, vol. 8, no. 3, pp. 1–12, 2014, doi: 10.26634/jse.8.3.2807.

[26] J. Magne, “Do Agile Methods Work for Large Software Projects ?,” pp. 179–190, 2018, doi: 10.1007/978-3-319-91602-6.

[27] K. Dikert, M. Paasivaara, and C. Lassenius, “Challenges and success factors for large-scale agile transformations: A systematic literature review,” *J. Syst. Softw.*, vol. 119, no. June, pp. 87–108, 2016, doi: 10.1016/j.jss.2016.06.013.

[28] M. Faisal Abrar *et al.*, “De-motivators for the adoption of agile methodologies for large-scale software development teams: An SLR from management perspective,” *J. Softw. Evol. Process*, vol. 32, no. 12, pp. 1–20, 2020, doi: 10.1002/smr.2268.

[29] K. Dikert, M. Paasivaara, and C. Lassenius, “Challenges and success factors for large-scale agile transformations: A systematic literature review,” *J. Syst. Softw.*, vol. 119, pp. 87–108, 2016, doi: 10.1016/j.jss.2016.06.013.

[30] M. Kalenda, P. Hyna, and B. Rossi, “Scaling agile in large organizations: Practices, challenges, and success factors,” *J. Softw. Evol. Process*, vol. 30, no. 10, pp. 1–24, 2018, doi: 10.1002/smr.1954.

[31] M. Shameem, R. R. Kumar, M. Nadeem, and A. A. Khan, “Taxonomical classification of barriers for scaling agile methods in global software development environment using fuzzy analytic hierarchy process,” *Appl. Soft Comput. J.*, vol. 90, p. 106122, 2020, doi: 10.1016/j.asoc.2020.106122.

[32] D. Wright, “Best Practices for Large-Scale AgileTransformations,” vol. 1277, no. 800, p. 64, 2018, [Online]. Available: https://scholarsbank.uoregon.edu/xmlui/handle/1794/23896.

[33] J. Iqbal *et al.*, *Requirements engineering issues causing software development*

outsourcing failure, vol. 15, no. 4. 2020.

[34] O. Kaiwartya *et al.*, “Internet of Vehicles: Motivation, Layered Architecture, Network Model, Challenges, and Future Aspects,” *IEEE Access*, vol. 4, pp. 5356–5373, 2016, doi: 10.1109/ACCESS.2016.2603219.

[35] I. M. Del Águila, J. Palma, and S. Túnez, “Milestones in software engineering and knowledge engineering history: A comparative review,” *Sci. World J.*, vol. 2014, 2014, doi: 10.1155/2014/692510.

[36] B. John N and B. Randell, “Software Engineering Techniques: Report on a Conference Sponsored by the NATO Science Committee.,” no. April, p. 16, 1970.

[37] B. Randell, “Fifty Years of Software Engineering - or - The View from Garmisch,” no. May, pp. 1–9, 2018, [Online]. Available: http://arxiv.org/abs/1805.02742.

[38] M. Anjum and D. Budgen, *An investigation of modelling and design for software service applications*, vol. 12, no. 5. 2017.

[39] J. D. Morgan, “Applying 1970 waterfall lessons learned within today’s agile development process,” *PM World J.*, vol. VII, no. Vii, pp. 1–19, 2018, [Online]. Available: www.pmworldlibrary.net.

[40] H. K. Aroral, “Waterfall Process Operations in the Fast-paced World: Project Management Exploratory Analysis,” *Int. J. Appl. Bus. Manag. Stud.*, vol. 6, no. 1, pp. 91–99, 2021, [Online]. Available: http://www.ijabms.com/wp-content/uploads/2021/05/05_ARORAL_PB.pdf.

[41] S. Nunez, M. Kabalan, P. Singh, and V. Moncada, “The Waterfall Model in Large-Scale Development,” *2015 IEEE Canada Int. Humanit. Technol. Conf. IHTC 2015*, pp. 386–400, 2015, doi: 10.1109/IHTC.2015.7238067.

[42] D. A. Chart, B. D. Swanson, T. Knight, and J. A. Nido, “The software development process – why it has failed us and a different approach going forward,” *AIAA Scitech 2021 Forum*, no. January, pp. 1–12, 2021, doi: 10.2514/6.2021-1916.

[43] R. Pellerin, N. Perrier, X. Guillot, and P.-M. Léger, “Project Management Software Utilization and Project Performance,” *Procedia Technol.*, vol. 9, pp. 857–866, 2013, doi: 10.1016/j.protcy.2013.12.095.

[44] G. L. Rexing, “Software Project Management: Moving Beyond Project Plans,” *AT&T Tech. J.*, vol. 70, no. 2, pp. 40–48, 2008, doi: 10.1002/j.1538-

7305.1991.tb00344.x.

[45] R. Dwivedula and N. Bolloju, “Transitioning from plan-driven methods to agile methods - Preparation for a systematic literature review,” *Proc. 5th Int. Conf. Commun. Electron. Syst. ICCES 2020*, no. Icces, pp. 944–950, 2020, doi: 10.1109/ICCES48766.2020.09137917.

[46] K. A. O. Al-husseini and A. H. Obaid, “Usage of Prototyping in Software Testing,” *Multi-Knowledge Electron. Compr. J. Educ. Sci. Publ.*, no. November, 2018.

[47] R. Nacheva, “Prototyping Approach in User Interface,” *2Nd Conf. Innov. Teach. Methods*, no. June, pp. 80–87, 2017, [Online]. Available: https://www.researchgate.net/publication/317414969.

[48] C. W. Elverum, T. Welo, and S. Tronvoll, “Prototyping in New Product Development: Strategy Considerations,” *Procedia CIRP*, vol. 50, pp. 117–122, 2016, doi: 10.1016/j.procir.2016.05.010.

[49] A. Susanto and Meiryani, “System Development Method with The Prototype Method,” *Int. J. Sci. Technol. Res.*, vol. 8, no. 7, pp. 141–144, 2019.

[50] E. J. Christie *et al.*, “Prototyping strategies: Literature review and identification of critical variables,” *ASEE Annu. Conf. Expo. Conf. Proc.*, no. November 2018, 2015, doi: 10.18260/1-2--21848.

[51] T. Dingsøyr and N. B. Moe, “Research challenges in large-scale agile software development,” *ACM SIGSOFT Softw. Eng. Notes*, vol. 38, no. 5, pp. 38–39, 2013, doi: 10.1145/2507288.2507322.

[52] A. Srivastava, D. Mehrotra, P. K. Kapur, and A. G. Aggarwal, “Analytical evaluation of agile success factors influencing quality in software industry,” *Int. J. Syst. Assur. Eng. Manag.*, vol. 11, pp. 247–257, 2020, doi: 10.1007/s13198-020-00966-z.

[53] K. Petersen and C. Wohlin, “The effect of moving from a plan-driven to an incremental software development approach with agile practices: An industrial case study,” *Empir. Softw. Eng.*, vol. 15, no. 6, pp. 654–693, 2010, doi: 10.1007/s10664-010-9136-6.

[54] E. Altameem, “Impact of Agile Methodology on Software Development,” *Comput. Inf. Sci.*, vol. 8, no. 2, 2015, doi: 10.5539/cis.v8n2p9.

[55] M. Špundak, “Mixed Agile/Traditional Project Management Methodology – Reality or Illusion?,” *Procedia - Soc. Behav. Sci.*, vol. 119, pp. 939–948, 2014,

doi: 10.1016/j.sbspro.2014.03.105.

[56] H. Salameh, “What, When, Why, and How? A Comparison between Agile Project Management and Traditional Project Management Methods,” *Int. J. Bus. Manag. Rev.*, vol. 2, no. 5, pp. 52–74, 2014, [Online]. Available: http://www.eajournals.org/wp-content/uploads/What-When-Why-and-How-A-Comparison-between-Agile-Project-Management-and-Traditional-Project-Management-Methods.pdf.

[57] U. Muhammad *et al.*, “Impact of agile management on project performance: Evidence from I.T sector of Pakistan,” *PLoS One*, vol. 16, no. 4 April 2021, pp. 1–24, 2021, doi: 10.1371/journal.pone.0249311.

[58] M. Jørgensen and K. Moløkken, “How Large Are Software Cost Overruns ? A Review of the 1994 CHAOS Report 3 A Comparison with Other Cost Estimation Accuracy,” *Inf. Softw. Technol.*, vol. 48, no. 4, pp. 297–301, 2006.

[59] C. Ebert and M. Paasivaara, “Scaling Agile,” *IEEE Softw.*, vol. 34, no. 6, pp. 98–103, 2017, doi: 10.1109/MS.2017.4121226.

[60] M. Alqudah and R. Razali, “A review of scaling agile methods in large software development,” *Int. J. Adv. Sci. Eng. Inf. Technol.*, vol. 6, no. 6, pp. 828–837, 2016, doi: 10.18517/ijaseit.6.6.1374.

[61] F. Almeida and E. Espinheira, “Large-Scale Agile Frameworks: A Comparative Review,” *J. Appl. Sci. Manag. Eng. Technol.*, vol. 2, no. 1, pp. 16–29, 2021, doi: 10.31284/j.jasmet.2021.v2i1.1832.

[62] VersionOne, “COLLAB.NET VERSIONONE.COM #StateOfAgile,” *12. Annu. State Agil. Rep.*, 2018, [Online]. Available: https://www.versionone.com/about/press-releases/12th-annual-state-of-agile-survey-open/.

[63] G. Gruver and T. Mouser, “Leading the Transformation: Applying Agile and DevOps Principles at Scale,” *アエラ*, vol. 13, no. 29, p. 107, 2015, [Online]. Available: http://ci.nii.ac.jp/naid/40004728554/.

[64] A. Putta, M. Paasivaara, and C. Lassenius, *Benefits and challenges of adopting the Scaled Agile Framework (SAFe): Preliminary results from a multivocal literature review*, vol. 11271 LNCS, no. January. Springer International Publishing, 2018.

[65] K. Conboy and N. Carroll, “Implementing Large-Scale Agile Frameworks:

Challenges and Recommendations,” *IEEE Softw.*, vol. 36, no. 2, pp. 44–50, 2019, doi: 10.1109/MS.2018.2884865.

[66] B. Hobbs and Y. Petit, “Agile Methods on Large Projects in Large Organizations,” *Proj. Manag. J.*, vol. 48, no. 3, pp. 3–19, 2017, doi: 10.1177/875697281704800301.

[67] N. B. Moe and M. Mikalsen, *Large-Scale Agile Transformation: A Case Study of Transforming Business, Development and Operations*, vol. 383 LNBIP. Springer International Publishing, 2020.

[68] S. A. K. Ghayyur, S. Ahmed, S. Ullah, and W. Ahmed, “The impact of motivator and demotivator factors on agile software development. The case of Pakistan,” *Int. J. Adv. Comput. Sci. Appl.*, vol. 9, no. 7, pp. 80–93, 2018, doi: 10.14569/IJACSA.2018.090712.

[69] M. Faisal, S. Rehman, N. Rashid, and S. Ali, “Large Scale Agile Adoption Model from Management Perspective,” *Int. J. Comput. Appl.*, vol. 152, no. 2, pp. 31–35, 2016, doi: 10.5120/ijca2016911783.

[70] H. Tobi and J. K. Kampen, “Research design: the methodology for interdisciplinary research framework,” *Qual. Quant.*, vol. 52, no. 3, pp. 1209–1225, 2018, doi: 10.1007/s11135-017-0513-8.

[71] C. Igwenagu, “Fundamentals of Research Methodology and Data Collection,” *L. Lambert Acad. Publ.*, no. June, p. 4, 2016, [Online]. Available: https://www.researchgate.net/publication/303381524_Fundamentals_of_research_methodology_and_data_collection.

[72] D. Cvetkovic and B. Medic, “Research methodology in the 21st Century,” in *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2017, pp. 891–894, doi: 10.23919/MIPRO.2017.7973548.

[73] S. Jamshed, “Qualitative research method-interviewing and observation,” *J. Basic Clin. Pharm.*, vol. 5, no. 4, p. 87, 2014, doi: 10.4103/0976-0105.141942.

[74] Ponto J, “Understanding and Evaluating Survey Research,” *J. Adv. Pract. Oncol.*, vol. 6, no. 2, pp. 168–171, 2015, [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4601897/pdf/jadp-06-168.pdf.

[75] S. Vemuri, J. Hynson, L. Gillam, and K. Williams, “Simulation-Based Research: A Scoping Review,” *Qual. Health Res.*, vol. 30, no. 14, pp. 2351–

2360, 2020, doi: 10.1177/1049732320946893.

[76] D. M. Mertens, “Mixed methods,” *Rev. Qual. Res. Soc. Sci.*, pp. 139–150, 2013, doi: 10.4324/9780203813324-11.

[77] A. Shorten and J. Smith, “Mixed methods research: Expanding the evidence base,” *Evid. Based. Nurs.*, vol. 20, no. 3, pp. 74–75, 2017, doi: 10.1136/eb-2017-102699.

[78] M. Patel and N. Patel, “Exploring Research Methodology : Review Article,” *Int. J. Res. Rev.*, vol. 6, no. 3, pp. 48–55, 2019.

[79] O. D. Apuke, “Quantitative Research Methods : A Synopsis Approach,” *Kuwait Chapter Arab. J. Bus. Manag. Rev.*, vol. 6, no. 11, pp. 40–47, 2017, doi: 10.12816/0040336.

[80] K. L. Wester, L. D. Borders, S. Boul, and E. Horton, “Research quality: Critique of quantitative articles in the journal of counseling & development,” *J. Couns. Dev.*, vol. 91, no. 3, pp. 280–290, 2013, doi: 10.1002/j.1556-6676.2013.00096.x.

[81] A. Babu, A. Maiya, P. Shah, and S. Veluswamy, “Clinical trial registration in physiotherapy research,” *Perspect. Clin. Res.*, vol. 4, no. 3, p. 191, 2013, doi: 10.4103/2229-3485.115387.

[82] P. Aspers and U. Corte, “What is Qualitative in Research,” *Qual. Sociol.*, vol. 44, no. 4, pp. 599–608, 2021, doi: 10.1007/s11133-021-09497-w.

[83] L. Longo, “Empowering Qualitative Research Methods in Education with Artificial Intelligence,” *Adv. Intell. Syst. Comput.*, vol. 1068, no. December 2019, pp. 1–21, 2020, doi: 10.1007/978-3-030-31787-4_1.

[84] E. DePoy and L. N. Gitlin, “Mixed Method Designs,” *Introd. to Res.*, pp. 173–179, 2016, doi: 10.1016/b978-0-323-26171-5.00012-4.

[85] T. L. Jones, M. Baxter, and V. Khanduja, “A quick guide to survey research,” *Ann. R. Coll. Surg. Engl.*, vol. 95, no. 1, pp. 5–7, 2013, doi: 10.1308/003588413X13511609956372.

[86] K. Kelley, B. Clark, V. Brown, and J. Sitzia, “Good practice in the conduct and reporting of survey research,” *Int. J. Qual. Heal. Care*, vol. 15, no. 3, pp. 261–266, 2003, doi: 10.1093/intqhc/mzg031.

[87] G. Szolnoki and D. Hoffmann, “Online, face-to-face and telephone surveys - Comparing different sampling methods in wine consumer research,” *Wine Econ. Policy*, vol. 2, no. 2, pp. 57–66, 2013, doi: 10.1016/j.wep.2013.10.001.

[88] N. Michaelidou and S. Dibb, “Using email questionnaires for research: Good practice in tackling non-response,” *J. Targeting, Meas. Anal. Mark.*, vol. 14, no. 4, pp. 289–296, 2006, doi: 10.1057/palgrave.jt.5740189.

[89] B. Duffy, K. Smith, G. Terhanian, and J. Bremer, “Comparing data from online and face-to-face surveys,” *Int. J. Mark. Res.*, vol. 47, no. 6, pp. 615–630, 2005, doi: 10.1177/147078530504700602.

[90] H. L. Ball, “Conducting Online Surveys,” *J. Hum. Lact.*, vol. 35, no. 3, pp. 413–417, 2019, doi: 10.1177/0890334419848734.

[91] J. F. Ebert, L. Huibers, B. Christensen, and M. B. Christensen, “Paper- or Web-Based Questionnaire Invitations as a Method for Data Collection: Cross-Sectional Comparative Study of Differences in Response Rate, Completeness of Data, and Financial Cost,” *J Med Internet Res*, vol. 20, no. 1, p. e24, 2018, doi: 10.2196/jmir.8353.

[92] S. Roopa and M. Rani, “Questionnaire Designing for a Survey,” *J. Indian Orthod. Soc.*, vol. 46, no. 4_suppl1, pp. 273–277, 2012, doi: 10.1177/0974909820120509s.

[93] M. R. Hyman and J. J. Sierra, “Open- versus close-ended survey questions,” no. February, 2016.

[94] S. C. Desai, “Comparing the use of open and closed questions for Web-based measures of the continued-influence effect,” 2018.

[95] B. A. Kitchenham and S. L. Pfleeger, “Personal opinion surveys,” *Guid. to Adv. Empir. Softw. Eng.*, pp. 63–92, 2008, doi: 10.1007/978-1-84800-044-5_3.

[96] I. Science, “Keywords: Fuzzy-Likert Scale, Perceived Risk, Social Research, Structured Questionnaire. *,” vol. 6, no. 2, pp. 138–150, 2020.

[97] N. J. Duijm, “Recommendations on the use and design of risk matrices,” *Saf. Sci.*, vol. 76, no. July 2015, pp. 21–31, 2015, doi: 10.1016/j.ssci.2015.02.014.

[98] D. George and P. Mallery, “SPSS for Windows step by step: A simple guide and reference. 11.0 update,” 2003.

[99] A. Casteel and N. L. Bridier, “Describing populations and samples in doctoral student research,” *Int. J. Dr. Stud.*, vol. 16, pp. 339–362, 2021, doi: 10.28945/4766.

[100] A. Delİce, “The sampling issues in quantitative research,” *Educ. Sci. Theory Pract.*, vol. 10, no. 4, pp. 2001–2019, 2001.

[101] H. Ames, C. Glenton, and S. Lewin, “Purposive sampling in a qualitative

evidence synthesis: A worked example from a synthesis on parental perceptions of vaccination communication," *BMC Med. Res. Methodol.*, vol. 19, no. 1, pp. 1–9, 2019, doi: 10.1186/s12874-019-0665-4.

[102] N. Hospital, "Probability Sampling - A Guideline for Quantitative Health Care Research," *Ann. African Surg.*, vol. 12, no. 2, pp. 95–99, 2015.

[103] E. Tipton, D. S. Yeager, R. Iachan, and B. Schneider, "Designing probability samples to study treatment effect heterogeneity," *Exp. Methods Surv. Res. Tech. that Comb. Random Sampl. with Random Assign.*, vol. 5, pp. 435–456, 2019, doi: 10.1002/9781119083771.ch22.

[104] P. Lavrakas, "Nonprobability Sampling," *Encycl. Surv. Res. Methods*, no. 2004, p. 2010, 2013, doi: 10.4135/9781412963947.n337.

[105] T. W. Edgar and D. O. Manz, *Exploratory Study*. 2017.

[106] I. Etikan, "Comparison of Convenience Sampling and Purposive Sampling," *Am. J. Theor. Appl. Stat.*, vol. 5, no. 1, p. 1, 2016, doi: 10.11648/j.ajtas.20160501.11.

[107] D. Rukmana, "Quota Sampling," in *Encyclopedia of Quality of Life and Well-Being Research*, A. C. Michalos, Ed. Dordrecht: Springer Netherlands, 2014, pp. 5382–5384.

[108] R. S. Robinson, "Purposive Sampling," in *Encyclopedia of Quality of Life and Well-Being Research*, A. C. Michalos, Ed. Dordrecht: Springer Netherlands, 2014, pp. 5243–5245.

[109] B. B. Frey, "The SAGE Encyclopedia of Educational Research, Measurement, and Evaluation." Thousand Oaks,, California, 2018, doi: 10.4135/9781506326139 NV - 4.

[110] J. Kirchherr and K. Charles, "Enhancing the sample diversity of snowball samples: Recommendations from a research project on anti-dam movements in Southeast Asia," *PLoS One*, vol. 13, no. 8, pp. 1–17, 2018, doi: 10.1371/journal.pone.0201710.

[111] D. P. Turner, "Sampling Methods in Research Design," *Headache*, vol. 60, no. 1, pp. 8–12, 2020, doi: 10.1111/head.13707.

[112] R. G. Brereton, "Populations and samples," *J. Chemom.*, vol. 29, no. 6, pp. 325–328, 2015, doi: 10.1002/cem.2695.

[113] H. Taherdoost, "Sampling Methods in Research Methodology ; How to Choose a Sampling Technique for Research Hamed Taherdoost To cite this version :

HAL Id : hal-02546796 Sampling Methods in Research Methodology ; How to Choose a Sampling Technique for," *Int. J. Acad. Res. Manag.*, vol. 5, no. 2, pp. 18–27, 2016.

[114] "No Title." .

[115] J. Zhang, B. Hanik, and B. Chaney, "Confidence Intervals: Evaluating and Facilitating Their Use in Health Education Research.," *Heal. Educ.*, vol. 40, no. 1, pp. 29–36, 2008.

[116] U. S. C. Bureau, "https://www.census.gov/programs-surveys/saipe/guidance/confidence-intervals.html." .

[117] K. Singh and R. Wajgi, "Data analysis and visualization of sales data," *IEEE WCTFTR 2016 - Proc. 2016 World Conf. Futur. Trends Res. Innov. Soc. Welf.*, 2016, doi: 10.1109/STARTUP.2016.7583967.

[118] T. Aven, "Risk assessment and risk management: Review of recent advances on their foundation," *Eur. J. Oper. Res.*, vol. 253, no. 1, pp. 1–13, 2016, doi: https://doi.org/10.1016/j.ejor.2015.12.023.

[119] M. Pasha, G. Qaiser, and U. Pasha, "A critical analysis of software risk management techniques in large scale systems," *IEEE Access*, vol. 6, pp. 12412–12424, 2018, doi: 10.1109/ACCESS.2018.2805862.

[120] U. Faber, "Requirements Engineering A Good Practice Guide," 2016.

[121] H. Taherdoost and A. Keshavarzsaleh, "Critical Factors that Lead to Projects' Success/Failure in Global Marketplace," *Procedia Technol.*, vol. 22, pp. 1066–1075, 2016, doi: 10.1016/j.protcy.2016.01.151.

[122] X. Wu *et al.*, "Top 10 algorithms in data mining," *Knowl. Inf. Syst.*, vol. 14, no. 1, pp. 1–37, 2008, doi: 10.1007/s10115-007-0114-2.

[123] M. D. Miller, "Classical test theory reliability," *Int. Encycl. Educ.*, pp. 27–30, 2010, doi: 10.1016/B978-0-08-044894-7.00235-9.

[124] S. K. Wadkar, K. Singh, R. Chakravarty, and S. D. Argade, "Assessing the Reliability of Attitude Scale by Cronbach's Alpha," *J. Glob. Commun.*, vol. 9, no. 2, p. 113, 2016, doi: 10.5958/0976-2442.2016.00019.7.

[125] R. K. Prematunga, "Correlational analysis," *Aust. Crit. Care*, vol. 25, no. 3, pp. 195–199, 2012, doi: 10.1016/j.aucc.2012.02.003.

[126] R. J. Janse *et al.*, "Conducting correlation analysis: important limitations and pitfalls," *Clin. Kidney J.*, vol. 14, no. 11, pp. 2332–2337, 2021, doi: 10.1093/ckj/sfab085.

[127] P. Schober and L. A. Schwarte, “Correlation coefficients: Appropriate use and interpretation,” *Anesth. Analg.*, vol. 126, no. 5, pp. 1763–1768, 2018, doi: 10.1213/ANE.0000000000002864.

[128] “Interpreation of the Correlation Coefficients A Basic Review.pdf.” .

[129] M. F. Abrar, M. S. Khan, S. Ali, and N. Rasheed, “Motivators for Large-Scale Agile Adoption From Management Perspective : A Systematic Literature Review,” *IEEE Access*, vol. 7, pp. 22660–22674, 2019, doi: 10.1109/ACCESS.2019.2896212.

[130] M. F. Abrar *et al.*, “Motivators for Large-Scale Agile Adoption from Management Perspective: A Systematic Literature Review,” *IEEE Access*, vol. 7, pp. 22660–22674, 2019, doi: 10.1109/ACCESS.2019.2896212.

APPENDIX A

An Industry Survey of Demotivators for Scaling up Agile Methodology

This questionnaire survey is being conducted as part of MS Software Engineering Thesis Research program. The data collected shall solely be used for the purpose it is intended and shall not, in any way, be shared with third parties. The confidentiality of the respondents and their opinions shall be protected by all means.

Please take into account that this questionnaire pertains to those individuals who have worked on large scale agile software development projects. If you are not one of those, please ignore this questionnaire.

This questionnaire contains 3 sections which shall take approximately 10 to 15 minutes to complete. Your responses will make a huge contribution in my research study and I shall ever be indebted to your support for filling out this form.

Demotivators are the factors that hinder the successful implementation of agile methodologies on large scale software development projects. This survey is being conducted for getting the opinion of industry practitioners regarding the demotivators faced while scaling up agile methodologies.

Name

Your answer

Large Scale Agile Work Experience *

Less than 1 year
1 to 3 years
3 to 5 years
More than 5 years

Total Work Experience in Industry *

Less than 1 year
1 to 3 years
3 to 5 years
More than 5 years

Designation *

Software Developer
Team Leader
Manager
Other:

Organization Name *

Your answer

Organization Size *

Small Scale
Medium Scale
Large Scale
Very Large Scale

List of Demotivators for Scaling Up Agile Methodologies

The first table contains demotivators and the frequency of occurrence of each demotivator in large scale agile software projects is provided based on 5 options. Please select the relevant option that you think is the most relevant in each case. The options are described as below.

1. Rare = (< 10%)
2. Unlikely = (10% - 35%)
3. Possible = (35% - 65%)
4. Likely = (65% - 90%)
5. Almost Certain = (> 90%)

The second table contains demotivators and the relevant impact of each demotivator in large scale agile software projects. Please select the relevant option that you think is the most relevant in each case.

Demotivators & Their Frequency of Occurrence *

	Rare	Unlikely	Possible	Likely	Almost Certain
Traditional Organizational Culture					
General Resistance to Change					
Lack of Management and Commitment Support					
Lack of Agile Experts					
Reluctance to Adopt					
Bad Customer Relationship					
Problem in Requirement Elicitation					
Lack of Knowledge					
Problem of Team Feedback					
Reduced Productivity due to Delay					
Lack of Customer Presence					
Lack of Team Training					
Lack of Effective Communication					
Lack of Team Orientation					
Management Unwilling to Change					
Too High Workload and Pressure					
Misunderstanding Agile Concepts					
Agile Customized Poorly					
Reverting to the Old Way of Working					

Using Old and New Approaches Side by Side
Creating and Estimating User Stories Hard
Requirements Ambiguity Affects Quality Assurance
Lack of Proper Planning for Large Scale Agile Projects
Complexity of Large Scale Projects

Demotivators & Their Impact *

Incidental
Minor
Moderate
Major
Extreme

Traditional Organizational Culture
General Resistance to Change
Lack of Management and Commitment Support
Lack of Agile Experts
Reluctance to Adopt
Bad Customer Relationship
Problem in Requirement Elicitation
Lack of Knowledge
Problem of Team Feedback
Reduced Productivity due to Delay
Lack of Customer Presence
Lack of Team Training
Lack of Effective Communication
Lack of Team Orientation
Management Unwilling to Change
Too High Workload and Pressure
Misunderstanding Agile Concepts
Agile Customized Poorly
Reverting to the Old Way of Working
Using Old and New Approaches Side by Side
Creating and Estimating User Stories Hard
Requirements Ambiguity Affects Quality Assurance
Lack of Proper Planning for Large Scale Agile Projects
Complexity of Large-Scale Projects

Demotivators and Practices to Address Them

The demotivators are listed along with the practices to address them as found in the literature. Please select the practices which you think can effectively address the respective demotivators. If you need to suggest any other practice not listed here, please feel free to do so.

Traditional Organizational Culture *
Reduce Bureaucracy

Self-Autonomous Teams
Independent Team Leaders
Other:

General Resistance to Change *
Innovative Thinking Culture
Flexible Development Approach
Change Embracing Attitude
Other:

Lack of Management and Commitment Support *
User Centric Approach
Self-Autonomous Teams
Leadership Change
Other:

Lack of Agile Experts *
Investment in Human Resource
Hiring Agile Experts
Retention of Seasoned Experts
Other:

Reluctance to Adopt *
Change Embracing Attitude
Flexible Development Approach
Promotion of Innovative Thinking Culture
Other:

Bad Customer Relationship *
Consider Customer as a Necessary Stakeholder
Active Involvement of Customer Throughout the Project
Enhance Customer's Confidence
Other:

Problem in Requirement Elicitation *
Devote Sufficient Time to Requirement Elicitation Phase
Complete and Correct Identification of Real Stakeholders
Utilization of all Possible Requirement Elicitation Techniques
Other:

Lack of Knowledge *
On-Job Training for Employees
Inter & Intra Team Knowledge Sharing
Arranging Formal & Informal Training Sessions
Other:

Problem of Team Feedback *
Strong Cohesion of Teams
Keep Everyone on Board
Encourage Shuffling of Team Members

Other:

Reduced Productivity Due to Delay *
Discourage Heavy Upfront Planning
Plan as you go Approach
Formulate Realistic Timelines
Other:

Lack of Customer Presence *
Encourage Customer Involvement Throughout the Project
Increase Customer's Confidence in Project's Ongoings
Ensure Virtual, if not Physical, Presence of Customer
Other:

Lack of Team Training *
On-Job Training for Team Members
Formal / Informal Training Sessions by Team Leaders
Induct Experienced Team Members
Other:

Lack of Effective Communication *
Formulate Proper Communication Mechanism
Ensure Optimum Communication and Knowledge Sharing Between Distributed Teams
Appoint a Project Facilitator to Ensure Coordination Between Teams
Other:

Lack of Team Orientation *
Organize Teams with Low Coupling and High Cohesion
Non-Overlapping Teams with Defined Roles
Make Team Coordination Top Priority
Other:

Management Unwilling to Change *
Mindset Towards Adoption of New Technologies
Leadership Change
Transfer of Powers from Management to Team Leaders
Other:

Too High Workload and Pressure *
Formulate Realistic Timelines
Ensure Strict Adherance to the Timelines
Enable Team Members to Follow Their Own Schedule
Routine Progress Feedback to Point Out Slacks
Other:

Misunderstanding Agile Concepts *
Organize Team Training Sessions
Hands on Experience on New Technologies
Outsource Agile Projects
Other:

Agile Customized Poorly *
Proper Implementation of Agile Concepts
Implement Proven & Tested Agile Scaling Frameworks
Outsource Agile Projects
Other:

Reverting to the Old Way of Working *
Invest in New Technologies
Develop Forward Advancing Attitude
Reduce Bureaucracy
Maintain Self-Autonomous Teams
Other:

Using Old and New Approaches Side by Side *
Adopt a Single Approach Organization Wide
Maintain Consistency in Work Practices
Adopt a Hybrid Strategy
Other:

Creating and Estimating User Stories Hard *
Realistic Work Breakdown Structure
Utilization of Proper Requirement Elicitation Technique
Realistic Requirement Inclusion in Elicitation Process
Other:

Requirement Ambiguity Affects Quality Assurance *
Ensure Adherence to Formal Elicitation Techniques
Manage Conflicting Requirements
Complete Involvement of all Stakeholders in Requirements Elicitation Process
Integrate Quality Assurance Activities in Each Phase
Other:

Lack of Proper Planning for Large Scale Agile Projects *
Employ Proven Project Management Techniques
Formulate Realistic Timelines
Follow up Meetings to Discuss Progress
Maintain Flexible Timelines
Other:

Complexity of Large-Scale Projects *
Identification of Implementable Tasks
Formulation of Realistic Work Breakdown Structure
Adoption of Proven Project Management Practices
Other:

Back

Submit

Page 3 of 3

Clear form

Never submit passwords through Google Forms.