M. ALMASHHADANI

ATILIM UNIVERSITY 2023

AGILE SOFTWARE MAINTENANCE AND DEVELOPMENT USING CLOUD COMPUTING FRAMEWORK

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

OF

ATILIM UNIVERSITY

MOHAMMED ALMASHHADANI

A DOCTOR OF PHILOSOPHY THESIS

IN

THE DEPARTMENT OF SOFTWARE ENGINEERING

JANUARY 2023

AGILE SOFTWARE MAINTENANCE AND DEVELOPMENT USING CLOUD COMPUTING FRAMEWORK

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

OF

ATILIM UNIVERSITY

MOHAMMED ALMASHHADANI

A DOCTOR OF PHILOSOPHY THESIS

IN

THE DEPARTMENT OF SOFTWARE ENGINEERING

JANUARY 2023

Approval of the Graduate School of Natural and Applied Sciences, Atilim University.

_____

Prof. Dr. Ender Keskinkilic
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of **Doctor of Philosophy in Software Engineering, Atılım University.**

_____

Prof. Dr. Ali Yazici
Head of Department

This is to certify that we have read the thesis "Agile Software Maintenance And Development Using Cloud Computing Framework" submitted by "MOHAMMED ALMASHHADANI" and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Doctor of Philosophy.

_____	_____
Prof. Dr. Alok Mishra Co-Supervisor	Prof. Dr. Ali Yazici Supervisor

Examining Committee Members:

Prof. Dr. Ali Yazıcı
Software Eng. Department, Atılım University _____

Asst. Prof. Dr. Beytullah Yildiz
Software Eng. Department, Atılım University _____

Prof. Dr. Murat Koyuncu
Information Systems Department, Atılım University _____

Prof. Dr. Seref Sagiroglu,
Computer Eng. Department, Gazi University _____

Asst. Prof. Dr. Gül Tokdemir
Computer Eng. Department, Cankaya University _____

Date: 09/01/2023

I declare and guarantee that all data, knowledge and information in this document has been obtained, processed and presented in accordance with academic rules and ethical conduct. Based on these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: MOHAMMED ALMASHHADANI

Signature:

ABSTRACT

AGILE SOFTWARE MAINTENANCE AND DEVELOPMENT USING CLOUD COMPUTING FRAMEWORK

ALMASHHADANI, MOHAMMED

PhD., Department of Software Engineering

Supervisor: Prof. Dr. Ali Yazici

Co-Supervisor: Prof. Dr. Alok Mishra

JAN 2023, #147 Pages

Agile methods have emerged to overcome the obstacles faced in traditional software methodologies, such as the Waterfall, Prototype, Spiral, etc. There have been many studies that show the numerous features of the Agile methodologies, making them useful for software development. However, many studies have also proposed a framework to adapt the Agile methods to Cloud Computing to leverage the benefits from this environment. The existing studies focus on the adaptive development life cycle for Agile with the Cloud, but have so far been unable to include the maintenance process in a detailed manner. Among these attempts and as further contribution, the present work intends to introduce Agile software maintenance and development using Cloud Computing framework (ASMDCC) as a reference for developing software with the Cloud in respect of maintenance activities. The case study findings reveal that the combination of Agile with Cloud Computing can resolve the major issues faced in traditional software maintenance, making the role of this approach significant in globally/distributed software maintenance. Furthermore, it is shown that Cloud Computing services play a vital part in resolving software maintenance. Finally, the results indicate that using the ASMDCC framework

improves the challenges faced by the maintenance team compared to the traditional environment regarding management, infrastructure, collaboration, and transparency.

Keywords: Software maintenance, Agile methods, Cloud Computing, Global environment.

ÖZ

BULUT HESAPLAMA YAPISI İLE ÇEVİK YAZILIM BAKIMI VE GELİŞTİRMESİ

ALMASHHADANI, MOHAMMED
Doktora, Yazılım Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Ali Yazici
Ortak Tez Yöneticisi: Prof. Dr. Alok Mishra

Ocak 2023, # 147 sayfa

Çevik yöntemler şelale, prototipleme, spiral ve diğer geleneksel yazılım metodolojilerinin karşılaştığı sorunları önlemek amacı ile ortaya atılmıştır. Yapılan birçok akademik çalışmada çevik yöntemlerin yazılım geliştirmedeki yararlarını gösteren farklı özellikleri ve yönleri işlenmiştir. Bununla birlikte, birçok çalışmada bulut ortamında da çevik yaklaşımın yararlı olacğını gösteren yeni yapılar önerilmektedir. Mevcut çalışmalar bulutta çevik adaptif geliştirme üzerine yoğunlaşırken, bakım sürecinin çevik yaşam döngüsüne kapsamlı olarak katılmadığı görülmektedir. Bu çalışmanın amacı, bulut ortamında çevik yazılım bakımı ve geliştirmesi için bakım süreçlerinde kullanılmak üzere yeni bir Bulut Hesaplama Yapısı (ASMDCC) önermektir. Bu yapı üzerinde yapılan vaka çalışması bulguları, çevik yaklaşımla bulut kombinasyonunun geleneksel yazılım bakımında karşılaşılan başlıca sorunları çözebileceğini ve bu yaklaşımın küresel/dağıtılmış yazılım bakımındaki rolünü önemli kıldığını ortaya koymuştur. Ayrıca, Bulut Bilişim hizmetlerinin yazılım bakımını çözmede hayati bir rol oynadığı gösterilmiştir. Son olarak, sonuçlar, önerilen yapının kullanılmasının, yönetim, altyapı, işbirliği ve şeffaflık ile ilgili geleneksel ortama kıyasla bakım ekibinin karşılaştığı zorlukları iyileştirdiğini göstermektedir.

Anahtar Kelimeler: Yazılım Bakımı, Çevik Yöntemler, Bulut Bilişim, Küresel Çevre.

To My Dear Family

ACKNOWLEDGMENTS

I would like to express my sincere appreciation to my supervisor Prof. Dr. Ali Yazıcı, who advised and guide me during my studies. Additionally, my many thanks go to Prof. Dr. Alok Mishra, who guided and supported me all along.

I wish to thank my thesis jury members, Prof. Dr. Seref Sagıroglu and Asst. Prof. Dr. Beytullah Yıldız for their guidance in the evaluation and review of my study. My appreciations go to Prof. Dr. Muhammad Younas, who supported and advised me from the beginning.

Sincere gratitude to my parents, dear wife, and children, Maryam and Yaseen, as well as all those who stayed by my side to help me complete this thesis. Finally, I truly appreciate their efforts.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS/ABBREVIATIONS

No	Short Form	Key Words
1.	**GSD**	Global Software Development
2.	**XP**	Extreme Programming
3.	**SLR**	Systematic Literature Review
4.	**QDA**	Qualitative Data Analysis
5.	**SM**	Software Maintenance
6.	**ADCC**	Agile Development Cloud Computing
7.	**SM**	Systematic Mapping
8.	**DSDM**	Dynamic Systems Development Method
9.	**GCP**	Google Cloud Platform
10.	**AWS**	Amazon Web Services
11.	**CSP**	Cloud Service Provider
12.	**ASMDCC**	Agile Software Maintenance Development Cloud Computing

CHAPTER 1

INTRODUCTION

In this chapter, we briefly review the information about Agile history and Agile Maintenance, and define our research problem, objectives, and questions.

1.1. Agile Revolution

In 1968, the NATO Science Committee held its first software conference in order to seek a replacement for the 'ad-hoc' approach concerning developing software. After that, the Waterfall model was proposed by Royce in 1970, which has been adopted as a model for developing software to this day in some companies [1].

In early 1970's, the software development process moved to another stage by following a series of phases starting with the concept, then feasibility study, design, development, a pilot version, and ultimately a final version [2].

The flexibility was needed to keep pace with the development of production. In 1986, the author in [3] proposed a new comprehensive process based on speed and flexibility in production. Subsequently, an approach called 'Scrum' and based on rugby was introduced that relies on six features, namely: self-organizing, multi-learning, built-in instability, organizational transfer of learning, overlapping development phases, and subtle control.

Agile is based on an iterative development approach that is considered as an umbrella for different practices. The Agile methods used to overcome the difficulties encountered by traditional software development through focusing on customer satisfaction, changing requirements, encouraging small self-organizing teams, and direct collaboration with customers [3].

1.2. Agile Maintenance

The Agile methods accommodate very popular software development processes, and due to their characteristics, such as speed, scalability, reliability, and flexibility, they have also been employed in maintenance processes [4], [5]. In this case, one of the Agile principles is to increase the interaction between maintainers and customers. This is one of the reasons that has encouraged the use of the Agile methods for the maintenance process. As Figure 1.1 shows, Agile maintenance lies at the intersection of Agile methods and classical software maintenance [6]. In general, The Agile methods have a group of common practices, some of which work for both development and maintenance activities [7], [8] These practices were extracted from Agile methods such as XP, Scrum, and other related families to support the Agile principles.

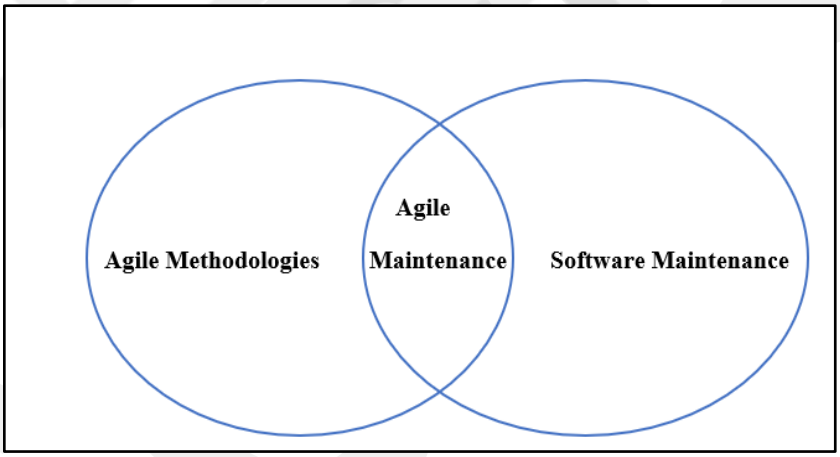


Figure 1.1: Combination Agile and Maintenance [6].

1.3. Research Study Statement

It is well known that the maintenance process is an integral part of the software life cycle, and that it is the longest stage; thus, requiring more research. In the previous section, we saw some of the Agile maintenance benefits, which encourage us to link and adapt Agile maintenance with the Cloud Computing environment to take advantage of the characteristics of this environment with the ultimate goal to facilitate the maintenance process.

1.4. Research Problems

For long, Software Maintenance (SM) has been considered as a significant challenge for both IT management and customers. SM is different from software development; that is because the maintenance process needs a full understanding of the system to perform it; whereas, this procedure is not necessary in the development process because the it always starts from scratch. Thus, software maintenance has its own activities and needs to adopt certain practices. As it is known, Software Maintenance consumes between 40-70% of the life cycle [9], which means that far more studies are required to reduce the cost and efforts.

According to Lehman [10], "The need to minimize software cost suggests that large-program structure must not only be created but must also be maintained if decay is to be avoided or postponed. For this reason, we need to minimize the time and effort for achieving maintenance activities which will definitely lead to minimizing the total cost as a result. The many benefits gained by using Agile for maintenance, as stated in [11], are listed below:

1. Speeding up the process;
2. Improving communication among team members;
3. Enhancing user satisfaction and team motivation; and
4. Increasing code quality.

Cloud Computing has many features that are utilized to simplify Agile development [12], [13], [14]. Our aim is to employ Cloud attributes and perform Software Maintenance in the Cloud environment to facilitate the maintenance activities.

Agile Development in Cloud Computing (ADCC), is a framework proposed by Younas in [12], and has been evaluated in [13] and [14]. This framework is used for developing software over the Cloud environment. ADCC introduces useful tools and techniques to create an interconnection between the Agile and the Cloud environment to simplify the developing software, but there is a gap in this framework which can be described as follows:

1- The ADCC framework does not include any maintenance activities, which are a significant part of software life cycle; and
2- The ADCC framework tends to overlook some important Agile tools and practices that could otherwise be employed to increase the development productivity.

With these in the background, the present work attempts to extend the ADCC framework by involving it in the maintenance process to help simplify the latter through the use of Cloud Computing platform.

1.5. Research aims and Objectives

The ultimate goal here is to help practitioners to develop and maintain software to complete their projects with minimal effort by utilizing Cloud Computing characteristics for these purposes. To this end, the following tasks are set:

- Identifying the existing Agile Maintenance challenges through conducting Systematic Mapping (SM) and Survey;
- Extending the ADCC framework to include the maintenance process, and determining the necessary tools that would overcome existing challenges; and
- Analyzing the results of applying Agile maintenance in the ADCC framework.

1.6. Research Questions and Motivation

RQ1: *What are the different Agile practices that can be used to facilitate the maintenance process in the Cloud environment?*

Concerning the motivation, for this query, in a rapid global business environment, many organizations have shifted to Agile development instead of traditional development methods owing to the flexibility of this approach to meet the changing customer requirements. Therefore, many studies have proposed frameworks or models for adapting the Agile practices for software development over the Cloud platform; among them [12], [14], [15], and [16]. Although, many research studies have covered the issue of adapting Agile development with the Cloud, there is a need to adapt the Agile practices to include the software maintenance phase as well. Therefore, the goal of this research is to Identify

which Agile practices are used for maintenance and knowing the impact of Cloud Computing to facilitate Agile practices.

RQ2: *What are the advantages of using Agile and its practices during Software Maintenance?*

This is motivated by the initiative to identify the effectiveness of applying the Agile during Software Maintenance.

RQ3: *What are the challenges in using Agile maintenance in local and global environments?*

Identifying and verifying the existing challenges faced by the maintenance process using the Agile method for both environments is the main motivation here, not to mention trying to solve these challenges with the help of the Cloud.

RQ4: *What are the outcomes of involving an Agile maintenance in ADCC framework?*

This final question is inspired by the need to obtain a better grasp on the results of adapting the maintenance process within the ADCC framework and the Cloud environment.

In this research, there are certainly some limitations to achieving our objectives, which are discussed in the following.

- Some relevant studies may have been missed while searching the databases; and
- The number of organizations that adopted Agile for software maintenance is limited, and we faced difficulties finding a prominent number of participants for the survey and case study.

1.7. Thesis Structure

This thesis consists of 7 chapters; each containing several sections. The first chapter provides an introduction to the Agile and Agile maintenance, as well as identifying the problem scope and research questions aimed to answer. The second chapter is an extensive search in respect of the related literature studies through Systematic Mapping to identify the challenges that face Agile maintenance. In the third chapter, a survey is conducted to accurately determine the challenges identified in Chapter Two based on the opinions of an Agile maintenance professional. In the fourth chapter, the survey is analyzed using

appropriate methods. The fifth chapter extends the Agile Development in Cloud Computing Environment (ADCC) framework to include the maintenance process by means of a case study to evaluate the framework. The sixth chapter, the discussion is conducted. Finally, in the seventh chapter, future works are presented. Figure 1.2 illustrates the thesis structure.

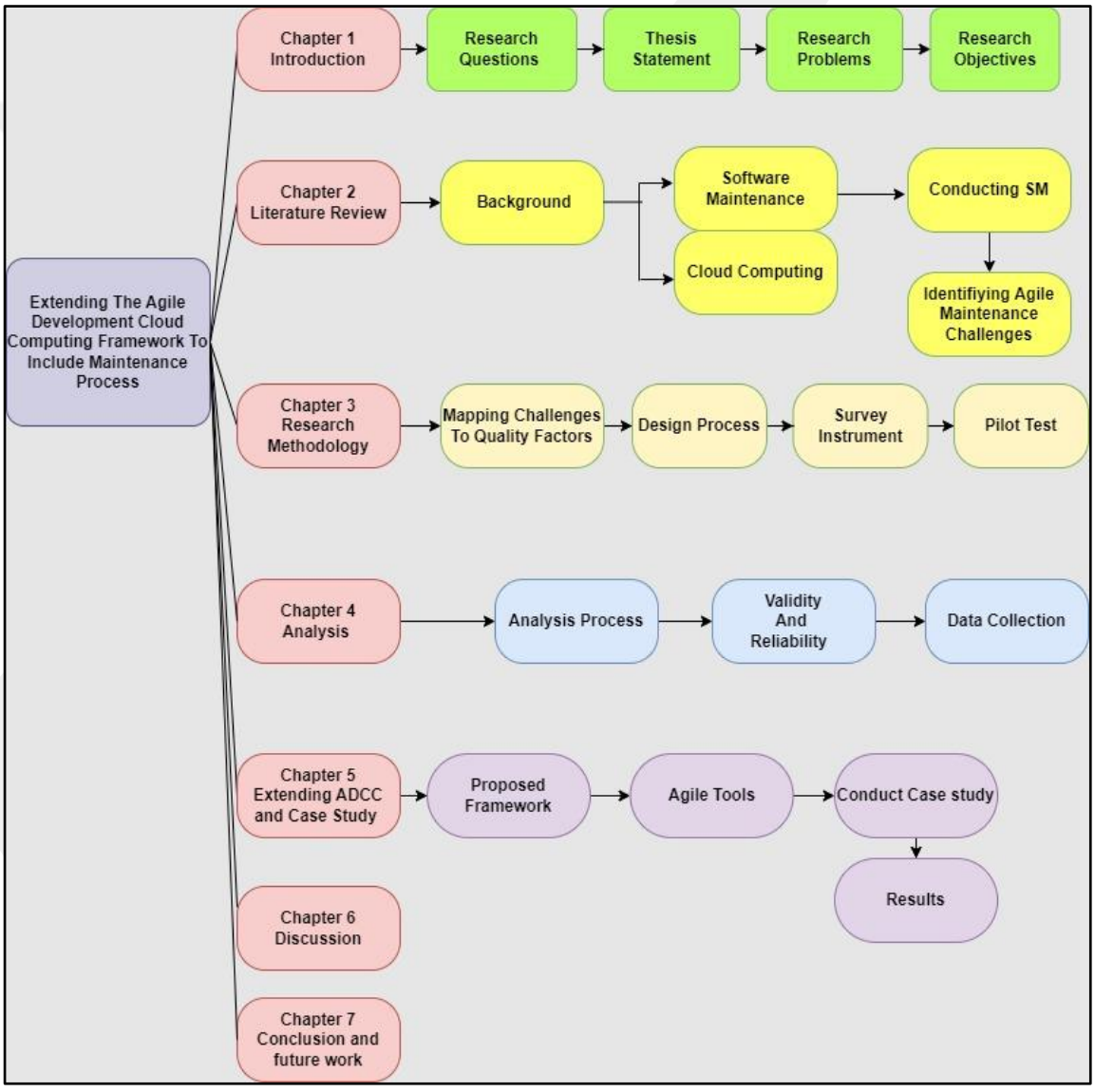


Figure 1.2: Overview of the Thesis Chapters

CHAPTER 2

LITERATURE REVIEW

In this chapter, we carry out a Systematic Mapping (SM) analysis to identify the practices used in the Agile Maintenance process that could be used in the Cloud environment. Upon conducting the systematic research, nine research questions were set up and answered related to Agile maintenance.

2.1. Software Maintenance

One of the most essential features of software development is the need for continuous change to keep pace with technological progress [10], [17]. For that reason, the software is constantly evolving according to business needs and the system, as such, requires continuous maintenance.

According to Pigoski [18], the maintenance process is not separate from the development process. According to his view, the maintenance process is initiated at the start of software development. According to The Institute of Electrical and Electronics Engineers (IEEE) definition [19]:

> *"Software maintenance is the process of modifying a software system or component after delivery to correct faults, improve performances or other attributes, or adapt to a changed environment."*

The maintenance process is one of the longest stages in the Software Development Life Cycle (SDLC) and, as a result, it consumes the largest part of the system total cost. Many studies have been conducted to facilitate the maintenance process and reduce the efforts spent during the maintenance activities; and still, there is a need for more studies in this area [20] .

There are many other parts that should be changed together with the system code, such as system design, documentation, and user manual. All of these falls under the scope of maintenance, which, in general, is about changing the management process [20], [21].

2.1.1. The need for Software Maintenance

The maintenance process is an indispensable part of maintaining the software due to constant changes in the environment, and the software needs to change to adapt to such environments [22].

The main objective of the maintenance process is to keep the system stable and working in accordance to the user requirements. There are many reasons to perform maintenance, such as fixing errors, preventing system failure, and performing system updates to keep pace with the changes [20].

The maintenance process is necessary according to the users' requirements needs; as a result, the system also needs to change in accordance with these requirements. There are several reasons behind changing the system; for example, improving performance, adding new functions, and so on. The need for the maintenance process can be summarized under three main categories according to [23]; these are:

- Fixing bugs: fixing errors to keep the system running;
- Adapting the system: adapting the system to continue operations within the changing business environment; and
- Supporting users: to provide users with backup and assistance when needed.

In the next section, different Software Maintenance types are discussed and illustrated in more details.

2.1.2. Software Maintenance Types

There are four main types of software maintenance, and the purpose of classified maintenance is to find out why maintenance is necessary [18]. Understanding a certain flaw in the right manner can help in fixing a problem without complications in any context; in the same way, software maintainers can easily perform maintenance if they classify the

problems based on type [20]. According to the IEEE standards, there are four maintenance types; namely, adaptive, preventive, corrective, and emergency [24]. These are discussed in the following with brief explanations.

2.1.2.1. Corrective maintenance

It includes correcting errors after delivering the software to the end user. These errors are of different natures; such as logic, design, or code [23], [25].

2.1.2.2. Adaptive maintenance

This type of maintenance includes updates and modifications to keep the system pace with the changing technology and business environments [23], [25].

2.1.2.3. Perfective maintenance

This type of maintenance includes updates and modifications to maintain the system for the longest period of time, and to make it integrated in terms of functional and non-functional requirements [23], [25].

2.1.2.4. Preventive maintenance

This type of maintenance includes carrying out changes and updates to prevent problems that may occur in the future and cause system failure [23], [25].

2.1.2.5. Emergency maintenance

The IEEE standard has defined this type of maintenance as when emergency errors occur and require fixing to keep the system operating with continuity and consistency [24].

2.1.3. Maintenance Cost

Maintenance consumes the highest percentage of the total cost. According to surveys, this rate stands at 60-80% [20], with the highest expenditure being for adding new requirements, as opposed to corrective maintenance.

According to these surveys, the percentages of maintenance costs vary according to its types; for instance, 50% of the maintenance cost is used for perfective maintenance, 21% for corrective maintenance, 25% for adaptive maintenance, and only 4% for preventive maintenance [26]. Figure 2.1 illustrates the maintenance effort distribution.

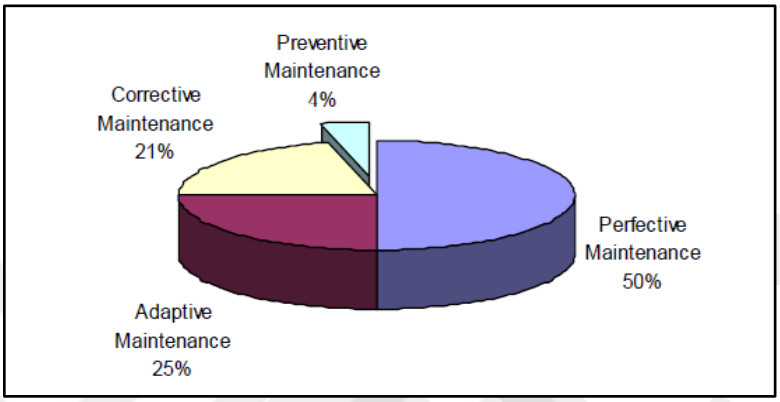


Figure 2.1: The Distribution of Maintenance Efforts [26].

2.1.4. Maintenance Models

Many software maintenance models have been proposed and, in what follows, we review the most common ones as mentioned in [23].

2.1.4.1. Quick-fix model

It is also known as the 'firefighting' approach. This model is proposed to maintain the system (i.e., try to fix the problems facing the system as soon as possible). In this model, bugs are fixed without any concern for future consequences. Figure 2.2 shows the Quick-fix model [23].

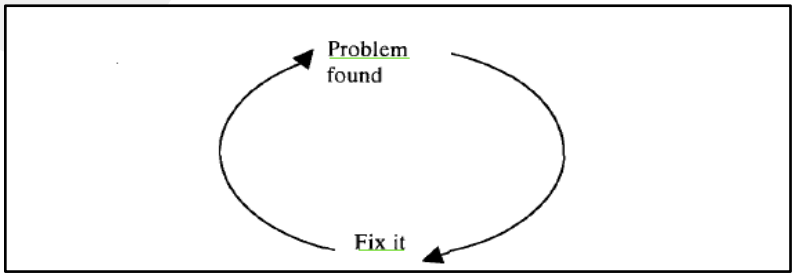


Figure 2.2: The Quick-fix model [23].

2.1.4.2. Boehm's Model

This model was proposed by Boehm in 1983 based on economic models and principles. The maintenance is represented as a closed loop, as shown in Figure 2.3, which illustrates more about this model [23].

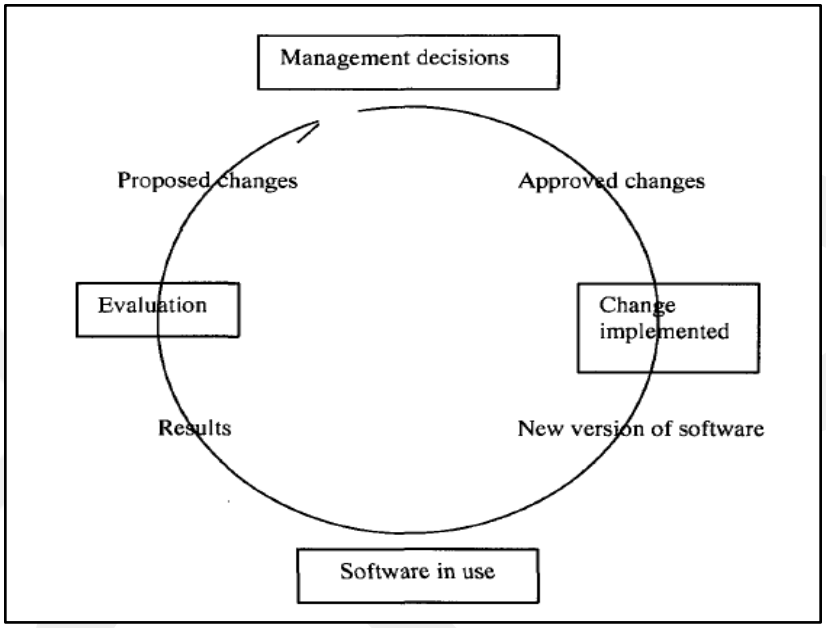


Figure 2.3: Boehm's Model [23].

2.1.4.3. Osborne's Model

This model, proposed by Osborne, differs from others as it deals directly with the maintenance environment. It considers that most of the problems are caused by a lack of management communication and control [23].

2.2. Cloud Computing

Cloud Computing is one of the recent technological trends based on distributed and grid computing. There are many definitions for Cloud Computing, but the most popular one is presented by the National Institute of Standards and Technology (NIST).

According to the NIST [27], Cloud Computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

It appears from the definition that Cloud Computing contributes to reducing the total expenditure of the organization through managing resources and reducing the burden of system maintenance [28].

2.2.1. Cloud Computing Advantages

There are many advantages of Cloud Computing as reflected in the literature studies, which have contributed to spreading this technology and encouraging organizations to transfer to this new environment. The most important benefits are [29]:

- Reduced Cost: Cloud Computing is based on a pay-as-you-go principle; thus, the total cost is reduced;
- Flexibility: This refers to the ability of employees to work from everywhere and to share files while via the Internet;
- Scalability: In the Cloud platform, the organization can scale up or scale down according to need, thus helping to reduce the organization's overhead.
- Maintainability: In Cloud Computing, the service provider takes care of maintenance while the organization focuses on business and innovation. In addition, there is no need to pay for maintenance technicians.
- Reliability: Cloud Computing provides data backup, disaster recovery, and business continuity more easily and less expensively because data can be copied to different locations.
- Data security: Users can access their data with end-to-end encryption and isolate the resources among different users, thus helping to protect their data.

2.2.2. Cloud Computing service models

There are many Clouds service models, among which the organizations can select according to their needs. These models are reviewed in more details below:

2.2.2.1. Software as a Service (SaaS)

In this model, users release applications in the host environment and can access them through their Internet browser. In this model the users cannot control the Cloud infrastructure. There are many platforms that provide this model, such as Google Mail, SalesForce.com, Google Docs, and so on [30].

2.2.2.2. Platform as a Service (PaaS)

This model allows developers to create and deploy applications on the platform. PaaS has developing environments, such as IDE, along with configuration management and necessary tools for developing software. Many platforms provide this model; among them, Google AppEngine and Microsoft Azure [30].

2.2.2.3. Infrastructure as a Service (IaaS)

IaaS provides users with essential computing and services; for example, storage, processors, network, and other essentials on the basis of 'pay-as-you go'. IaaS is based on the virtualization principle to a large extent to provide and distribute resources in an isolated manner, ensuring the privacy of each user. Many Cloud platforms provide IaaS, such as Amazon EC2, Google Compute Engine, Rackspace, Nimbus, etc.[30].

2.2.2.4. Anything as a Service (XaaS)

It refers to delivering everything as a service, such as tools, products, and technologies delivered to users through the Internet as opposed to their provision on the website of the organizations [30]. Figure 2.4 illustrates different Cloud service models.

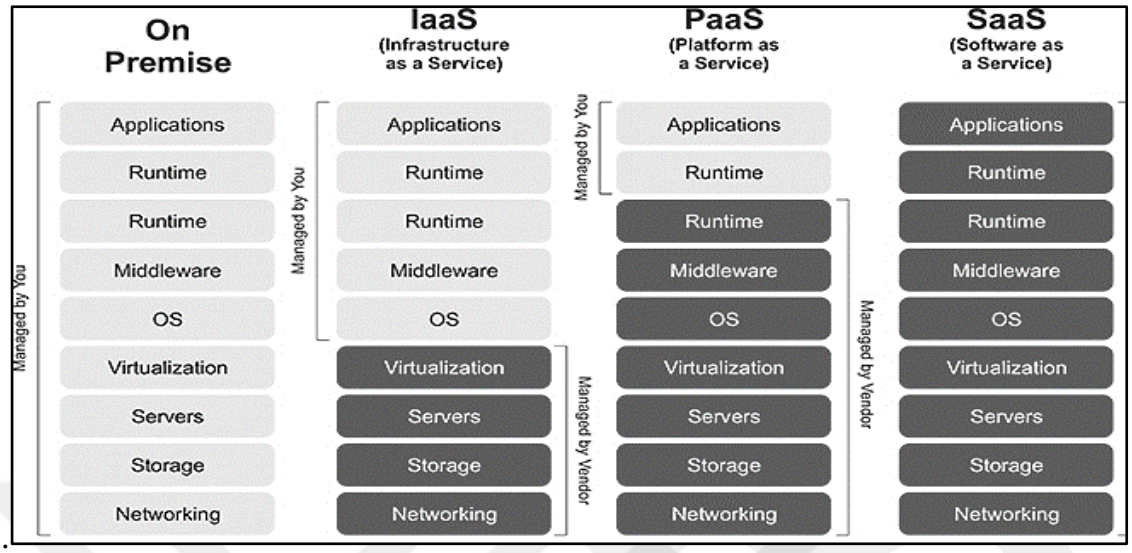


Figure 2.4: Cloud Service Models

2.2.3. Cloud Deployment Model

2.2.3.1. Private Cloud

This model is created by organizations for exclusive purposes, especially when the organization has sensitive data – as do government institutions. It is recommended to build a private model due to ensuring privacy; in which case, the entire infrastructure will be under control [30], [31].

2.2.3.2. Public Cloud

In this model, the infrastructure is created for a public purpose other than in the private model. Here, the infrastructure is under the Cloud provider's control, thus implying that neither the service provider responsible for managing, operating, and maintaining the entire infrastructure, nor the customers have any physical control over the hardware infrastructure [30], [31].

2.2.3.3. Hybrid Cloud

This model is incorporated between private and public Cloud; it is used when an enterprise owns some resources and rents other resources to a third party [30], [31].

2.2.3.4. Community Cloud

In this model, many organizations cooperate to build this infrastructure to support a specific community. The infrastructure may be managed by the organizations or a third party [30], [31]. See Figure 2.5, which illustrates alternative Cloud deployment models.

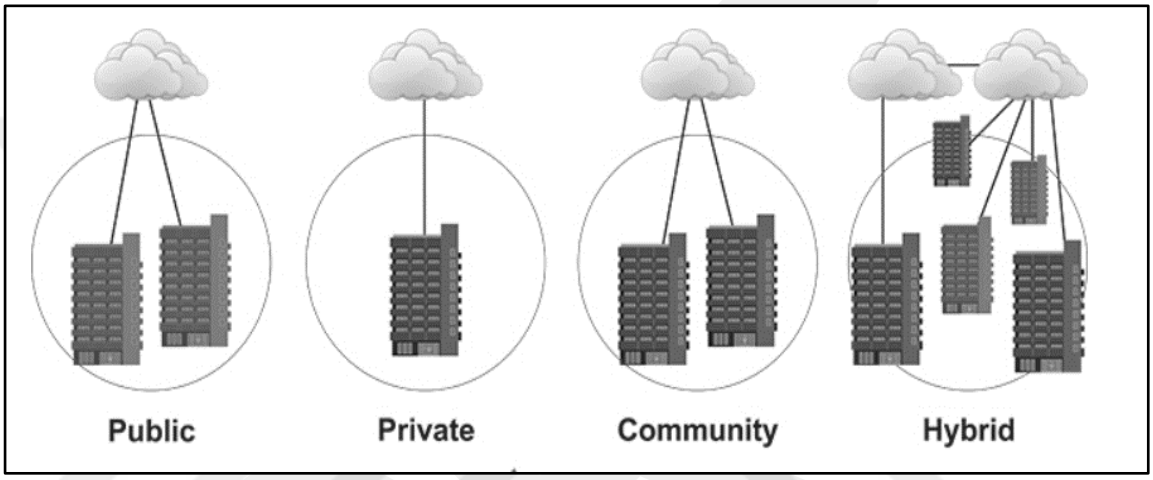


Figure 2.5: Cloud Deployment Models

2.3. Agile And Cloud Computing

There are many studies about Global Software Development (GSD), some of which show the challenges facing development in the global environment and introduce possible solutions [32]. However, there are many common challenges in a globally distributed context such as lack of communication, lack of management, low team morale, and unclear or uncertain requirements, which can be solved with the help of Cloud platforms [12], [15]. Other studies have suggested models or frameworks for adapting Agile to Cloud Computing environment.

The authors in [15], introduced the Dynamic Systems Development model by using Google App-Engine as a Cloud platform. In addition, Skype app is used to support collaboration among development team members. The model was evaluated by developing a warehouse management application using both environments (i.e., on-premises and on-Cloud), and the two environments were compared in terms of the

development time. The results of the study indicate that the development time using Cloud Computing is faster due to the application of Cloud resources in the development process. The proposed model consisted of three phases: (1) pre-project phase, which is responsible for identifying and arranging the priorities, feasibility studies, and defining the project goal; (2) Life Cycle Project phase, where functional model iteration is defined, the design is conducted, and the implementation of the project is done; and (3) post-project phase, which measures the functioning efficiency and effectiveness and is interested in error detection and correction [15].

In another study [16], the interconnectedness between Agile and Cloud platforms is studied by surveying one of the development organizations. The survey questions mainly focus on collaboration between development team members and the Cloud services provider. The company uses Scrum as the Agile methods, and Skype as a tool for collaboration among development team members. The results of this study showed that the use of Cloud Computing greatly contributed to facilitating cooperation among team members, which was positively reflected in the increased agility.

In [33], a crowd-sourcing model is proposed, which can also be used for Agile-Cloud development. In the study, Confluence is used as a tool for communication between team members, and GitHub for code repository. For Agile Cloud integration, Chef and Puppet Lab, are utilized.

In [12] , the authors introduced a framework for Agile Development in Cloud Computing Environment (ADCC). In [13], and [14], the same authors evaluated the framework by introducing a structural environment for software development using distributed Agile over Cloud Computing platform. The ADCC framework has four steps: (1) Agile features selection, where the desired Agile method from the Agile family is selected, (2) Cloud platform selection according to organization size (small, middle, big) and, depending on business requirements, the organization's privacy (if there is a need to secure information); and (3) code management and repository, where tools such as GitHub, FishEye, Bamboo, and BitBucket are utilized to facilitate common development tasks. The ADCC framework is a novel approach for adapting Agile development software on the Cloud. However, in the same study, in the case of Agile maintenance, some shortcomings are reported.

2.4. Systematic Mapping

2.4.1. SM Process

To achieve our goal, a SM study is conducted to answer the research questions posed in Section 3.3. Kitchenham in [34], introduced a guideline for SM in software engineering. And for the execution of the SM, he suggested three steps, namely, planning, conducting, and documenting as shown in Figure 2.6.

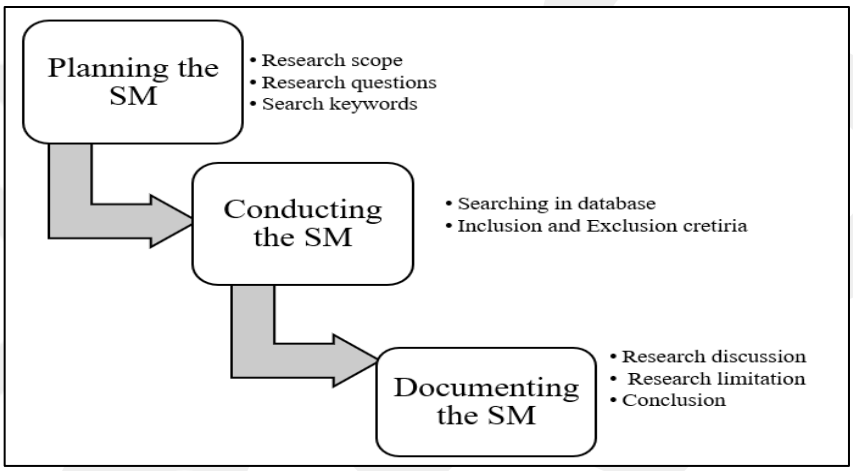


Figure 2.6:SM PROCESS [34].

2.4.2. Research Scope

We have searched in different databases and identified 48 studies out of 322 studies related to the scope, which covers the following inquiries:

1. What are the existing studies for Agile maintenance?
2. What are the different Agile practices for maintenance?
3. What are the benefits of using Agile for maintenance in local and global environments?
4. What are the challenges of using Agile for maintenance in local and global environments?

This SM study focuses on Agile maintenance models and practices based on XP and

Scrum, and it seeks the possibility of adapting the Agile maintenance model for the Cloud environment. The reason behind choosing these two approaches is the popularity of these two methods and the focus of related literature on them. It is clear that these two are the most widely used than the rest of the agile methods. Additionally, the Scrum method has many practices effecting the project management. Similarly, XP has many practices that are suitable for project engineering such as; continuous integration, pair programming, and refactoring. In other words, these two methods complement each other in many aspects of software development [8]. Moreover, project management and project re-engineering are the main factors in software maintenance. Additionally, there are many studies that proved the effectiveness of using XP and Scrum practices for maintenance [7], [8], [35], [36], [37], [38], [39], [40], and [41].

2.4.3. Research Questions

In this SM study, the main goal is to find answers and explanations to some detailed questions related to our scope. Table 2.1 includes seven research questions along with the purpose of each. All research questions are focused on software maintenance and Agile methods.

Table 2.1: Research Questions

#	Research Question	Main Purpose
RQ1	What are the existing studies in Agile software maintenance?	Investigate the existing studies about Agile maintenance.
RQ2	What are the differences between software development and maintenance?	Identify the main differences between the development and maintenance processes.
RQ3	What are the differences between traditional and Agile maintenance?	Identify the main differences between traditional and Agile maintenance processes.

Table 2.1 (cont'd). Research Questions

#	Research Question	Main Purpose
RQ4	What are the use and benefit of using an Agile approach for maintenance?	Identify the benefits of applying the Agile method for maintenance.
RQ5	What are the Agile practices (XP/Scrum) for local and global environments?	Identify which Agile practices can be used for the local and global environment**.**
RQ6	What are the Agile practices (XP/Scrum) that are used for maintenance?	Identify what Agile practices can be used for maintenance**.**
RQ7	What are the advantages of applying Agile practices for maintenance?	Identify the benefits of using Agile practices for maintenance.
RQ8	What are the challenges in using Agile maintenance for a local and global environment?	Identify existing challenges faced by the maintenance process using the Agile method for both environments.
RQ9	What are the advantages of applying Agile methods in a Cloud Computing environment?	Identify all the benefits of using SDLC in the Cloud.

2.4.4. Search Keywords Strategy

For the searching process, a group of keywords that are related to the current SM are identified and arranged as search strings and are illustrated in Table 2.2.

Table 2.2: Search Strings

Search strings
1. Agile and software maintenance model
2. Agile and maintenance framework
3. Agile and maintenance tools
4. Agile and XP maintenance
5. Agile and Scrum maintenance
6. Agile development and Global environment
7. Agile development and Cloud

2.4.5. Search Databases

Following the guidelines given in [34], the databases listed in Figure 2.7 are searched. In the same figure, the distribution of the 322 articles is given among, IEEE Xplore, Scopus, Springer, Web of Science, Science direct, and other databases such as (ACM, Google Scholar, CiteSeer (X), ACSI, etc.).

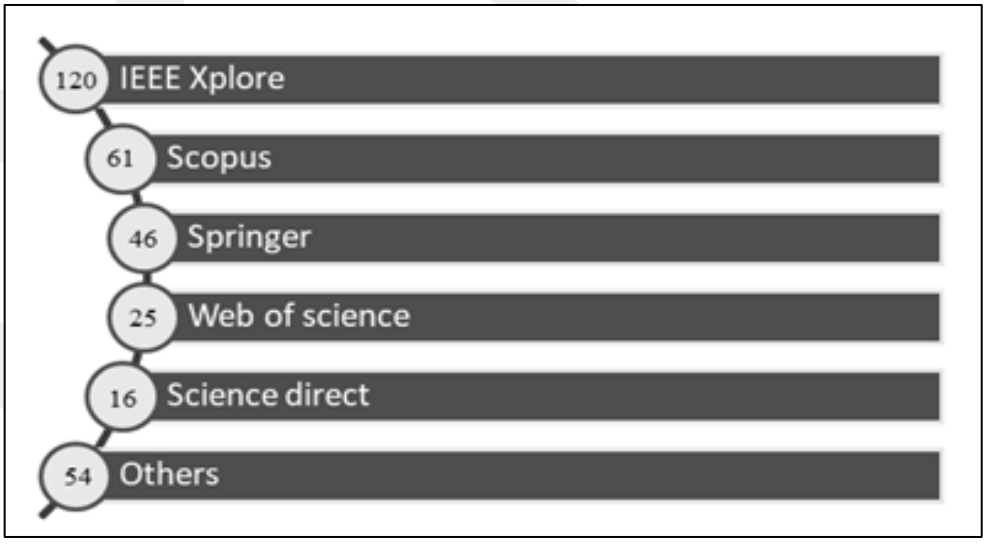


Figure 2.7: Distributed of Studies Over Databases.

2.4.6. Inclusion And Exclusion Criteria

In the selection process, studies that do not conform to the objectives of the study and the RQs are excluded. The inclusion and exclusion criteria for our research is listed below:

Inclusion criteria:

- Publications between 2000 and 2021.
- Publications proposing Agile maintenance methods, such as XP or Scrum.
- Case studies for Agile maintenance models.
- Publications addressing Agile practices for maintenance.
- Publications addressing Agile practices for the global environment.
- Publications addressing Agile practices for Cloud environment.

Exclusion criteria:

- Publications written in languages other than English.
- Workshops and web links.
- Publications not focusing on Agile maintenance models or Agile practices.

In addition, for the studies that appear in more than one database (overlapping studies), only one is included in the outcome set. The selection process is illustrated in Figure 2.8. In the first phase, articles according to the title related to research the scope and the results are filtered (322 papers). In the next phase, the papers are filtered, after reading the abstract and conclusion (102 papers). After that, in the third phase, 74 papers are read in detail. In the final phase, the remaining 48 articles are investigated in detail and analyzed to extract results and findings.

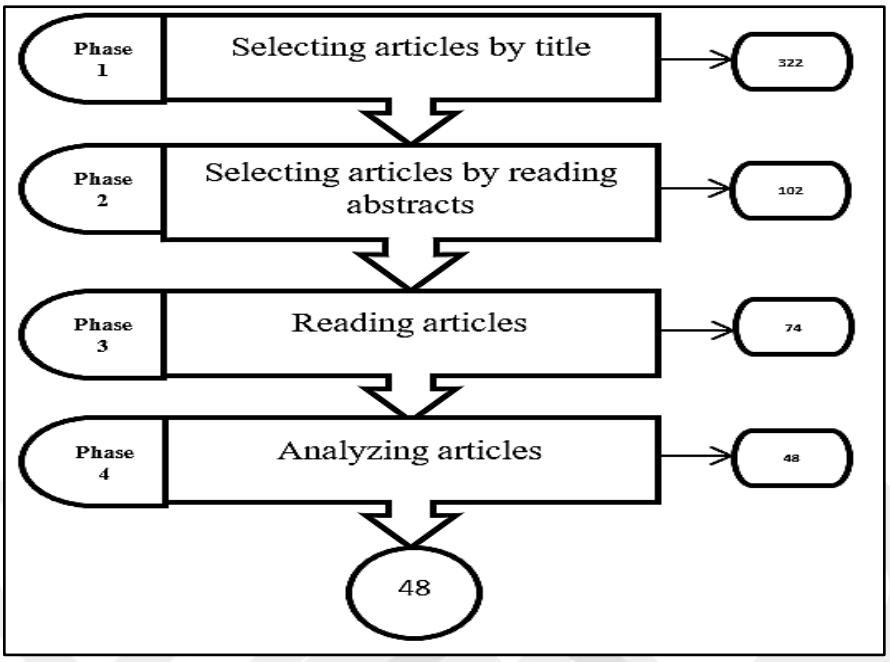


Figure 2.8: Phases of Selection Process.

2.4.7. Answers To Research Questions

RQ1. What are the existing studies in Agile software maintenance?

In this section, the studies that propose Agile maintenance models or studies related to Agile maintenance practices are reviewed. The main goal of this step is to identify articles in this field and help researchers to study other aspects that have not been addressed so far. In many studies software maintenance for Agile methods are discussed. A list of these studies is provided in Table 2.3.

Table 2.3: List of Studies

Legend: - CS: Case Study, MOD: model, TOL: Tool, FRM: Framework, OTH: Other.

Study ID	Studies description	Study type
[42]	A case study of an offshore Agile maintenance project which introduces practices that would address challenges facing offshore software maintenance, and present some observations about the project.	CS

Table 2.3 (cont'd). List of Studies

Study ID	Studies description	Study type
[7]	A case study for adapting Extreme programming XP practices at Lona Technologies. The maintenance method was demonstrated along with the case study from the company, and comments were made on the maintenance process.	CS
[35]	A case study conducted at a large software development company to verify the results of introducing XP for maintenance. The result proved that the XP practices need re-designing and adaptation to fulfill the requirements and needs.	CS
[43]	A case study was conducted at an industrial company to identify the challenges faced in developing and maintaining software using the Agile methods. A literature review is provided in which tools, methods, and knowledge which are available for refactoring of code smells are investigated.	CS, LR.
[8]	Researchers proposed a model for evolution and maintaining software based on the Agile methods. XP and Scrum approaches are compared and maintenance practices are investigated. Also, the proposed model was evaluated using two Canadian companies.	MOD, CS.
[44]	A framework is based on Agile techniques and principles is proposed, and then applied in 13 IT services.	FRM, CS
[36]	A maintenance model is proposed applying the Scrum method which deals with urgent requests during the current sprint. The authors evaluated the study by maintaining a fitness system. The maintenance team was in South Asia and customers were in the USA.	MOD, CS

Table 2.3 (cont'd). List of Studies

Study ID	Studies description	Study type
[11]	The authors adapted practices of Agile development for maintenance activities and produced nine heuristics. This study was conducted in collaboration with the IT department at Aalborg University and the maintenance team at Aveva organization.	CS
[45]	The researchers introduced a systematic review about Agile maintenance covering 30 research articles to help practitioners and maintainers. This study answered 10 questions, leading to adopting an Agile method for maintenance.	SR
[37]	This study highlights the impact of different Agile methods on maintainability as seen by fans and critics; it also shows the benefits of using Agile (XP, Scrum) approach for the maintenance process.	OTH
[9]	The main purpose of this study is to enhance software maintenance governance for the Agile maintenance team, and to propose a new tool called Axita developed to support the team by managing time projects effectively and arranging data in a central data warehouse. Furthermore, in this study, the authors proposed six practices for managing the maintenance process to overcome the challenges that would occur during searching information and reduce the processing time.	TOL
[46]	This study was conducted on Open Stack platform to evaluate software maintenance for PF Cloud systems that have been developed using the Agile development method.	CS

Table 2.3 (cont'd). List of Studies

Study ID	Studies description	Study type
[47]	This is an exploratory study that investigates whether the Agile methods can help software maintainers. In addition, it introduces a matrix framework based on practitioner views, thus aiming to tackle issues that impact Agile adoption decisions.	CS, FRM
[38]	The Agile process for maintenance life cycles based on the Scrum method was introduced.	OTH
[48]	A list of metrics that can define a degree of maintainability, and to examining whether these metrics can apply to the Agile methods is reported.	OTH
[5]	A literature review about Agile maintenance was conducted. Related works through investigating the challenges and obstacles that face Agile maintenance are analyzed.	LR
[6]	22 Agile practices that can be adopted in software maintenance by conducting interviews and defining the advantages and disadvantages of these practices for the maintenance process are identified.	CS, LR
[49]	An agile software maintenance method called Agile MANTEMA, focusing on small organizations, is proposed. As a case study two companies were chosen and the effectiveness of the method was measured.	MTH, CS
[39]	An iterative maintenance life cycle based on XP was proposed to solve maintenance challenges such as cost, time, and effort, and increase the software maintainability.	MOD

Table 2.3 (cont'd). List of Studies

Study ID	Studies description	Study type
[40]	An iterative maintenance model based on XP was proposed to enhance maintenance. The model was evaluated in an academic environment by applying it in many projects. The observations showed that the use of this model can produce maintainable codes with good quality, and that this model speeds up the maintenance process with less effort.	MOD, CS
[41]	It is an empirical study that tests the maintainability of the system for an academic project. Each project was assigned to two groups of postgraduate students. One of them used the XP model and the other used the Waterfall model. The observation stated that the group that used the XP method produced a more maintainable code than the group that used the Waterfall model.	CS

The evaluation of Agile maintenance studies per year, illustrated in Figure 2.9, shows that the highest ratio of publishing was 4 articles in 2015, then 2 articles for 2009, 2011, 2014, and 2018.

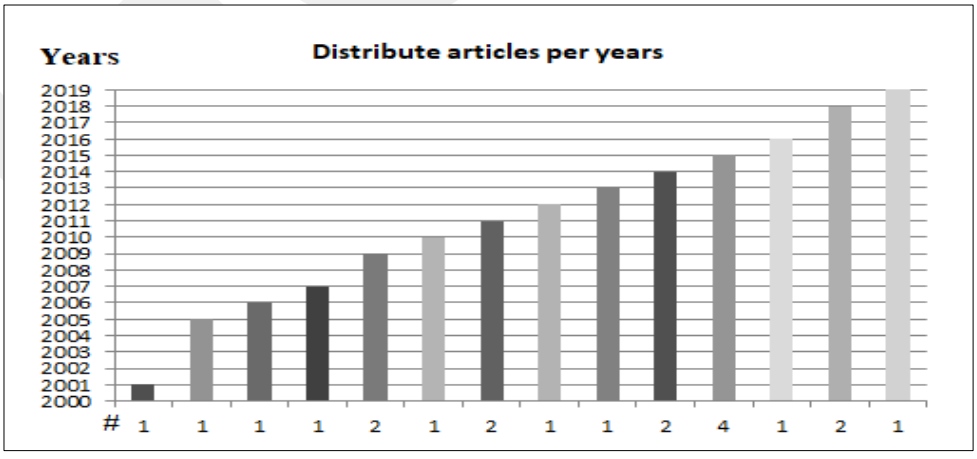


Figure 2.9: Distributed Agile Maintenance Studies Per Years.

Regarding publishing, 57% out of 21 studies are journal articles, while 33% are conference articles, along with 10% others (IBM report, thesis). In Figure 2.10 this distribution is shown.

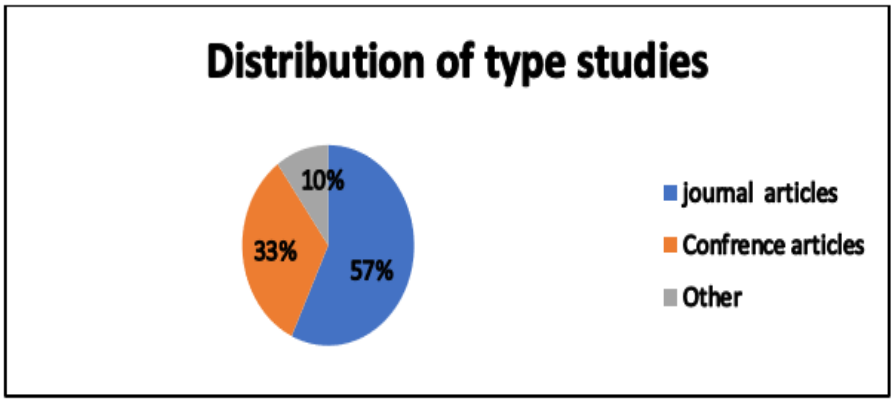


Figure 2.10: Distribution of Type Studies.

RQ2. What are the differences between software development and maintenance?

According to the IEEE definition [19], “Software Engineering describes framework for all phase of SDLC whereas software maintenance is about modification of software after delivery, correction of faults and performance improvement. We can understand from these definitions that there is a difference between software development and maintenance. Development refers to creating a product from the scratch “new product", while maintenance refers to continuous supporting to fix bugs or adapt the product after it is delivered to the end end-user [25], [50]. Software maintenance is different from Software development due to the dependability of maintenance on program comprehension [6]. The software maintenance life cycle has its models, namely Osborn, Boehm, Iterative Enhancement, IEEE and, reuse-oriented [6], [51]. Most of these models are built based on traditional models such as the waterfall [51], [39]. See Figure 2.11 which illustrates the development and maintenance process.

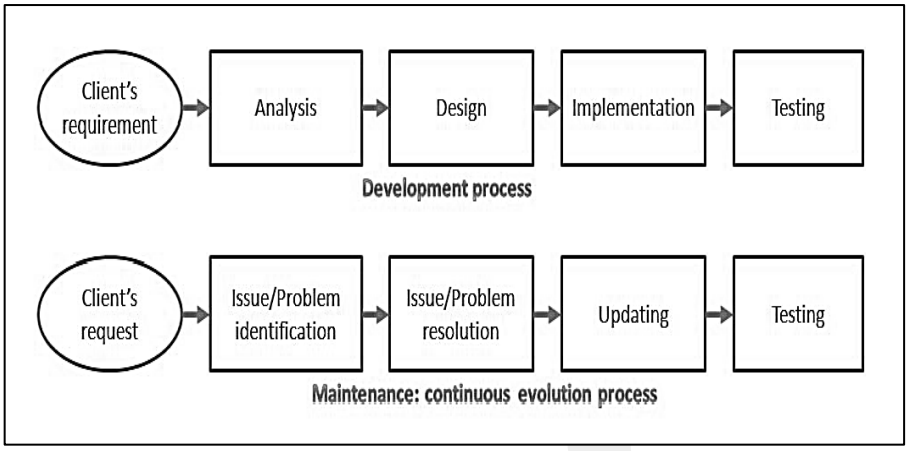


Figure 2.11: A Comparison Between SDLC And SMLC Process [51].

Table 2.4 compares the software development life cycle (SDLC) with the software maintenance life cycle (SMLC).

Table 2.4: SDLC VS SMLC.

Phases	**SDLC**	**SMLC**	**Comments**
1. Requirements	Identify user requirements.	Identify user problems.	Maintenance is more complicated because it needs product documents, while development does not depend on documents [11], [39].
2. Analysis	Analyze requirement.	Analyze problems or new functionality.	In maintenance, it requires more effort to avoid conflict between system functionalities [39].
3. Design	Design system according to SRS document.	Re-design the system according to new features.	In maintenance, this phase requires re-engineering to ensure this modification will not affect other system features [39], [51].

Table 2.4 (cont'd). SDLC VS SMLC.

Phases	SDLC	SMLC	Comments
4. Implementation	Implement the logic design.	Fix source code or add new code.	It is a very complicated phase because it needs to review the entire code to ensure the modification will not harm other models [39], [51].
5. Delivery	Testing the accepted system	Testing the accepted system	Same process [39], [51].

Therefore, it can be concluded that the maintenance phases require more effort to avoid damaging the rest of the system units.

RQ3. What are the differences between traditional and Agile maintenance?

The Agile development methods have emerged to overcome the obstacles that have been faced by conventional software development methods, especially regarding the increases in cost and efforts. Since maintenance is a vital phase in the software life cycle, and it consumes a major share in the total cost and efforts. Due to this reason, Agile methods are applied for software maintenance. To further investigate the differences between traditional maintenance and Agile maintenance, the literature studies have been reviewed and differences are highlighted in Table 2.5.

Table 2.5: Traditional Maintenance VS Agile Maintenance [40], [41].

Traditional maintenance model	Agile maintenance model based on XP
1. The requirements are changed by the developer.	1. Requirement changed by customers.
2. The process is based on the Waterfall model.	2. It is an iterative process such as Scrum and XP model.
3. The process has a long incremental.	3. It has a short incremental.

Table 2.5 (cont'd). Traditional Maintenance VS Agile Maintenance [40], [41].

Traditional maintenance model	Agile maintenance model based on XP
4. The customer role is limited to writing requirement submission and validation	4. The customer role is requirement submission, prioritization, and validation.
5. The plan is for the entire requirements change.	5. Each iteration requires planning according to the customer's need.
6. The design changes if necessary.	6. The design changes continuously by refactoring.
7. The style for programming is individual.	7. The style is pair-programming.
8. Code ownership is module-wise.	8. Code ownership is collective.
9. Integrations are Weekly or monthly.	9. Integrations is continuous.
10. Code reviewing is done at the end.	10. Code reviewing is done continuously by pair programming.
11. Testing is carried out after implementation.	11. Testing is carried out before implementation through TDD.
12. Maintenance efforts increase with the low rate of sequential stories.	12. Maintenance efforts increase with the high rate of sequential stories.
13. The documentation is huge and comprehensive.	13. Documentations are limited to necessary activities. (Light documentation)
14. The phases are sequential.	14. The 3-7 phases are parallel.
15. Regression tests use the traditional approach.	15. Regression tests are done frequently through continuous integration.

RQ4. What are the use and benefit of using an Agile approach for maintenance?

There are many advantages acquired by applying Agile methods for the software maintenance process. Table 2.6 summarizes the main advantages the literature studies have referred to.

Table 2.6: Advantages of Using Agile for Maintenance.

Agile maintenance benefits	Studies
1. Speed up in the maintenance process.	[11], [48], [39]
2. Improved communication.	[11], [39], [52]
3. Increase in customer satisfaction.	[36], [44], [48]
4. Increase in product quality.	[11], [37], [52]
5. Increase in the morale among the team members.	[11]
6. Increase in software maintainer's satisfaction due to reducing rework.	[36], [39]
7. Simplify knowledge transfer among the teams.	[7], [35], [11]
8. Improve productivity.	[35], [6], [39], [37]
9. Improve project visibility.	[11], [39], [7]
10. Fix bugs and merge them with code rapidly.	[37], [6]
11. Improv prioritization of tasks.	[6]
12. Enhance test suites.	[6]
13. Increase the index of maintainability.	[6]
14. Improve program comprehension.	[6], [39]
15. Improve the accuracy of cost and effort forecasts.	[6]
16. Improve the design at the abstraction level.	[6]

RQ5. What are Agile practices (XP/Scrum) for local and global environments?

The Agile approach is applied with different methods such as XP, Scrum, Crystal, Lean, Feature-Driven Development (FFD), etc. [53]. The most popular methods of Agile methodologies are XP and Scrum. While Scrum focuses on software development including project management, XP is more oriented towards engineering practices [8]. Cloud Computing can use local and distributed environments, and there is a need to know the practices in both. Table 2.7 shows the different practices for XP and Scrum methods. Some of these practices can be used for both the local and global environments, while others need to adapt themselves to the global environment.

Table 2.7: Scrum and XP Practices.

XP	**Local(L)** [7], [35], [8], [53] **Global(G)** [54], [55], [56]	**SCRUM**	**Local** [8], [57], [53], [58] **Global** [54], [59], [60]
1. Planning game	**L**	1. Sprint planning meeting	**L/G**
2. Pair programming	**L**	2. Distributed sprint planning meetings	**G**
3. Continuous Integration	**L**	3. Sprint Backlog	**L/G**
4. On-Site Customer	**L**	4. Product backlog	**L/G**
5. Testing	**L/G**	5. Sprint Review meeting	**L**
6. Refactoring	**L/G**	6. Daily scrum meeting	**L/G**
7. Small releases	**L/G**	7. Retrospective meeting	**L/G**
8. Collective ownership	**L/G**	8.Separate backlogs for each team	**G**
9. Metaphor	**L/G**	9. Scrum of scrum	**L**
10. 40-hour week	**L/G**	10. Distributed Scrum of Scrum	**G**
11. Coding standards	**L/G**	11. Sprint demo	**L/G**
12. Simple design	**L/G**	12. Two-week maintenance sprints	**G**

RQ6. What are Agile practices (XP/Scrum) that are used for maintenance?

In the previous question, we inquired about XP and Scrum practices. In this question, we aim to investigate Agile practices for maintenance. Agile practices that are used in the maintenance process are reviewed according to the related literature. Table 2.8 provides the Agile practices that have been adopted for maintenance.

Table 2.8: Maintenance Practices.

Agile practices	Agile method	Sources
1. Planning game	**XP**	[42] ,[7], [35], [40]
2. Small releases	**XP**	[42], [7], [35], [40]
3. Refactoring	**XP**	[42], [7], [35], [40]
4.Pair Programming	**XP**	[42], [7], [35], [61]
5.Collective code ownership	**XP**	[42], [7], [35], [61], [8]
5. Continuous integration/ Automated Release	**XP**	[42], [7], [35], [40]
6. Test-Driven Development	**XP**	[42], [40]
7. Iteration Planning	**XP**	[42], [40]
8.40-hour week	**XP**	[42], [7], [35], [40]
9. Standup meetings	**XP-Scrum**	[42], [7], [40]
10. Coding standards	**XP**	[42], [7], [35]
11. Onsite Client	**XP**	[42], [35], [61]
12. Metaphor	**XP**	[42], [35]
13. Simple Design	**XP**	[7], [35]
14. Automated testing	**XP**	[35]
15. User Stories	**XP-Scrum**	[8], [42]
16. Task Prioritization	**XP-Scrum**	[5], [61]
17. Task Board	**XP-Scrum**	[42], [7], [35]
18. Product Backlog	**Scrum**	[5], [6], [8]

Table 2.8 (cont'd). Maintenance Practices.

Agile practices	Agile method	Sources
19. Retrospective	**Scrum**	[6], [35]
20. Scrum of Scrum	**Scrum**	[6], [8]
21. Planning meeting	**Scrum**	[8], [6], [42]
22. Demos	**Scrum**	[6], [35]
23. Unit test	**XP**	[6], [35]
24. Acceptance test	**XP**	[6], [8]
25. Small team	**XP-Scrum**	[7], [35], [42]
26. Velocity	**XP-Scrum**	[6], [61]
27. Release planning	**XP-Scrum**	[6], [7]
28. Scrum master	**Scrum**	[6], [8]

RQ7. What are the advantages of applying Agile practices for maintenance?

Table 2.9. shows the impact of Agile practices that lead to achieving the benefits of using Agile for maintenance according to the literature studies.

Table 2.9: Positive Influences for Agile Practices [11], [39], [35], [44].

Set of Agile practices	Impact on operation
1. Planning Game 2. Task Prioritization 3. Collective Code Ownership 4. Automated Testing 5. Stand-up	Improve the morale among team members.
1. Planning Game 2. Acceptance Test 3. Demo	Increased customer satisfaction.

Table 2.9 (cont'd). Positive Influences for Agile Practices [11], [39], [35], [44].

Set of Agile practices	Impact on operation
1. Planning Game 2. Task Prioritization 3. Collective Code Ownership 4. Automated Testing 5. Stand-up meeting	Using these practices will lead to improving productivity.
1. Pair-Programming 2. Code Reviews 3. Coding Standards 4. Small Team	Simplified knowledge transfer among the teams.
1. Task Prioritization 2. User Stories 3. Product Backlog	Improved Prioritization of tasks.
1. Unit Testing 2. Continuous Integration	Fix bugs and merge them with code rapidly.
1. Coding Standards 2. Code Reviews 3. Daily Builds	Increase the product quality.
1. User Stories 2. Product Backlog 3. Velocity	Increase the accuracy of time and efforts forecasts.
1. User Stories. 2. Task Prioritization 3. Refactoring	Enhanced test suites.
1. Refactoring 2. Coding Standards 3. Iteration Planning	Increasing the index of maintainability.
1. Retrospective 2. Refactoring	Improved program comprehension.
1. Pair-Programming	Improve designing at the abstraction level.

RQ8. What are the challenges in using Agile maintenance for a local and global environment?

In this section the challenges faced during maintenance process for local environments are reviewed and possible solutions stated in the literature are identified. (See Table 2.10).

Additionally, possible solutions provided by previous studies are listed. Based on the literature, three main challenges are reported for the global environment as mentioned in [60], [54], [56], [62], and [63]; these issues are further reviewed and possible solutions are identified accordingly (See Table 2.11).

Table 2.10: Agile Maintenance Challenges in Local Environment.

Agile maintenance challenges	Possible solutions
1. Iterative development: Applying tasks within an iteration (sprint) is common in Agile methods. In maintenance, this leads to lost synergy and maintenances objectives. This is due to the emergency tasks in maintenance that causes interrupting the current sprint [5], [7], [9], [11].	According to a study [11], which applied at Aveva company, to improve the performance of sprint, they encouraged the maintenance personnel to compare the sprints between team members.
2. Focusing on work objectives: One of the maintenance challenges is to focus on the current objective, due to the interruption caused to emergency requests, which loses and weakens focus on objectives [5], [9], [11].	One possible solution stated in [11], is to allocate some time for unpredictable tasks in sprint planning. In [36] proposed a model that treats with an urgent request, by pausing the current sprint after checking this request if it is corrective maintenance and storing the current sprint in the version control system to resume this sprint after handling the urgent request.
3. Team working closely: One of the maintenance challenges is when software maintainers work on individual tasks according to their project and acquire experience in one perspective instead of acquiring experience in multi-domain [5], [11], [64].	In [11], Aveva maintainers have developed an informal method, which provides expert assistance to the team instead of using pair programming.
4. Close customer involvement: In maintenance, it is a good idea to involve software maintainers with in-site customers, but it is difficult to apply this action. Thus, it is difficult to achieve closed customer's involvement. [5], [11], [64].	Giving customers who have a good relationship with the team more responsibilities in dealing with difficult problems and evaluation. This you will be able to engage the customer in a more effective manner [11].

Table 2.10 (cont'd). Agile Maintenance Challenges in Local Environment.

Agile maintenance challenges	Possible solutions
5. Face-to-face communication: In maintenance, the tasks are diverse, for that reason face-to-face communication is considered as a needless action [5], [11], [64].	In [11], they solved this challenge through intensive preparation by the Scrum master before the meeting. Thus, the team will regain confidence in the engineers' estimates.
6. Light documentation: In maintenance, documentation is considered as an irreplaceable part that helps maintainers to understand the system. The Agile method emphasizes on reducing documentation (light documentation) as possible as [5], [11], [37], [44], [46].	Simplify document and organize it in structure manner to include useful information and avoid heavy documentation that conflicted with Agile principles [5], [11].
7. Frequent testing: Usually, comprehensive system testing is considered impractical, because this kind of testing is limited and valid only for the current fix. The Agile method focuses on automated testing, but it is difficult to apply that for maintenance, so maintainers need to find proper solutions that affect positively the quality of the product [5], [11].	One of the effective solutions to moving to collective ownership, which helps engineers to test others' fixes, doing that will encourage co-education between team members [11].
8. Motivation through collective ownership: collective ownership is difficult to apply since maintainers work on individual tasks, i.e., there are no common tasks between each other, thus getting motivation is not easy [5], [11].	Devoting knowledge sharing among team members and making it a regular practice process [5], [11].
9. Knowledge transfer through openness: it might be no benefits of Openness and information sharing if there is no interconnection of tasks. Therefore, there is no transfer of knowledge in this case [5], [11].	It is necessary to find a possible way to transfer knowledge between the maintainer's team, one of these ways according to [11], is to apply the traditional Scrum board with some modification.

Table 2.11: Agile Maintenance Challenges for The Global Environment.

Agile maintenance challenges	Possible solutions
1. **Problems regarding Communication**: several problems existed such as misunderstanding problems, difficulties with a face-to-face meeting, and the high cost of the communications [42], [60], [62], [63], [65], [54], [66].	- Summarize details about tasks, thus will increase the communication speed. - Organize meetings between product owners and Scrum master. - Simplify communication between team members using Wiki pages or any other tools. - Keep up-to-date information between team members. - Determined end meeting time. - Specify local Scrum master as an alternative product owner [63], [67].
2. Problems regarding control: most common problems related to control are difficulties related to quality control and process, rearrangement of priorities, difficulties related to cooperation [60], [63], [65],[66], [68].	- For multiple backlogs, we can use Scrum of Scrums for high-level coordination. - Organize meetings according to time zones for each team. - Specify backlog for each region. - Synchronize each local backlog at the end of the day. - Use tracking bugs and tasks system. -Training the team on Agile principles and practices. - Discussion about current sprint progress and backlog three times weekly [63], [67].
3. Problems regarding trust: There are two main factors; trust among team members and team morale [56], [62], [63], [65], [68].	- Adapt the Scrum process, regarding with place value for people. - Use Instant Messaging (IM) to simplify communication between the team. - Enhance interconnectedness and collaboration through face-to-face interaction [63], [69], [67].

RQ9. What are the advantages of applying Agile methods in a Cloud Computing environment?

In this section, the benefits of applying Agile development in a Cloud Computing environment will be reviewed. According to the literature, development and maintenance differ from each other in detail, but they have the same phases in general [25], [50], [23]. Therefore, knowing the benefits of using development in Cloud will motivate and inspire us to apply the Cloud platform with software maintenance. There are several benefits of using Agile methods in the Cloud Computing environment. These are:

- **Quality factors:** Cloud Computing helps to facilitate Agile development and maintain quality throughout the development process. The quality factors are divided into sub-factors such as scalability, maintainability [29], which affect and accelerate Agile development in a Cloud Computing environment.

  The first factor is scalability, which means the ability to deals with a large team, through managing backlog, sprint, and feedbacks between customers and developers. Cloud Computing provides the infrastructure and resources which are required to manage a large team. Another perspective of scalability in Cloud Computing is automation, which is the ability to provide resources according to the customers' needs using scale-up or scale-down [13], [12], [29], [70], [71].

  A second factor of the quality attribute is maintainability. Cloud Computing reduces the running cost to maintain software by eliminating the need to hire more expert engineers to manage servers. Cloud Computing provide services/resources in terms of Infrastructure, Platform, Software, and everything as a service (X as a service), PaaS provides hardware resources and tools that enable developers to develop and maintain their projects for both local and global environments [29]. SaaS provides services for automating updating, simplifying administration, and patch management. All these services are based on Pay-As-You-Go. In addition, SaaS provides users the ability to share the codes among team members in the global environment using GitHub, CodeSpaces, SourceForge, and so on [29]. Another service of a Cloud platform is Test as a service (TaaS), which provides tools to simplify the test process [29]**.** laaS provides infrastructure for users with full control for hardware, operating system, network, data, and servers with low cost, less effort, and

effective service. Also, laaS provides both physical and virtual servers [29]. Due to the availability of the infrastructure, the process of maintaining, testing, and updating the programs will be easier due to the reduction of the time and effort spent during these processes [72].

- **Development infrastructure**: Cloud Computing provides information technology resources that can be assigned to customers wherever they are if the Internet is available. Thus, Cloud Computing can be a good environment for distributed development teams. For this reason, many organizations have employed Cloud environments to achieve their projects [29]. Cloud computing platforms provide many hardware and software capabilities that facilitate the software development process. Also, Cloud Computing provides storage, virtualizations, and networks. All these resources can be available rapidly without the need to interact with service providers [73].
- **Collaboration:** It is one of the significant principles of the Agile approach. Specifically, in a distributed environment, it is considered an essential activity among the team to reduce the time and efforts needed for development. There are several forms of cooperation represented by sharing the documents among team members, such as test reports, SRS documents, prototypes, sprint planning, backlog planning, stand-up meeting, and Scrum of Scrum. Clouds provide tools that can simplify development, for instance Cloud IDE, GitHub, Eclipse, and others [12], [29], [74], [73]. Additionally, communication among teams is considered as a way of collaboration used for sharing and discussing information through the available tools, such as Wikis, Skype, emails, etc. [13], [29], [70].
- **Regard to transparency:** In the Agile method, many activities that indicate transparency, such as burn-down charts, user story, and product backlog, these activities have to be visible to team members in the Scrum method to predicate the next sprint and future tasks. Therefore, any defects discovered by the team need to be addressed according to the priorities listed in the backlog. In a Cloud Computing environment, all these activities can be achieved with the support of project management tools, such as JIRA, Mingle, Rally, ScrumWorks, Trac, VersionOne, Xplanner, AgileFant, Stats, jmxtrans, Metrics, Esper, Ganglia, Graphite, Cube, CloudSpoke on Topcoder, Trustie, and REDMINE, not to mention code management tools including Code Spaces, GitHub, GoogleCode,

SourceForge, Unfuddle, GitLab, BitBucket, Git, and Mercurial [12], [29], [70]. More advantages for combination Agile and Cloud Computing are listed in Table 2.12.

Table 2.12: The Benefits of Using Agile in The Cloud

Factors	Advantages
1. Supporting software testing.	Software developers do not care about managing, maintaining, and plan hardware resources, such as the number of servers necessary for the testing process [12], [74], [70]
2. Virtualization	Virtualization helps to support parallel development required for small iterations in Agile development. In addition, users can scale up the number of processors and storage, which are considered essential factors for the parallel environment [12], [70].
3. Traceability	Traceability is one of the most important features in Cloud Computing as it enables developers to observe the changing code globally by all team members [12], [70].
4. Prototypes and Demo	In the Cloud, deploying prototypes and sharing them with customers is easier than in traditional environments [12].
5. Performance	Due to the nature of the Cloud that supports developing software in a parallel manner, establishing the principle of decentralization in decision-making among team members, it naturally leads to improving overall performance [12], [70], [71], [75].
6. Reduce cost	Cloud computing works on the "pay-as-you-go" principle, which helps to reduce Agile development method expenditures [12], [70], [71], [75].

CHAPTER 3

RESEARCH METHODOLOGY

3.1. Introduction

There are many methods of scientific research, but the most popular ones are quantitative and qualitative. Each is used in a specific scope, as will be further explained in what follows.

Quantitative research is a method that collects and analyzes data based on quantities [76]. The sample of this type of research includes people, groups, and cases. The quantitative research sample has a limited number of people, cases, and groups [77]. Additionally, this type of research tends to focus on statistics and numbers [76], [78]. According to [79], it is difficult to apply a quantitative approach in software engineering due to some limitations, such as the sample size.

Another popular type of scientific research is qualitative research, which is used for non-numerical data such as videos, audio, and texts. These are collected and analyzed to understand practical experiences, concepts, and expert opinions. Qualitative research is used if there is a need to innovate theory or test it [80].

3.2. Qualitative Research

There are two main types of qualitative data analysis (QDA): observations and interviews. However, there are some other methods that fall under this category, such as case studies, surveys, and historical and document analyses. The term ‘qualitative research’ is a comprehensive umbrella that refers to designs of theoretical perspectives, such as narrative, phenomenology, grounded theory, action research, case study, ethnography, historical research, and content analysis [81]. Table 3.1 briefly explains the different qualitative research types.

Table 3.1: Qualitative research types

Qualitative Type	Description	References
1. Narrative	In this method, the characteristics of the narrative text are analyzed.	[81], [82]
2. Phenomenology	This type of qualitative research focuses on the common factors pertaining to a particular group for the purpose of studying the nature of this phenomenon.	[81]
3.Grounded theory GT	This type focuses on generating a theory by collecting data systematically and analyzing it to reveal common behaviors and relationships within a certain group.	[81], [83]
4. Action research	This investigative method combines theory and action to gain access to organizational knowledge and address existing challenges.	[81], [84]
5. Case study	This method is used to study a phenomenon within a group or individual, program, event, activity, and process.	[81]
6. Ethnography	This method depends on the researcher's observation and the interaction in the participant's environment. The group behaviors should be studied, described, and analyzed by the researcher.	[81], [84], [85]
7. Historical research	This method describes how the study develops, what it achieves, how it begins, and the current status of the research.	[81], [86]
8. Content analysis	This method depends on the researcher's analysis of written messages, audio, and video recordings.	[81], [87]

According to [88], the main characteristics of qualitative research are: the production of oral data and the method of data collection that could be done through interviews and observations. The results are then analyzed based on the participants' views related to the subject in question.

3.3. Survey Process

Survey is defined as a way of seeking knowledge, attitudes, and behavior, and they could be done through comprehensive research for information and then description and analyses [89].

The reason behind choosing this method in the present work is that survey tend to be more objective and commensurate with the nature of our study. After stating the challenges in Systematic Mapping (Chapter 2) and investigating more about the Agile software maintenance challenges, a survey is carried out through direct communication with software professionals. According to Kitchenham [34], there are six basic steps for doing such surveys which will be followed in this study as highlighted in Figure 3.1.

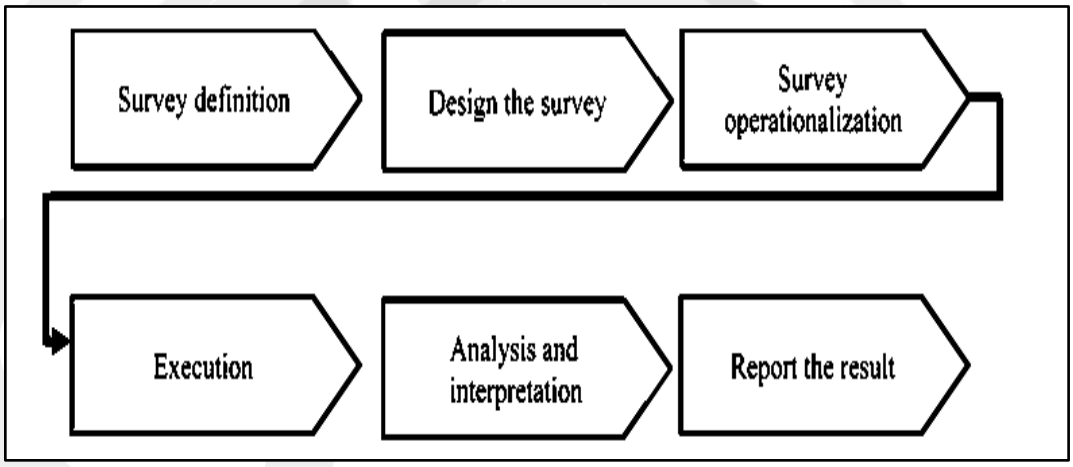


Figure 3.1: Survey Process [34].

3.3.1. Define and Classify Challenges

The main challenges with respect to the present work are classified, based on the Systematic Mapping in Chapter 2) and according to quality factors in order to generate the survey questions. See Table 3.2 and Table 3.3.

Table 3.2 : Challenges in the local environment

Agile maintenance challenges	Classify challenges according to quality factors
1. Iterative development	**Traceability**: Lack of traceability leads to loss of some project tasks. **Transparency:** Lack of transparency in a project leads to unclear views concerning that project. **Manageability requirements**: Lack of manageability leads to conflict in the sprint.
2. Focusing on work objectives	**Transparency:** Lack of transparency in the project leads to unclear views regarding the project.
3. Close team work	**Transparency:** Lack of transparency in the project can affect team collaboration. **Collaboration**: Lack of collaboration results in poor teamwork.
4. Close customers involvement	**Collaboration and Communication:** Lack of collaboration and communication between the customers and the teams will affect the project's quality**.** **Manageability requirements**: Lack of manageability could lead to conflict between customers' requirements and the maintenance team.
5. Face-to-face communication	**Collaboration and Communication:** Lack of collaboration and communication between the customers and the teams can affect the project’s quality.
6. Light documentation	**Manageability requirements**: Lack of manageability could lead to conflict between the customers' requirements and the maintenance team. **Traceability**: Lack of traceability could lead to losses in documentation.
7. Frequent testing	**Testability:** Lack of test tools could lead to a lack of testability.

Table 3.2 (cont'd). Challenges in the local environment

Agile maintenance challenges	**Classify challenges according to quality factors**
8. Motivation through collective ownership	**Collaboration and Communication:** Lack of collaboration and communication between teams will discourage collective ownership.
9. **Knowledge transfer through openness**	**Collaboration and Communication:** Lack of collaboration and communication between teams will discourage knowledge transfer. **Transparency:** Lack of transparency in a project is likely to affect knowledge transfer among the maintenance team members.

Table 3.3: Challenges in the global environment

Agile maintenance challenges	**Classify challenges according to quality factors**
1. Challenges regarding Communication	**Communication:** Lack of communication between teams will lead to obstacles in the collaboration within a global environment. **Manageability**: Lack of manageability will lead to conflicts in performing tasks.
2. Challenges regarding control	**Manageability requirements**: Lack of manageability will lead to poor control over the project. **Transparency:** Lack of transparency in a project is likely to affect the degree and quality of control over a project.
3. Challenges regarding trust	**Communication**: Lack of communication will lead to a lack of trust. **Collaboration:** Lack of collaboration between teams will lead to a lack of trust. **Manageability:** Excessive monitoring can lead to low trust.

Using the Cloud Computing platform as a tool to simplify the maintenance process may help to overcome challenges as stated in SM. Therefore, a survey is prepared and applied in an industrial environment that uses Agile in its maintenance process in order to obtain the practitioners' feedback about challenges that are commonly faced during the maintenance process.

3.3.2. Design Process

3.3.2.1. Survey Instrument

After the related studies are reviewed, a survey is carried out to determine the issues faced by Agile Maintenance teams. We used the survey method to gather the necessary data to support the theoretical research on the practical side regarding the research questions. In order to create the survey, we looked at the techniques and measurements used in past studies that were relevant to the present work. The survey includes a set of terms that are directly related to the study goals and questions and, hence, supports the research subject. See Appendix A.

There are many tools used for conducting a survey; one is the Google Forms - a simple survey software that can be used for multiple purposes, such as designing and analyzing surveys without the need for prior programming knowledge. All is needed is to create a Gmail account to sign in. The survey is developed using Google Forms for publishing retrieving the data. See Figure 3.2, which represents the main interface of the survey.

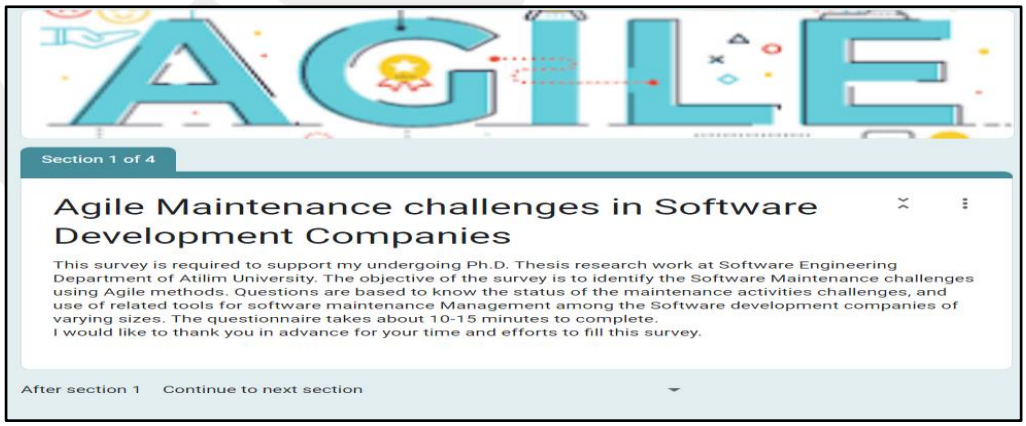


Figure 3.2: The Survey interface in Google Forms.

This survey follows the instructions by Burgess in [90], for designing as per three components:

(1) Determine survey sections.
(2) Determine the question formula for each section.
(3) Design the sequence of the questions for each section.

1) Determine survey sections

The survey questions are determined as per the challenges identified in Chapter 2 related to the distributed (global) environment and the local (on-site) environment.

2) Determine the question formula for each section

We selected the multiple-choice questions for this purpose, followed by an open question for the responders at the end of each section.

3) Design the sequence of the questions for each section.

In order to arrange the survey according to Agile challenges identified in literature studies, it was divided into three sections:

- Section A - General Information
- Section B - Agile software maintenance challenges
  - B1 Manageability
  - B2 Scalability
  - B3 Software Infrastructure
  - B4 Communication and Collaboration
  - B5 Transparency
- Section C - General question

The survey was sent to some experts and professionals as a pilot test. Then, the last version was deployed after many iterations by taking into consideration key comments. Figure 3.3 illustrates the steps for design and deployment of the survey.

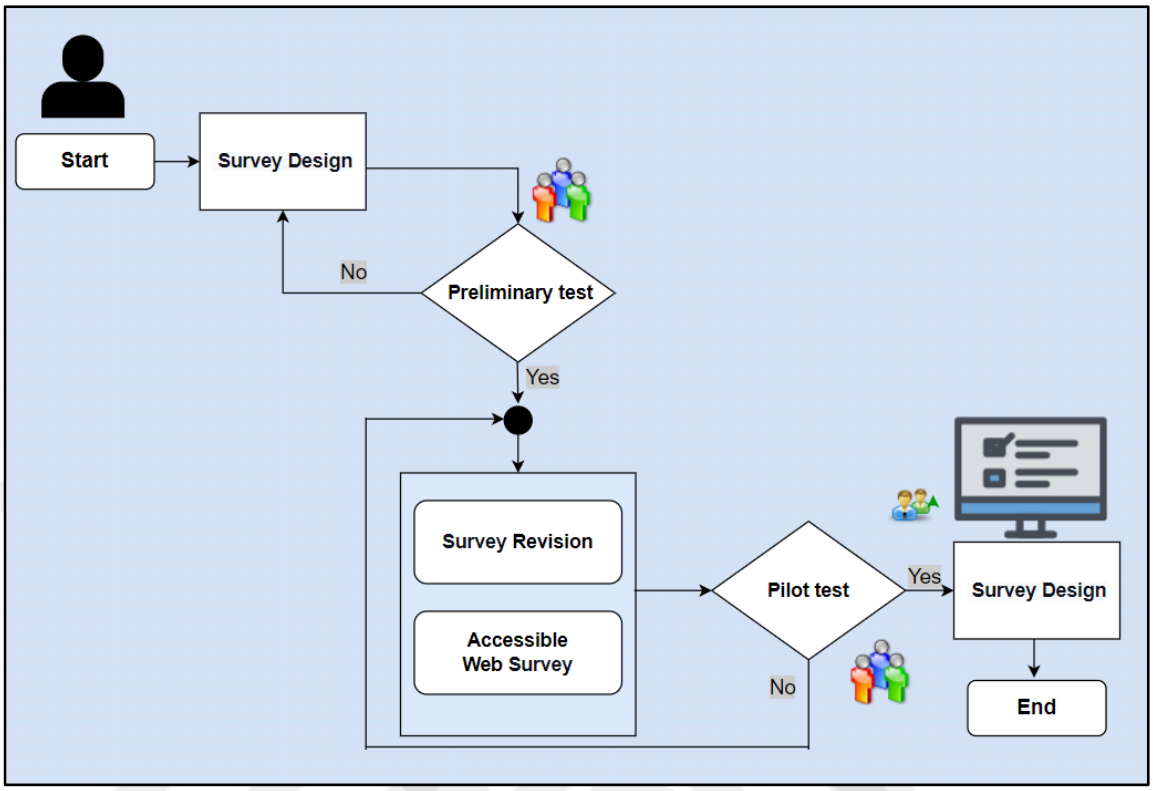


Figure 3. 3 : Steps for Designing and Deploying the Survey

CHAPTER 4

SURVEY ANALYSIS

4.1. Introduction

In this chapter, the data are reviewed as obtained from the respondents and, then, verified in terms of validity and reliability, followed by identifying the research technique, research community, research sample, and the research instrument. Additionally, these data are analyzed using appropriate statistical analysis methods. In order to achieve precision in the analysis, we include a degree of interpretation for these results; for this purpose, measurements, classifications, and interpretation methods are used to extract the key conclusions. The descriptive analytical method is employed to achieve the objective of the study.

4.2. Research Sample

The sample targeted in the survey are organizations that use Agile maintenance activities. After deploying the survey using Google Forms, the responses were collected, and the total sample size was found to be 56 participants, 42 of which (i.e., 75%) work at organization using Agile for maintenance.

The following sections are a comprehensive description of the research sample, described by the required statistical methods according to several demographic variables, which include country, experience, organization size and nature, use of Cloud, duration of using Agile, and the maintenance process.

4.2.1. Characteristics of the sample study

The sample size is calculated according to many characteristics described in the following steps.

1): Distribution of the study sample according to Country

Table 4.1 illustrates the distribution of respondents by country.

Table 4.1: Distribution of the study sample according to country

Country	Frequency	Percent
Brazil	1	2.4
India	8	19
Iraq	1	2.4
Jordan	5	11.9
Pakistan	22	52.4
Turkey	4	9.5
United Kingdom	1	2.4
Total	42	100

The results are as follows: 2.4% from Brazil, 19% from India, 2.4% from Iraq, 11.9% from Jordan, and 52.4% from Pakistan, as the largest category; followed by 9.5% from Turkey and 2.4% from the United Kingdom. Figure 4.1, shows the distribution of the results among different countries.

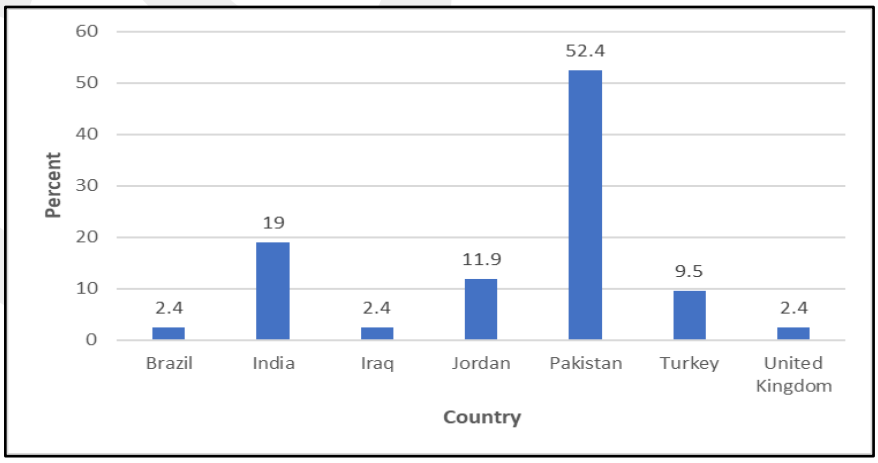


Figure 4.1: Distribution of the study sample according to country

2): Distribution of the study sample according to experience

Table 4.2 illustrates the study sample according to experience.

Table 4.2: Distribution of the study sample according to experience

Experience	Frequency	Percent
0-4 (years)	12	28.6
5-7 (years)	10	23.8
8-10 (years)	11	26.2
More than10 (years)	9	21.4
Total	42	100

Based on the results, 28.6% of the professional's experience was between 0 and 4 years, 23.8% 5 and 7 years, 26.2% 8 and 10 years as the largest, and finally 21.4% over 10 years. Figure 4.2, shows the corresponding diagram.

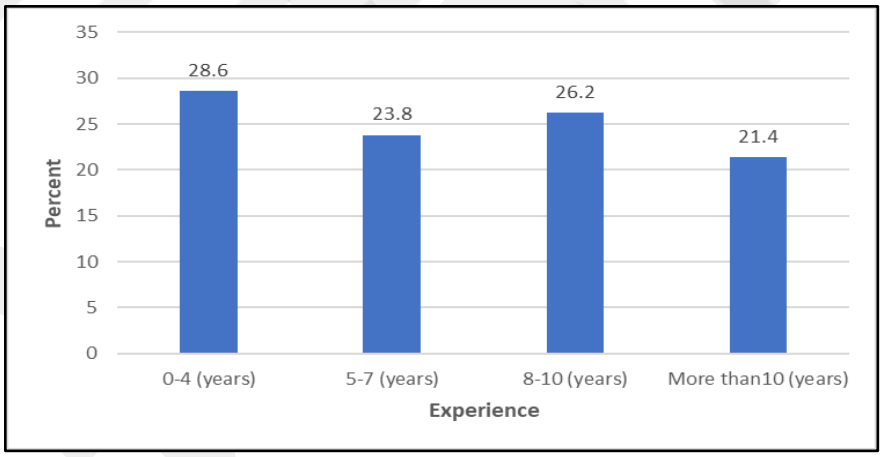


Figure 4.2: Distribution of the study sample according to experience.

3): Distribution of the study sample according to Organization size

Table 4.3 illustrates the distribution of study sample according to the organization size.

Table 4.3 : Distribution of the study sample according to organization size

Organization size	Frequency	Percent
Micro (<250 employees)	21	50
Small (<1500 employees)	12	28.6
Medium (<5000 employees)	2	4.8
Large (more than 5000)	7	16.7
Total	42	100

Accordingly, 50% of the professionals work at micro-size organizations, which is the largest category of the sample, 28.6% at small-size organizations, 4.8% at medium-size organization, and 16.7% at large-size organizations. Figure 4.3 shows the distribution of the sample according to organization size.

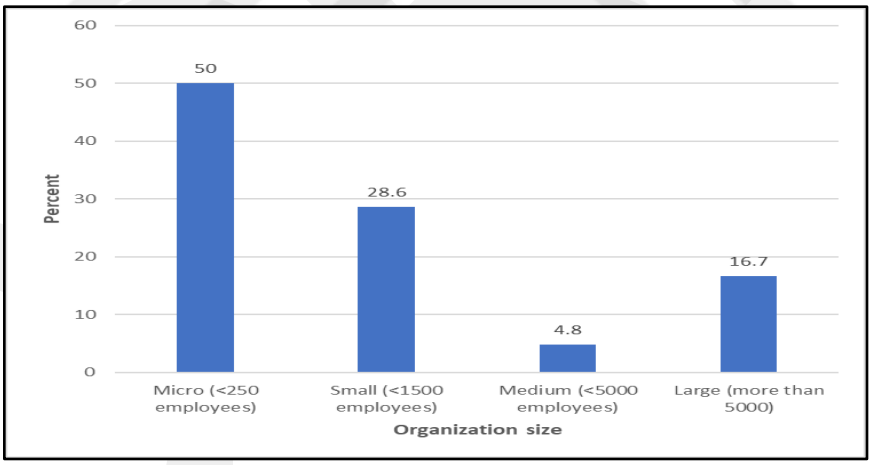


Figure 4.3: Distribution of the study sample according to organization size

4): Distribution of the study sample according to organization's nature

Table 4.4 illustrates the distribution of study sample study according to the nature of the organizations.

Table 4.4: Distribution of the study sample according to organization's nature

Organization's nature	Frequency	Percent
National	5	11.9
International	35	83.3
Other	2	4.8
Total	42	100

On this basis, it is seen that 11.9% of the sample is employed with national organizations, and 83.3% at international organizations, which is the largest category of the sample, and 4.8% in other categories. Figure 4.4 shows the diagram results.

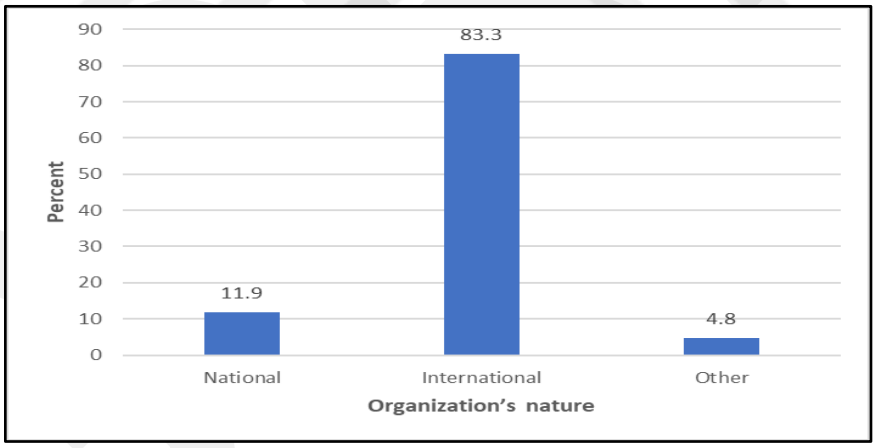


Figure 4.4: Distribution of the study sample according to organization's nature

5): Distribution of the study sample according to the duration of using Agile

Table 4.5 illustrates the distribution of study sample according to the duration of using Agile for maintenance.

Table 4.5: Distribution of the study sample according to the duration of using Agile

long of use Agile	Frequency	Percent
0-4 (years)	22	52.4
5-7 (years)	7	16.7
8-10 (years)	6	14.3
More than10 (years)	7	16.7
Total	42	100

It is evident that 52.4% of the sample have been using Agile for 0 to 4 years, which is the largest category of the sample, 16.7% for 5 to 7 years, 14.3% for 8 to 10 years, and 16.7% have done so or more than10 years. Figure 4.5 shows the results in diagram form.

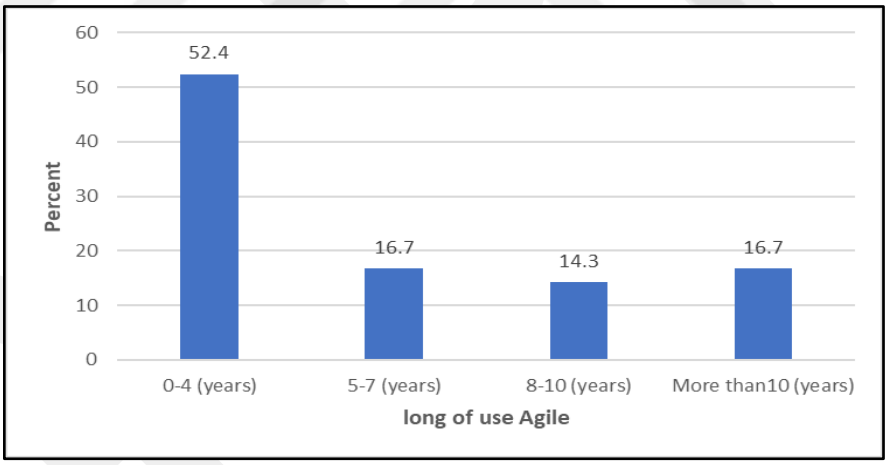


Figure 4.5: Distribution of the study sample according to the duration of using Agile

7): Distribution of the study sample according to the maintenance process.

Table 4.6 illustrates the distribution of the sample according to the maintenance process.

Table 4.6: Distribution of the study sample according to the maintenance process

Maintenance process	Frequency	Percent
On-site (local).	5	11.9
By remote access to the customer servers (global).	9	21.4
Depending on the nature of the fault, use local or global.	24	57.1
Support	2	4.8
Local, company-remote, or cloud-provided remote	2	4.8
Total	42	100

It can be observed that 11.9% of the professionals work as on-site or local, 21.4% by remote access to the customer servers or global, 57.1% depend on the nature of the fault and use either local or global, which is the largest category of the sample, 4.8% operate as support, and finally 4.8% choose either, local, company-remote, or cloud-provided remote. Figure 4.6 shows the results.

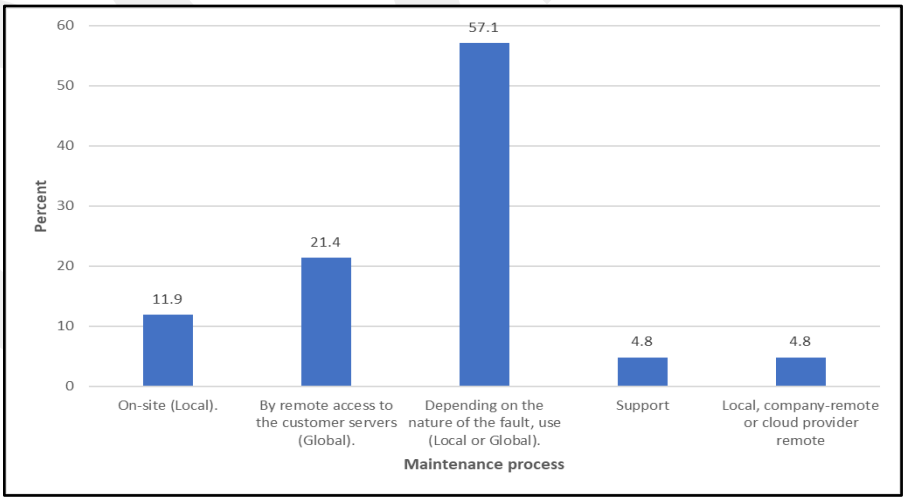


Figure 4.6 : Distribution of the study sample according to maintenance process

4.3. Survey Validity

After the preparation step, the survey is reviewed to check the validity and reliability before analyzing the results. For this, the steps illustrated in Figure 4.7 are followed.

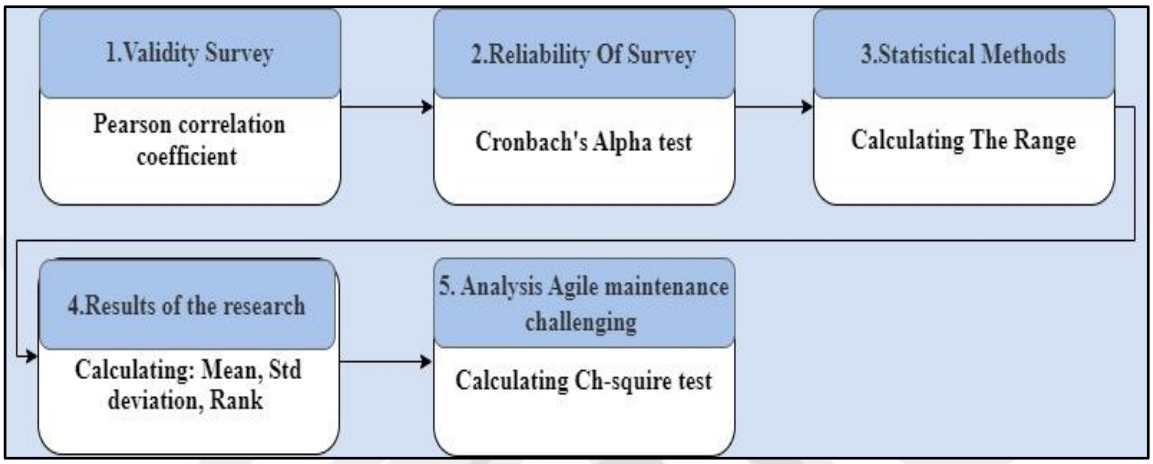


Figure 4.7: Analysis steps

To validate the internal consistency, the Pearson correlation coefficient is calculated between each paragraph and its corresponding axis, as seen in Table 4.7.

Table 4.7: Pearson correlation coefficient between the paragraph and the dimension for questions.

No	correlation coefficient	No	correlation coefficient	No	correlation coefficient	No	correlation coefficient
1	.897**	6	.811**	11	.872**	16	.703**
2	.799**	7	.705**	12	.727**	17	.863**
3	.786**	8	.859**	13	.742**	18	.770**
4	.618**	9	.796**	14	.679**	19	.704**
5	.704**	10	.607**	15	.731**	20	.647**

** All correlations among the variables are above 0.6, which consider a High Correlation according to the below Table 4.8:

Table 4.8: The scale of Pearson's Correlation Coefficient [91].

Scale of correlation coefficient	Value
$0< r \leq 0.19$	Very Low Correlation
$0.2 \leq r \leq 0.39$	Low Correlation
$0.4 \leq r \leq 0.59$	Moderate Correlation
$0.6 \leq r \leq 0.79$	High Correlation
$0.8 \leq r \leq 1.0$	Very High Correlation

Table 4.9: Internal consistency of the five axes

No	Factors	No of Items	Correlation Coefficient
1	Manageability	4	0.909
2	Scalability in Agile Software Maintenance	4	0.949
3	Software Infrastructure	5	0.979
4	Communication and Collaboration	3	0.888
5	Transparency	4	0.952

As stated in Table 4.9, all correlation values among the variables are above 0.9, which consider a very high correlation according to Table 4.8. The correlation between the two variables is noteworthy (2-tailed), and, according to [91], this is an acceptable score. As per the Pearson correlation coefficients between each paragraph and the table, there is excellent validity regarding the internal consistency of the first axis' words.

4.3.1. Calculate the Exploratory Factor Analysis (EFA), and Average Variance Extracted (AVE).

The Kaiser–Meyer–Olkin (KMO) and the Bartlett's Test are applied for the axis, as shown in Table 4.10 representing the results for Manageability.

Table 4.10: KMO and Bartlett's Test for Manageability

Kaiser-Meyer-Olkin Measure of Sampling Adequacy.		**0.715**
Bartlett's Test of Sphericity	Approx. Chi-Square	25.038
	Df	6
	Sig.	<.001

The results in the above table reveal the value of the Bartlett sphericity test (25.038), which indicates that the correlation is significant at a level of (< 0.001), which means the factor model is suitable for explaining the data [92]. The KMO score of 0.715 is above 0.5, according to Kaiser [93]; thus, the score is considered as acceptable.

Table 4.11 shows the Exploratory Factor Analysis (EFA) used to show construct reliability and validity.

Table 4.11: Exploratory Factor Analysis (EFA) Result of Manageability

Variable	**Items**	**Factor Loading**	**AVE**
Manageability	P1	0.870	**0.629**
	P2	0.842	
	P3	0.824	
	P4	0.609	
Eigenvalue % of Variance	**2.516**		
Cumulative % of Variance	**62.907**		

Based on the result that appeared in the above table, the factor loadings for the P1, P2, P3, and P4 questions were above 0.5. In addition, the cumulative variance is 62.907%, which is acceptable. The eigenvalue is (2.516), larger than 1, and considered passable as well, according to [94]. The Average Variance Extracted (AVE), as seen in the table, is 0.629, which should be at least 0.5 according to [94], putting the convergent validity within the normal range as such. See Figure 4.8 in the following.

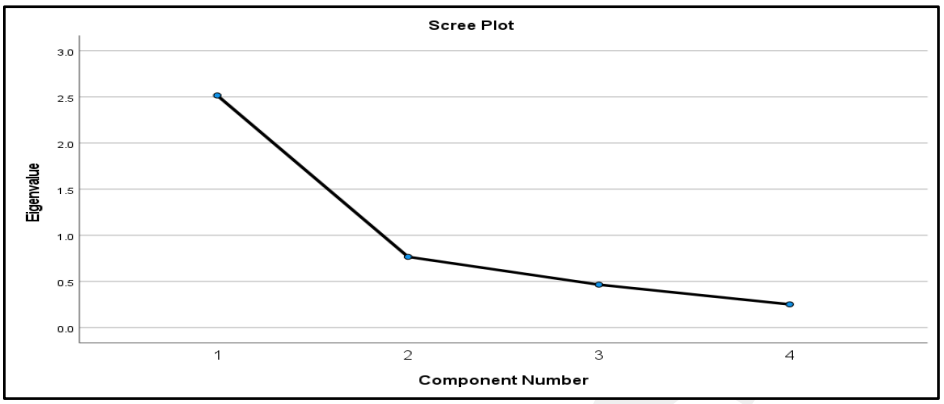


Figure 4.8: EFA Result of Manageability

We calculate the Kaiser–Meyer–Olkin (KMO) and Bartlett's Test for axis, Table 4.12 presents the results for Scalability.

Table 4.12: KMO and Bartlett's Test for Scalability in Agile Software Maintenance

Kaiser-Meyer-Olkin Measure of Sampling Adequacy.		**0.559**
Bartlett's Test of Sphericity	Approx. Chi-Square	38.477
	Df	6
	Sig.	<.001

The results in the above table show that the KMO score is 0.559, above 0.5, according to Kaiser [93]; the score is considered as acceptable. The value of the Bartlett sphericity test (38.477), which indicates that the correlation is significant at a level of (< 0.001), which means the factor model is suitable for explaining the data [92].

Table 4.13, shows the factor loading and the Average Variance Extracted (AVE) used to show construct reliability and validity.

Table 4.13: EFA Result of Scalability in Agile Software Maintenance

Variable	**Items**	**Factor Loading**	**AVE**
Scalability in Agile Software Maintenance	P5	0.895	**0.653**
	P6	0.845	
	P7	0.771	
	P8	0.708	
Eigenvalue % of Variance	**2.61**		
Cumulative % of Variance	**65.25**		

Accordingly, the factor loadings for the P5, P6, P7, and P8 questions are above 0.5. In addition, the cumulative variance is 62.25%, which is acceptable according to [94]. The eigenvalue is (3.812), which exceeds 1 and, as such, is considered passable, as in [94]. The Average Variance Extracted (AVE) is 0.653; this exceeds 0.5, making the convergent validity fall within the normal range [94]. See Figure 4.9 below.

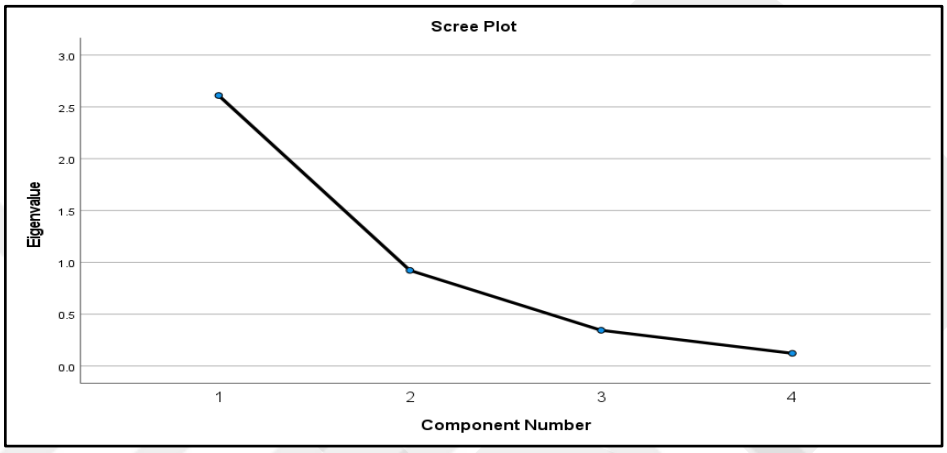


Figure 4.9: EFA Result of Scalability in Agile Software Maintenance

Now, we will calculate the Kaiser–Meyer–Olkin (KMO) and Bartlett's Test for axis, Table 4.14 contains the results of the KMO and Bartlett's Test for the Software Infrastructure.

Table 4.14: KMO and Bartlett's Test for the Software Infrastructure

Kaiser-Meyer-Olkin Measure of Sampling Adequacy.		**0.801**
Bartlett's Test of Sphericity	Approx. Chi-Square	49.426
	Df	10
	Sig.	<.001

It can be observed that the KMO score is 0.801, which is above 0.5, according to Kaiser [93]; thus, the score is considered as acceptable. Furthermore, the value of the Bartlett sphericity test (49.426), which indicates that the correlation is significant at a level of (< 0.001), which means the factor model is suitable for explaining the data [92].

Table 4.15 shows the factor loading along with the Average Variance Extracted (AVE) used to show construct reliability and validity.

Table 4.15: EFA Result of Software Infrastructure

Variable	Items	Factor Loading	AVE
Software Infrastructure	P9	0.879	**0.695**
	P10	0.858	
	P11	0.827	
	P12	0.807	
	P13	0.795	
Eigenvalue % of Variance	**3.476**		
Cumulative % of Variance	**69.527**		

Based on the results in the table, the factor loadings for the P9, P10, P11, P12, and P13 questions are above 0.5. In addition, the cumulative variance is 69.527%, which is acceptable according to [94]. The eigenvalues, as shown, are 3.476, above 1, and considered passable, as mentioned in [94]. The Average Variance Extracted (AVE), as seen in the table, is 0.695; thus, exceeding 0.5 and making the convergent validity fall within the normal range [94]. See Figure 4.10.

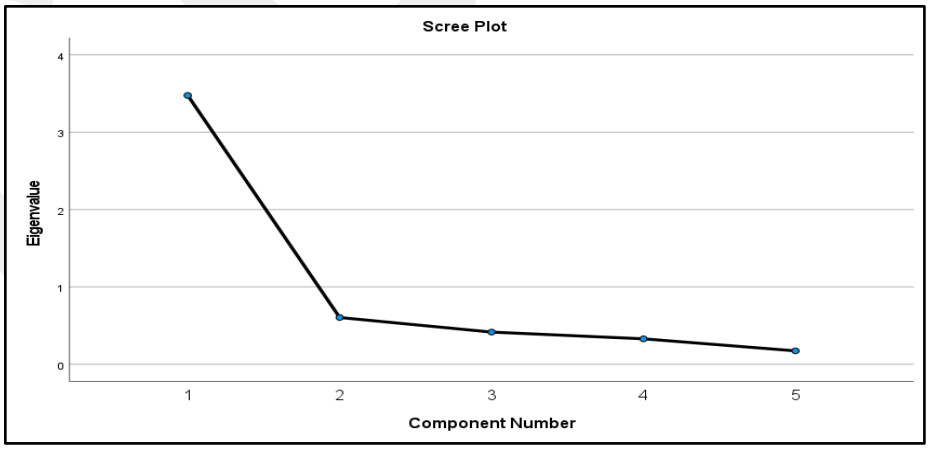


Figure 4.10: EFA Result of Software Infrastructure.

As per Table 4.16, the KMO and Bartlett's Test is calculated for the Communication and Collaboration axis.

Table 4.16: KMO and Bartlett's Test for the Communication and Collaboration

`Kaiser-Meyer-Olkin Measure of Sampling Adequacy.		**0.656**
Bartlett's Test of Sphericity	Approx. Chi-Square	33.217
	Df	3
	Sig.	<.001

The results in the above table show us that the KMO ratio is 0.656, above 0.5, according to Kaiser [93]; the score is considered as acceptable, and the value of the Bartlett sphericity test (33.217) is significant at level (<0.001) which means the factor model is suitable for explaining the data according to [92].

Table 4.17 shows the factor loading as well as the Average Variance Extracted (AVE) used to show construct reliability and validity.

Table 4.17: EFA Result of Communication and Collaboration

Variable	**Items**	**Factor Loading**	**AVE**
Communication and Collaboration	P14	0.951	**0.809**
	P15	0.903	
	P16	0.841	
Eigenvalue % of Variance	**2.428**		
Cumulative % of Variance	**80.919**		

Based on these findings, the factor loadings for the P14, P15, and P16 questions are above 0.5. In addition, the cumulative variance is 80.919%, which is an acceptable ratio according to [94]. The Eigenvalues as shown are 2.428, exceeding 1, and considered passable as Hair et al mentioned in [94]. The AVE result is 0.809, which is larger than 0.5, and makes the convergent validity fall within the normal range [94]. See Figure 4.11.

Figure 4.11: EFA Result of Communication and Collaboration

In Table 4.18, we calculate the KMO and Bartlett's Test for the Transparency axis.

Table 4.18: KMO and Bartlett's Test for the Transparency

Kaiser-Meyer-Olkin Measure of Sampling Adequacy.		**.807**
Bartlett's Test of Sphericity	Approx. Chi-Square	67.359
	Df	6
	Sig.	<.001

The results in the above table show us that the KMO ratio is 0.807, above 0.5, according to Kaiser [93]; the score is considered as acceptable, and the value of the Bartlett sphericity test (67.359) is significant at level (<0.001) which means the factor model is suitable for explaining the data according to [92].

Table 4.19 shows the factor loading along with the Average Variance Extracted (AVE) used to show construct reliability and validity.

Table 4.19 : EFA Result of Transparency

Variable	**Items**	**Factor Loading**	**AVE**
Transparency	P17	0.951	**0.849**
	P18	0.940	
	P19	0.907	
	P20	0.887	
Eigenvalue % of Variance	**3.398**		
Cumulative % of Variance	**84.954**		

According to the result that appeared in the above table, the factor loadings for the P17, P18, P19, and P20 questions were above 0.5. In addition, the cumulative variance is 84.954 %; this ratio is acceptable according to [94]. The eigenvalue as shown is 3.398, which is larger than 1 and considered passable, as mentioned in [94]. The AVE result as seen in the table is 0.849, > 0.5, and the convergent validity is within the normal range [94]. See Figure 4.12 below.

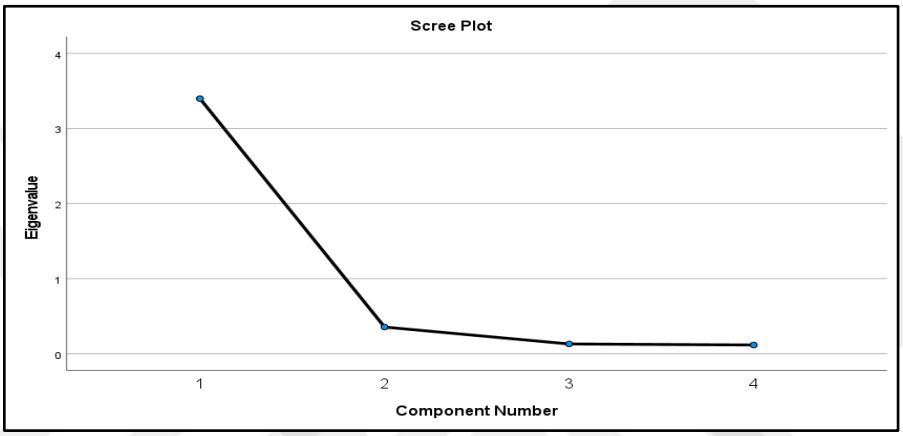


Figure 4.12: EFA Result of Transparency

4.4. Survey Reliability

In order to verify the survey's reliability, the Cronbach's Alpha test is applied. See Table 4.20 which illustrates the result.

Table 4.20: Cronbach's Alpha coefficients

Factors	No of Items	Cronbach's Alpha
Manageability	4	0.794
Scalability in Agile Software Maintenance	4	0.812
Software Infrastructure	5	0.888
Communication and Collaboration	3	0.877
Transparency	4	0.939
Total	20	0.969

From the above table, it is concluded that all Cronbach's Alpha values are above 0.7 and satisfy the reliability of the survey for application and the reliability of its results based on [94].

4.5. Statistical descriptive analysis

In order to identify the challenges related to the Agile software maintenance factors, the mean, standard deviation, rank, and degree for each question is calculated within the dimension using descriptive statistics and the SPSS software. The following illustrates the intervals for the three-point Likert scale according to [95]:

- Low: 1-1.66.
- Medium: 1.67-2.33.
- High: 2.34-3.

4.5.1. Manageability

The calculations begin with the questions related to the Manageability section, and the results are listed in Table 4.21:

Table 4.21: Calculate mean, standard deviation, rank, and the degree for Manageability

I. No	Questions	Mean	Std deviation	Rank	Degree
1	Challenges related to the availability of experts for configuring software and hardware resources and Agile maintenance experts	2.24	.790	1	Medium
2	Lack of technical support for the improvement of Agile maintenance practices	2.00	.826	3	Medium
3	Weakness in managing maintenance team projects	2.00	.733	4	Medium
4	Challenges related to lack of management commitment to support Agile maintenance members	2.02	.811	2	Medium
The general mean		**2.07**	**0.79**	**Medium**	

It can be concluded from values show that the degree is medium, the mean 2.07, and the standard deviation 0.79. Thus, the ratio of challenges faced by the Agile maintenance team regarding Manageability is low. The results indicate a homogeneity of respondents' opinions regarding the administrative challenges facing Agile maintenance professionals.

In the first order (challenges related to the availability of experts for configuring software and hardware resources and agile maintenance experts), the mean is 2.24, standard deviation 0.79, and degree at medium, while in the last order (Weakness of manage maintenance team project), these values are 2.02, 0.811, and medium, respectively.

It is, therefore, concluded that there are some challenging factors in agile maintenance faced by software maintainers as regards Manageability; in detail, challenges related to the availability of experts for configuring software and hardware resources and agile maintenance experts, along with challenges related to lack of management commitment to support agile maintenance members.

4.5.2. Scalability

The calculations continue with the sample related to the Scalability section, and the results are listed in Table 4.22:

Table 4.22 : Calculate mean, standard deviation, rank, and the degree for Scalability in Agile Software Maintenance

I. No	Questions	Mean	Std deviation	Rank	Degree
1	Challenges related to software's ability to interact with other systems in organization	2.19	.862	3	Medium
2	Challenges related to frequent planning of interactions among team members	2.38	.795	1	High
3	Challenges related to organizing large-size projects in agile software maintenance	2.21	.842	2	Medium
4	Challenges related to documentation of large-size projects in agile software maintenance	2.07	.867	4	Medium
The general mean		**2.21**	**0.84**	**Medium**	

As shown in Table 4.22, the results regarding Scalability are: the mean as 2.21 and standard deviation as 0.84. Accordingly, the challenging factors faced by Agile professionals regarding Scalability are low, indicating homogeny in the respondents' views on these issues.

The high grade is concerning the 'Challenges related to frequent planning of interactions among team members', the mean is 2.38 and standard deviation 0.795. Then, the lowest comes for 'Challenges related to documentation of large-size projects in agile software maintenance' with a mean of 2.07 and the same standard deviation and degree at medium.

We can conclude that there are some challenging factors in relation to Scalability that face Agile maintenance professionals**.**

4.5.3. Infrastructure

The questions related to Infrastructure are addressed at this point, and the calculation results are listed in Table 4.23:

Table 4.23: Calculate mean, standard deviation, rank, and the degree for Software Infrastructure

I. No	Questions	Mean	Std deviation	Rank	Degree
1	Challenges related to configuration infrastructure for implementing software maintenance	2.31	.780	2	Medium
2	Challenges related to the availability of infrastructure and resources in Agile software maintenance projects	2.21	.813	3	Medium
3	Challenges related to necessary software tools for artifacts management in agile software maintenance	2.14	.783	4	Medium
4	Challenges related to the necessary testing server for both frequent and automated tests in for Agile software maintenance	2.33	.816	1	Medium
5	Challenges related to the necessary server for frequent delivery of software	2.00	.796	5	Medium
The general mean		**2.20**	**0.80**	**Medium**	

As shown in Table 4.23, the results regarding Scalability are: the mean as 2.20 and standard deviation as 0.80. Accordingly, the challenging factors faced by Agile professionals regarding Infrastructure are low, indicating homogeny in the respondents' views on these issues.

According to the results, 'Challenges related to a necessary testing server for both frequent and automated tests for Agile software maintenance' is at the first place with a mean of 2.33, a standard deviation of 0.816, and a degree at medium; whereas, the last place belongs to 'Challenges related to a necessary server for frequent delivery of software' with a mean of 2.0, a standard deviation of 0.796, and a degree at medium.

It can, then, be concluded that there are some challenging factors facing Agile maintainers regarding the availability of a testing server for both frequent and automated tests.

4.5.4. Communication and Collaboration

We will calculate the values related to questions for the Communication and Collaboration section, and the results appear in Table 4.24:

Table 4.24:Calculate mean, standard deviation, rank, and the degree for Communication and Collaboration

I. No	Questions	Mean	Std deviation	Rank	Degree
1	Challenges concerning geographically distributed teams to reduce the issues related to communication, coordination. Collaboration, etc. among team members.	2.24	.906	2	Medium
2	Challenges related to using advanced tools and techniques for continuous communication between team members.	1.98	.841	3	Medium
3	Challenges related to obtaining customer feedback and involvement in projects.	2.29	.864	1	Medium
The general mean		**2.17**	**0.87**	**Medium**	

Based on the above data, it can be seen that the degree is medium, the mean is 2.17, and standard deviation is 0.87; therefore, the low value indicates that there is homogeny in the views of the respondents' concerning these issues.

In the first place is 'Challenges related to obtaining customers' feedback participation in projects' with a mean of 2.29, a standard deviation of 0.864, and a degree at medium. The last place belongs to 'Challenges related to using advanced tools and techniques for continuous communication among team members' with a mean of 1.98, a standard deviation of 0.841, and a degree at medium.

It can be derived from the above that there are some challenging factors facing Agile maintainers with regard to communication and collaboration, such as those related to obtaining customers' feedback participation in projects.

4.5.5. Transparency

The process is repeated for the questions related to Transparency, and the results appear in Table 4.25:

Table 4.25: Calculate mean, standard deviation, rank, and the degree for Transparency

I. No	Questions	Mean	Std deviation	Rank	Degree
1	Challenges related to providing source code management in agile software maintenance.	2.14	.872	3	Medium
2	Challenges related to sharing data, code, built results, and test reports in agile software maintenance.	2.24	.821	1	Medium
3	Challenges related to traceability mechanism for project artifacts in agile software maintenances.	2.24	.821	2	Medium
4	Challenges related to the client's monitoring the progress of software maintenance in Agile projects.	2.05	.854	4	Medium
The general mean		**2.17**	**0.84**	**Medium**	

This table shows the respondents' opinions about Transparency challenges faced by Agile maintainers to be homogeneous, Where the degree is medium, the mean is 2.17, and

standard deviation is 0.84. This reflects that Agile maintainers have faced problems in the same way regarding Transparency.

In the first rank is the 'Challenges related to sharing data, code, built results, and test reports in agile software maintenance' with a mean of 2.24, standard deviation of 0.821, and a degree at medium. While in the last rank is the 'Challenges related to the clients' monitoring of the progress of software maintenance in agile projects', where the mean ratio is 2.05, the standard deviation is 0.854, and the degree is medium.

From the above, we conclude that there are some challenging factors facing Agile maintainers in terms of Transparency, such as sharing data and code, built results, and test reports.

4.6. Analysis Agile maintenance challenging using Ch-squire test.

4.6.1. Manageability

P1 Challenges related to the availability of experts for configuring software and hardware resources and Agile maintenance experts

Table 4.26 represents the Chi-square test of challenges related to the availability of experts for configuring software and hardware resources and agile maintenance.

Table 4.26: Challenges related to the availability of experts for configuring software and hardware resources and agile maintenance.

		Frequency	Percent	Chi-square	Df	Sig
Valid	None	9	21.4	2.762	3	0.430
	Local	14	33.3			
	Global	12	28.6			
	Local & Global	7	16.7			
	Total	42	100.0			

There are some important points that should be clarified before starting the process of interpreting the results; namely, significant and non-significant Chi-square. According to [94], if there are no statistically significant differences among options as None, Local, Global, and Local & Global, this is referred to as a 'non-significant' Chi-square. In other words, the respondents have rated the options equally as challenges, which implies that all options have the same degree of challenge.

In contrast, if there are statistically significant differences among these options, this is called a 'significant' Chi-square, whereby the respondents give one of the options more importance than the others.

Now, the results are explained in the above table, which indicate that there are no statistically significant differences among the options (None, Local, Global, and Local & Global). The Chi-square is 2.762), and this is considered as non-significant. The results for the challenges related to the availability of experts for configuring software and hardware resources and Agile maintenance for the 'None' option is 21.4%, while the result for the 'Local' option is 33.3% (the largest ratio); 'Global' stands at 28.6%, and 16.7% is the ratio of both options (Local & Global). Figure 4.13 illustrates the results.

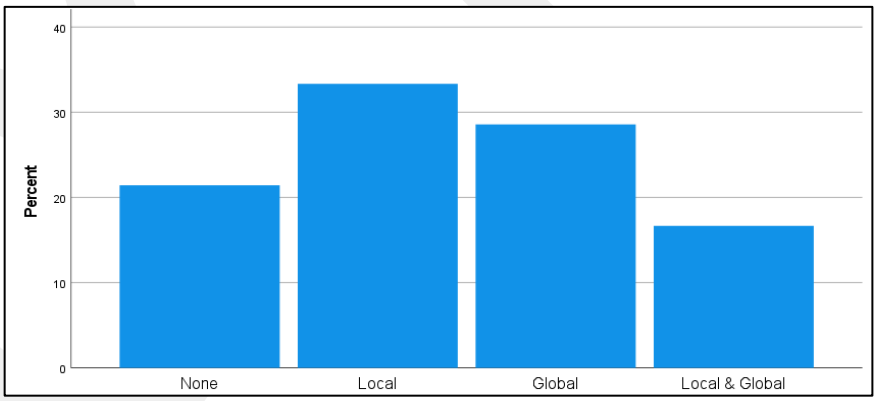


Figure 4.13: Challenges related to the availability of experts for configuring software and hardware resources and Agile maintenance.

P2 Lack of technical support for the improvement of agile maintenance practices.

Table 4.27 represents the Chi-square test of lack of technical support for the improvement of agile maintenance practices.

Table 4.27: Lack of technical support for the improvement of agile maintenance practices.

		Frequency	Percent	Chi-square	Df	Sig
Valid	None	14	33.3	7.714	3	0.052
	Local	14	33.3			
	Global	11	26.2			
	Local & Global	3	7.1			
	Total	42	100.0			

The results in the above table indicate that there are no statistically significant differences among the options (None, Local, Global, and Local & Global). The Chi-square value is 7.714, which is considered non-significant; i.e., all options have the same level of challenge according to respondents [94]. The results related to the lack of technical support for the improvement of agile maintenance practices show the largest ratio (33.3%) for the 'None' and 'Local' options, 26.2% for 'Global', and 7.1% for 'Local & Global'. Figure 4.14 illustrates the findings.

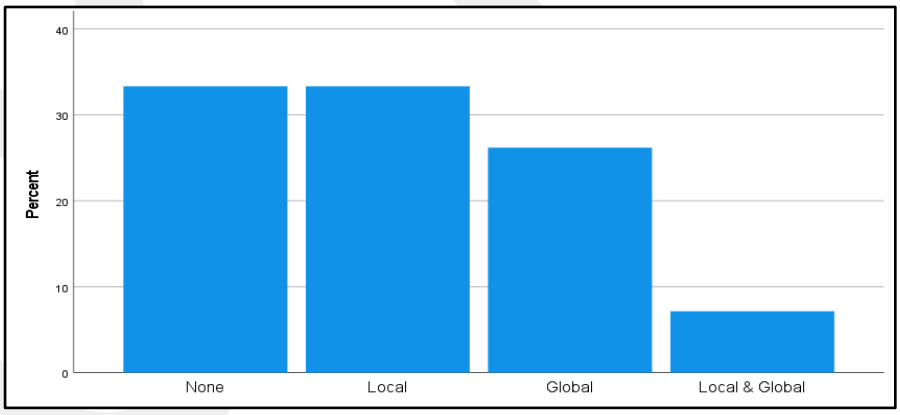


Figure 4.14 : Lack of technical support for the improvement of agile maintenance practices

P3 Weakness in managing maintenance team projects

Table 4.28 represents the Chi-square test of weakness in managing maintenance team projects.

Table 4.28: Weakness in managing maintenance team projects

		Frequency	Percent	Chi-square	Df	Sig
Valid	None	11	26.2	13.429*	3	0.004
	Local	20	47.6			
	Global	5	11.9			
	Local & Global	6	14.3			
	Total	42	100.0			

The results in the table reveal that there are statistically significant differences among the options (None, Local, Global, and Local & Global). The Chi-square value is 13.429, and this is significant and in favor of 'Local'. The results related to the 'Weakness in managing maintenance team projects' are 26.2% for 'None', 47.6% for 'Local' (largest value), 11.9% for 'Global', and finally 14.3% for 'Local & Global'. Figure 4.15 shows the responses for each option.

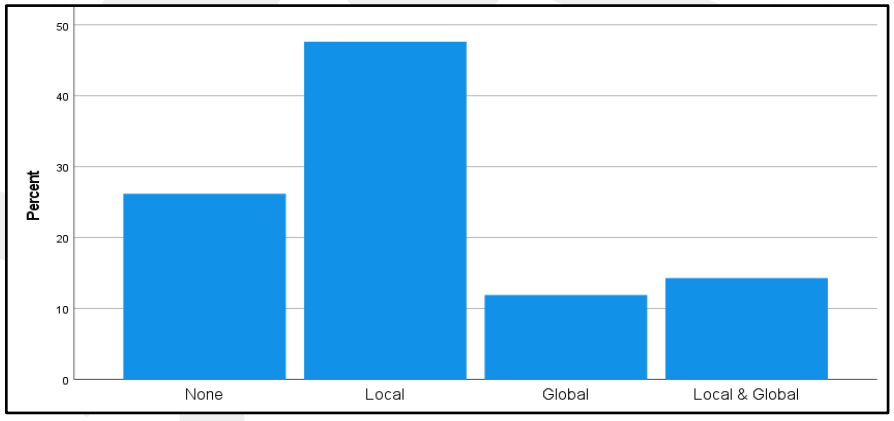


Figure 4.15: Weakness in managing maintenance team projects

P4 Challenges related to lack of management commitment to support agile maintenance members

Table 4.29 represents the Chi-square test of the challenges related to lack of management commitment to support agile maintenance members.

Table 4.29: Challenges related to lack of management commitment to support agile maintenance members

		Frequency	Percent	Chi-square	Df	Sig
Valid	None	13	31.0	4.857	3	0.183
	Local	15	35.7			
	Global	7	16.7			
	Local & Global	7	16.7			
	Total	42	100.0			

The results indicate that there are no statistically significant differences among the options (None, Local, Global, and Local & Global). The Chi-square value is 4.857, considered non-significant; i.e., all options have the same level of challenge according to respondents [94]. The results of the challenges related to the lack of management commitment to support agile maintenance members show that 31% of respondents choose 'None', 35.7% 'Local' (largest value), 16.7% 'Global', and 16.7% 'Local & Global'. Figure 4.16 illustrates the responses related to P4.

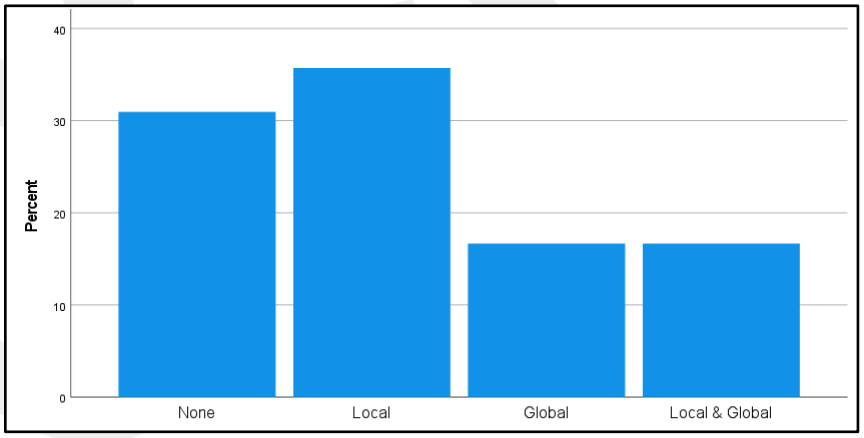


Figure 4.16: Challenges related to lack of management commitment to support agile maintenance members

4.6.2. Scalability in Agile Software Maintenance

P5 Challenges related to software's ability to interact with other systems in organization

Table 4.30 represents the Chi-square test of challenges related to the software's ability to interact with other systems in organization.

Table 4.30 : Challenges related to software's ability to interact with other systems in organization.

		Frequency	Percent	Chi-square	Df	Sig
Valid	None	12	28.6	5.048	3	0.168
	Local	10	23.8			
	Global	15	35.7			
	Local & Global	5	11.9			
	Total	42	100.0			

Accordingly, there are no statistically significant differences among the options (None, Local, Global, and Local & Global). The Chi-square value is 5.048, regarded as non- significant – that is, all options have the same level of challenge according to the responses [94]. The results of the challenges related to the software's ability to interact with other systems in organization are 28.6% for 'None' (highest), 23.8% 'Local', 35.7% 'Global', and 11.9% for 'Local & Global'. In Figure 4.17, the differences are illustrated for the responses.

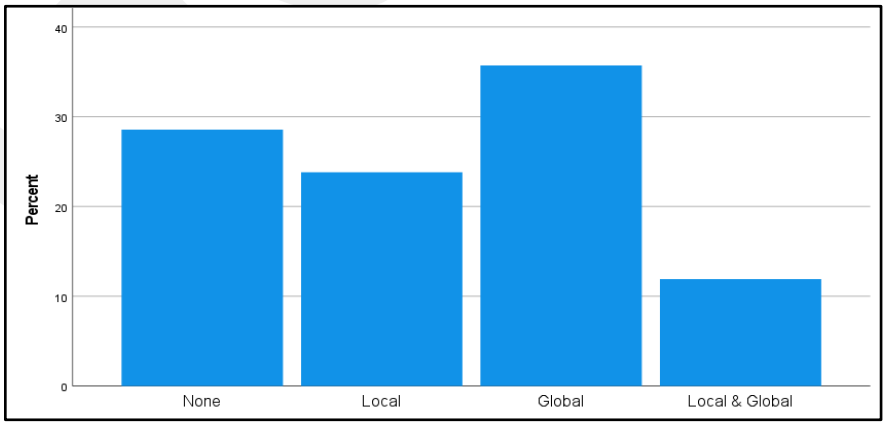


Figure 4.17: Challenges related to software's ability to interact with other systems in organization

P6 Challenges related to frequent planning of interactions among team members

Table 4.31 represents the Chi-square test of challenges related to the frequent planning of interactions among team members.

Table 4.31: Challenges related to frequent planning of interactions among team members

		Frequency	Percent	Chi-square	df	Sig
Valid	None	8	19.0	7.905*	3	0.048
	Local	10	23.8			
	Global	18	42.9			
	Local & Global	6	14.3			
	Total	42	100.0			

The results in the above table show that there are statistically significant differences among the options (None, Local, Global, and Local & Global). The Chi-square value is 7.905, which is significant for the 'Global' option. The results of the challenges related to frequent planning of interactions among team members are 19% for 'None', 23.8% for 'Local', 42.9% for 'Global' as the largest value, and finally 14.3% for 'Local & Global'. Figure 4.18 shows the respondents' answers.

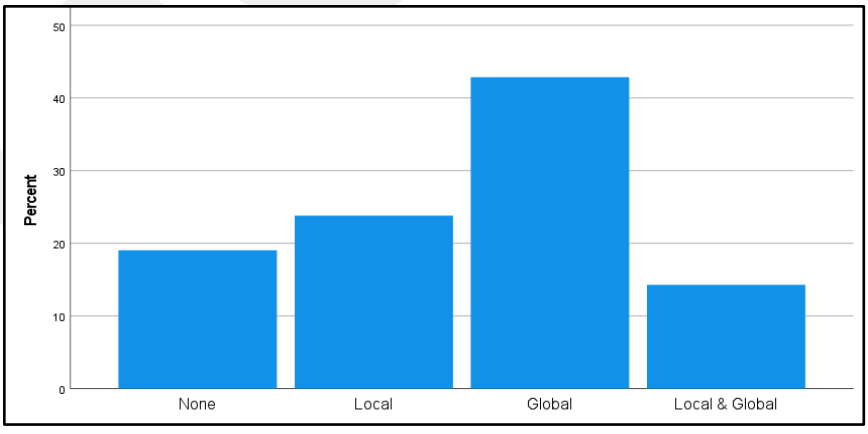


Figure 4.18: Challenges related to frequent planning of interactions among team members

P7 Challenges related to organizing large-size projects in agile software maintenance

Table 4.32 represents the Chi-square test of challenges related to organizing large-size projects in agile software maintenance.

Table 4.32: Challenges related to organizing large-size projects in agile software maintenance

		Frequency	Percent	Chi-square	Df	Sig
Valid	None	10	23.8	5.048	3	0.168
	Local	12	28.6			
	Global	15	35.7			
	Local & Global	5	11.9			
	Total	42	100.0			

As can be seen, there are no statistically significant differences among the options (None, Local, Global, and Local & Global). The Chi-square value is 5.048, non-significant; i.e., all options have the same level of challenge according to respondents [94]. The results of the challenges related to organizing large-size projects in agile software maintenance are 23.8% 'None', 28.6% 'Local', '35.7% 'Global' (largest), and 11.9% 'Local & Global'. Figure 4.19 shows the distribution of the responses.

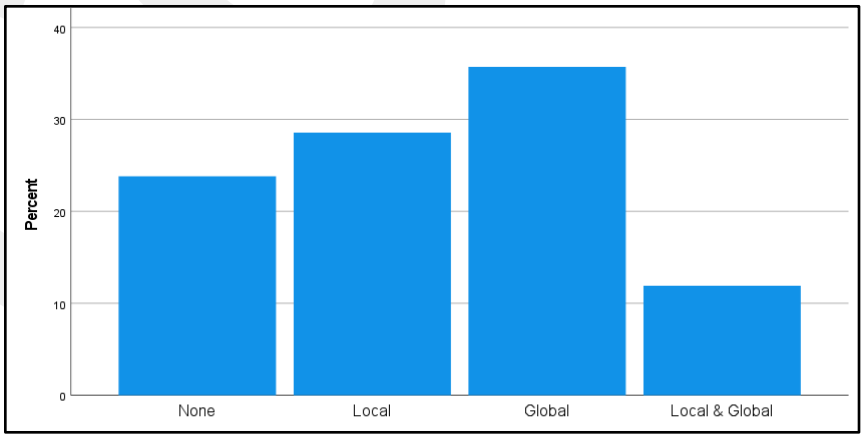


Figure 4.19: Challenges related to organizing large-size projects in agile software maintenance

P8 Challenges related to documentation of large-size projects in Agile software maintenance

Table 4.33 represents the Chi-square test of the challenges related to the documentation of large-size projects in agile software maintenance.

Table 4.33: Challenges related to documentation of large-size projects in agile software maintenance

		Frequency	Percent	Chi-square	Df	Sig
Valid	None	14	33.3	3.143	3	0.370
	Local	11	26.2			
	Global	11	26.2			
	Local & Global	6	14.3			
	Total	42	100.0			

The results in the table indicate that there are no statistically significant differences among the options (None, Local, Global, and Local & Global). The Chi-square value is 3.143, which is non-significant and means that all options have the same level of challenge according to the answers [94]. The results of the challenges related to the documentation of large-size projects in agile software maintenance are 33.3% as 'None' and as the largest value, 26.2% 'Local', 26.2% 'Global', and 14.3% that goes to 'Local & Global'. Figure 4.20 shows the distribution of answers.

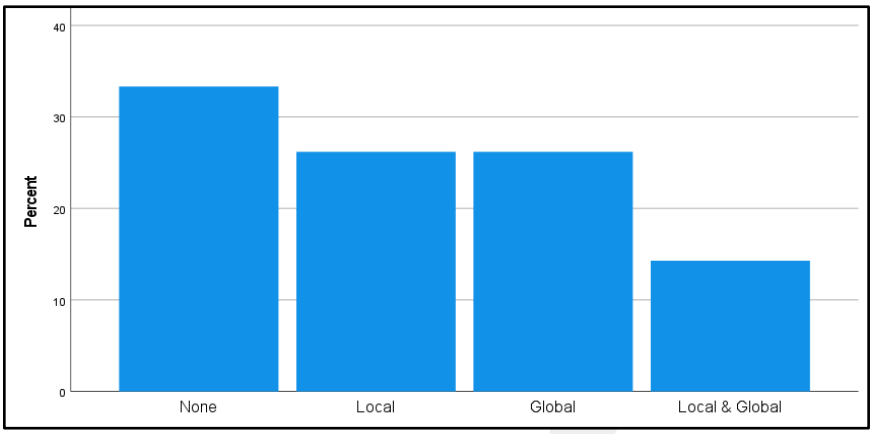


Figure 4.20: Challenges related to documentation of large-size projects in Agile software maintenance

4.6.3. Infrastructure

P9 Challenges related to configuration infrastructure for implementing software maintenance

Table 4.34 represents the Chi-square test of challenges related to configuration infrastructure for implementing software maintenance.

Table 4.34: Challenges related to configuration infrastructure for implementing software maintenance

		Frequency	Percent	Chi-square	Df	Sig
Valid	None	8	19.0	1.238	3	0.744
	Local	13	31.0			
	Global	11	26.2			
	Local & Global	10	23.8			
	Total	42	100.0			

The results indicate that there are no statistically significant differences among the options (None, Local, Global, and Local & Global). The Chi-square value is 1.238, thus non-significant (i.e., all options have the same level of challenge according to the respondents) [94]. The results of the challenges related to configuration infrastructure for implementing

software maintenance show that 19% of answers go to 'None', 31% to 'Local' (largest), 26.2% to 'Global', and 23.8% to 'Local & Global'. Figure 4.21 shows the corresponding distribution.

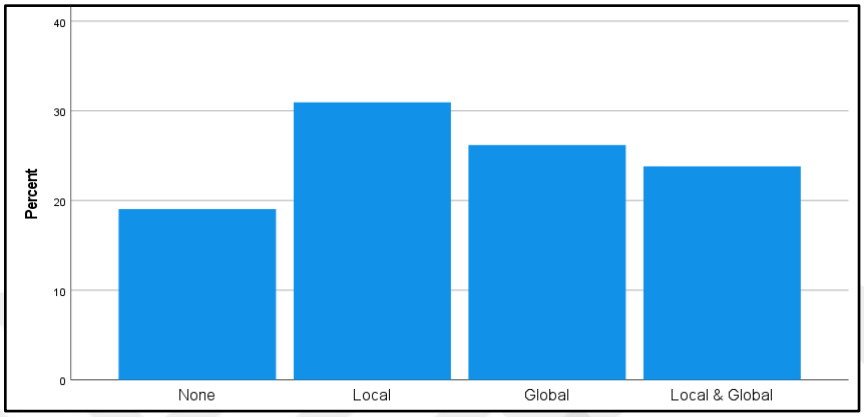


Figure 4.21: Challenges related to configuration infrastructure for implementing software maintenance

P10 Challenges related to the availability of infrastructure and resources in Agile software maintenance projects

Table 4.35 represents the Chi-square test of challenges related to the availability of infrastructure and resources in Agile software maintenance projects.

Table 4.35 : Challenges related to the availability of infrastructure and resources in Agile software maintenance projects

		Frequency	Percent	Chi-square	Df	Sig
Valid	None	10	23.8	0.857	3	0.836
	Local	13	31.0			
	Global	10	23.8			
	Local & Global	9	21.4			
	Total	42	100.0			

The results in the table show that there are no statistically significant differences among the options (None, Local, Global, and Local & Global). The Chi-square value is 0.857, which is

non-significant; i.e., all options have the same level of challenge according to respondents) [94]. Accordingly, 23.8% of the answers are for 'None', 31% 'Local', 23.8% 'Global' and the largest, and 21.4% 'Local & Global' Figure 4.22 shows the distribution.

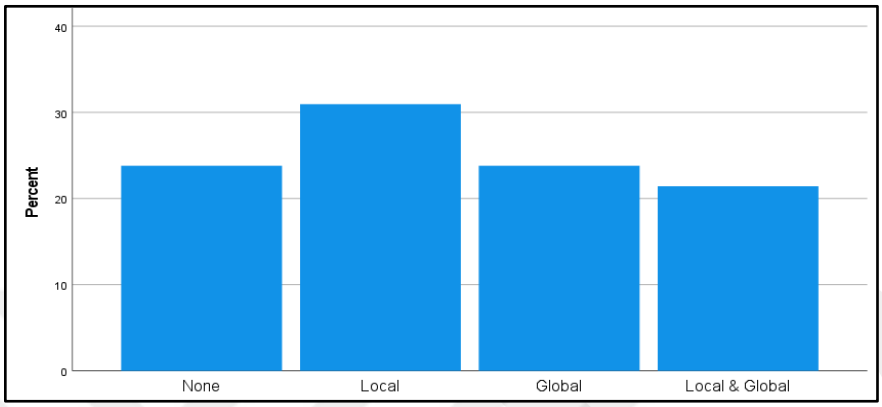


Figure 4.22: Challenges related to the availability of infrastructure and resources in Agile software maintenance projects

P11 Challenges related to necessary software tools for artifacts management in agile software maintenance

Table 4.36 represents the Chi-square test of challenges related to necessary software tools for artifacts management in agile software maintenance.

Table 4.36 : Challenges related to necessary software tools for artifacts management in agile software maintenance

		Frequency	Percent	Chi-square	Df	Sig
Valid	None	10	23.8	4.095	3	0.251
	Local	16	38.1			
	Global	8	19.0			
	Local & Global	8	19.0			
	Total	42	100.0			

The results indicate that there are no statistically significant differences among the options (None, Local, Global, and Local & Global). The Chi-square value is 4.095 and non-

significant, which means all options have the same level of challenge according to the answers [94]. The results of the challenges related to necessary software tools for artifacts management in agile software maintenance are 23.8% as 'None', 38.1% as 'Local' (largest), 19% as 'Global', and 19% as 'Local & Global'. Figure 4.23 shows the results.

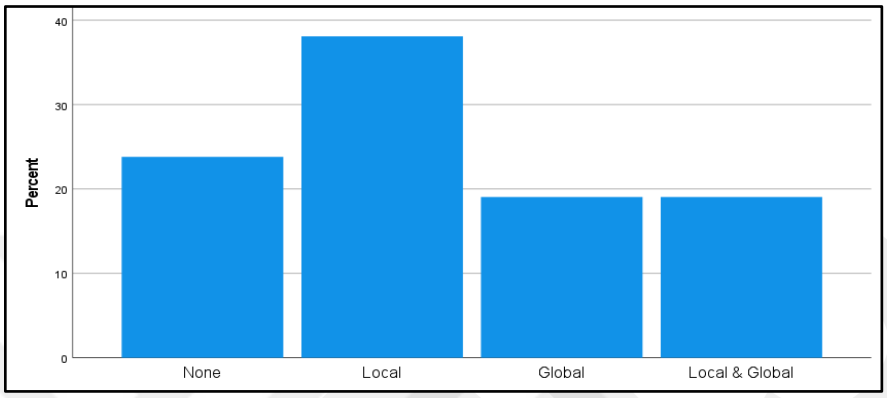


Figure 4.23 : Challenges related to necessary software tools for artifacts management in agile software maintenance

P12 Challenges related to the necessary testing server for both frequent and automated tests in for Agile software maintenance

Table 4.37 represents the Chi-square test of challenges related to the necessary testing server for both frequent and automated tests in Agile software maintenance.

Table 4.37 : Challenges related to the necessary testing server for both frequent and automated tests in Agile software maintenance

		Frequency	Percent	Chi-square	Df	Sig
Valid	None	9	21.4	0.857	3	0.836
	Local	10	23.8			
	Global	13	31.0			
	Local & Global	10	23.8			
	Total	42	100.0			

The results in the table indicate that there are no statistically significant differences among the options (None, Local, Global, and Local & Global). The Chi-square value is 0.857 and

considered non-significant, meaning all options have the same level of challenge according to the respondents [94]. The results of the challenges related to the necessary testing server for both frequent and automated tests in Agile software maintenance are 21.4% 'None', 23.8% 'Local', 31% 'Global' (highest, and 23.8% 'Local & Global'. Figure 4.24 shows the distribution.

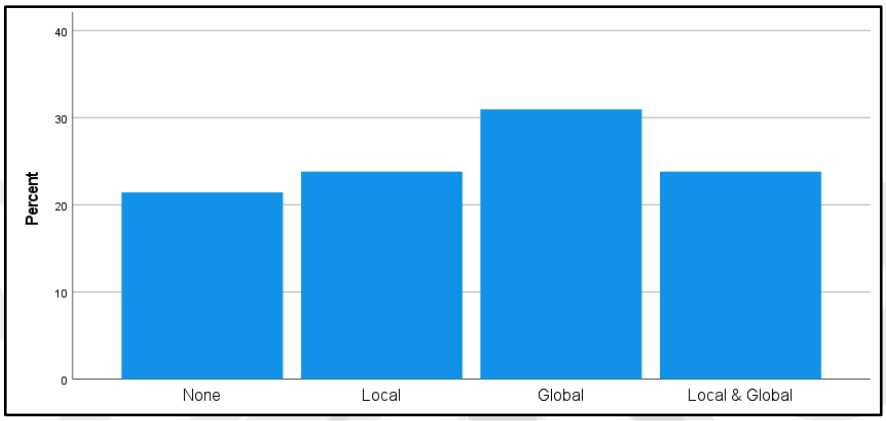


Figure 4.24 : Challenges related to the necessary testing server for both frequent and automated tests in Agile software maintenance

P13 Challenges related to the necessary server for frequent delivery of software

Table 4.38 represents the Chi-square test of challenges related to the necessary server for frequent delivery of software.

Table 4.38 : Challenges related to the necessary server for frequent delivery of software.

		Frequency	Percent	Chi-square	df	Sig
Valid	None	12	28.6	7.714	3	0.052
	Local	17	40.5			
	Global	5	11.9			
	Local & Global	8	19.0			
	Total	42	100.0			

The results in the table indicate that there are no statistically significant differences among the options (None, Local, Global, and Local & Global). The Chi-square value is 7.714 and

considered non-significant (i.e., all options have the same level of challenge based on the responses) [94]. The results of the challenges related to the necessary server for frequent delivery of software are 28.6% 'None', 40.5% 'Local' (largest value), 11.9% 'Global, and 19% 'Local & Global'. Figure 4.25 shows the distribution of answers.

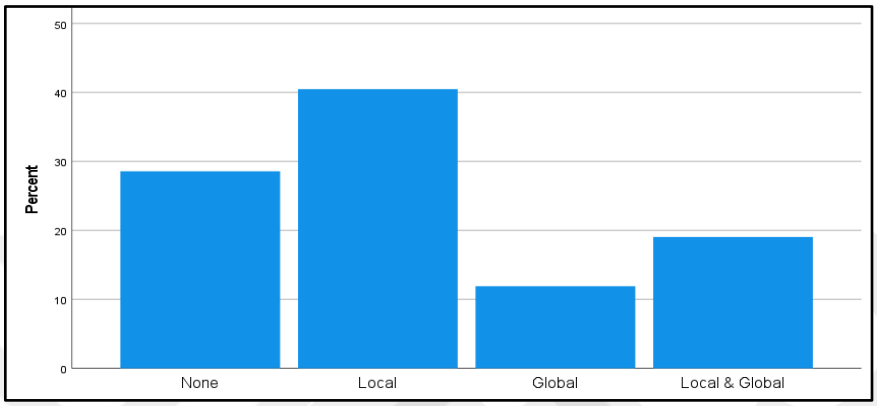


Figure 4.25: Challenges related to the necessary server for frequent delivery of software

4.6.4. Communication and Collaboration

P14 Challenges concerning geographically distributed teams to reduce the issues related to communication, coordination. Collaboration, etc. among team members.

Table 4.39 represents the Chi-square test of challenges concerning geographically distributed teams to reduce issues related with communication, coordination. Collaboration, etc. among team members.

Table 4.39: Challenges concerning geographically distributed teams to reduce issues related with reducing communication, coordination. collaboration etc. among team members.

		Frequency	Percent	Chi-square	Df	Sig
Valid	None	13	31.0	8.476*	3	0.037
	Local	6	14.3			
	Global	17	40.5			
	Local & Global	6	14.3			
	Total	42	100.0			

The results show that there are statistically significant differences among the options (None, Local, Global, and Local & Global). The Chi-square value is 8.476, which considered as significant in favor of the option 'Global'. The percentages are 31% for 'None', 14.3% for 'Local', 40.5% for 'Global' (highest), and 14.3% for 'Local & Global'. Figure 4.26 shows the distribution of answers.

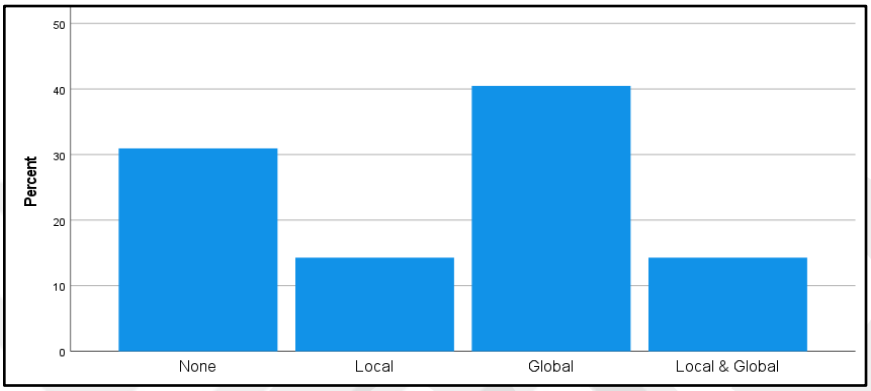


Figure 4.26 : Challenges concerning geographically distributed teams to reduce issues related with communication, coordination. Collaboration, etc. among team members

P15 Challenges related to using advanced tools and techniques for continuous communication between team members

Table 4.40 represents the Chi-square test of challenges related to using advanced tools and techniques for continuous communication between team members.

Table 4.40 : Challenges related to using advanced tools and techniques for continuous communication between team members

		Frequency	Percent	Chi-square	Df	Sig
Valid	None	15	35.7	5.619	3	0.132
	Local	13	31.0			
	Global	5	11.9			
	Local & Global	9	21.4			
	Total	42	100.0			

Accordingly, there are no statistically significant differences among the options (None, Local, Global, and Local & Global). The Chi-square value is (5.619) and considered non-significant (i.e., all options have the same level of challenge according to the respondents) [94]. The results of the challenges related to using advanced tools and techniques for continuous communication between team members are 35.7% 'None' (largest value), 31% 'Local', '11.9%' Global, and 21.4% 'Local & Global'. Figure 4.27 shows the related distribution.

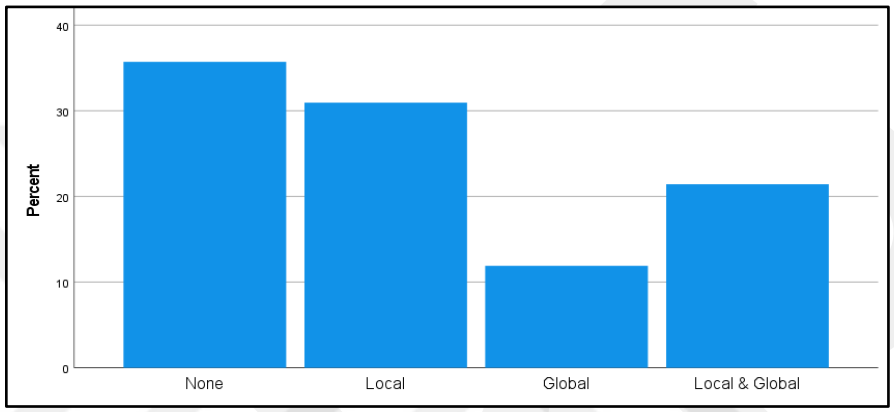


Figure 4.27 : Challenges related to using advanced tools and techniques for continuous communication between team members

P16 Challenges related to obtaining customer feedback and involvement in projects

Table 4.41 represents the Chi-square test of challenges related to obtaining customer feedback and involvement in projects.

Table 4.41 : Challenges related to obtaining customer feedback and involvement in projects

		Frequency	Percent	Chi-square	Df	Sig
Valid	None	11	26.2	3.143	3	0.370
	Local	8	19.0			
	Global	15	35.7			
	Local & Global	8	19.0			
	Total	42	100.0			

There are no statistically significant differences among the options (None, Local, Global, and Local & Global). The Chi-square value is 3.143 and non-significant (i.e., all options have the

same level of challenge according to the responses) [94]. The percentages are 26.2% for 'None', 19% for 'Local', 35.7% for 'Global' highest), and 19% for 'Local & Global'. Figure 4.28 shows the result distribution.

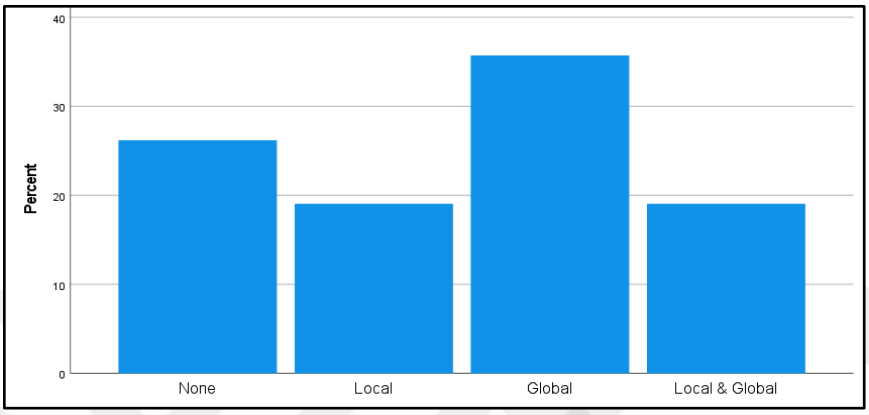


Figure 4.28: Challenges related to obtaining customer feedback and involvement in projects

4.6.5. Transparency

P17 Challenges related to providing source code management in agile software maintenance

Table 4.42 represents the Chi-square test of challenges related to providing source code management in agile software maintenance.

Table 4.42 : Challenges related to providing for source code management in agile software maintenance

		Frequency	Percent	Chi-square	Df	Sig
Valid	None	13	31.0	2	3	0.572
	Local	10	23.8			
	Global	12	28.6			
	Local & Global	7	16.7			
	Total	42	100.0			

The results in the table indicate that there are no statistically significant differences among the options (None, Local, Global, and Local & Global). The Chi-square value is 2 and non-significant (i.e., all options have the same level of challenge according to results) [94]. The percentages are 31% for 'None' as the highest, 23.8% for 'Local', 28.6% for 'Global', and 16.7% for 'Local & Global'. Figure 4.29 shows the distribution.

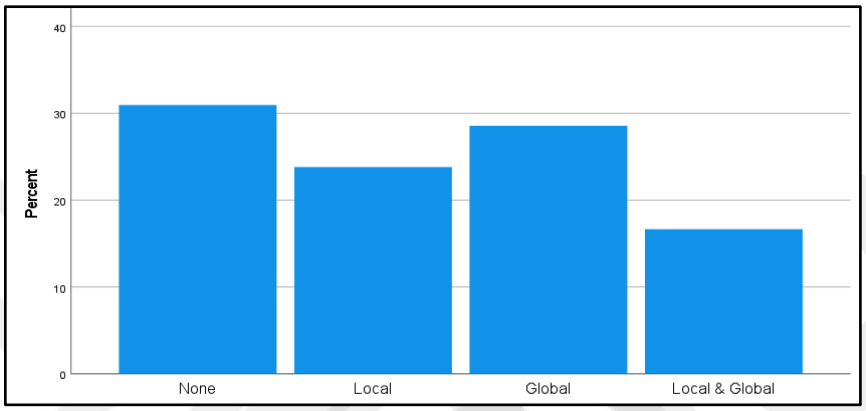


Figure 4.29 : Challenges related to providing source code management in agile software maintenance

P18 Challenges related to sharing data, code, built results and test reports in agile software maintenance

Table 4.43 represents the Chi-square test of challenges related to sharing data, code, built results and test reports in agile software maintenance.

Table 4.43: Challenges related to sharing data, code, built results and test reports in agile software maintenance

		Frequency	Percent	Chi-square	Df	Sig
Valid	None	10	23.8	2	3	0.572
	Local	12	28.6			
	Global	13	31.0			
	Local & Global	7	16.7			
	Total	42	100.0			

Accordingly, there are no statistically significant differences among the options (None, Local, Global, and Local & Global). The Chi-square value is 2, which is non-significant, meaning all options have the same level of challenge according to the responses [94]. The percentages are 23.8% 'None', 28.6% 'Local', 31% 'Global' as the largest value, and 16.7% 'Local & Global'. Figure 4.30 illustrates the distribution.

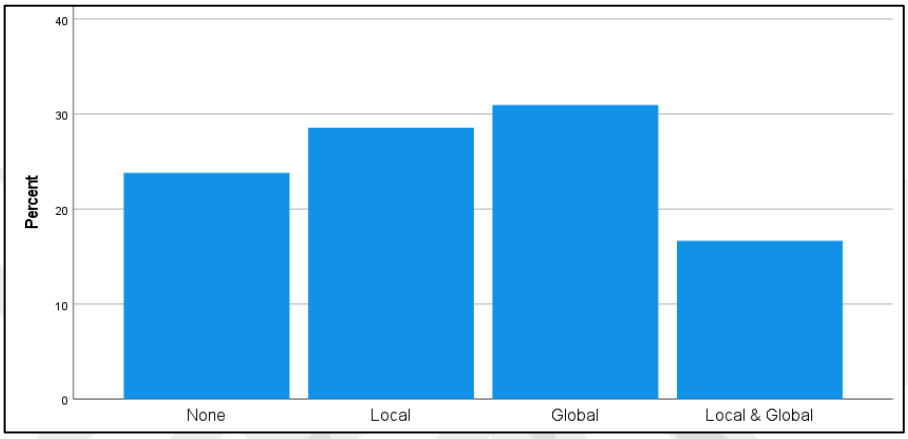


Figure 4.30: Challenges related to sharing data, code, built results and test reports in agile software maintenance

P19 Challenges related to traceability mechanism for project artifacts in agile software maintenances

Table 4.44 represents the Chi-square test of Challenges related to the traceability mechanism for project artifacts in agile software maintenance.

Table 4.44 **:** Challenges related to traceability mechanism for project artifacts in agile software maintenance

		Frequency	Percent	Chi-square	Df	Sig
Valid	None	10	23.8	0.286	3	0.963
	Local	12	28.6			
	Global	10	23.8			
	Local & Global	10	23.8			
	Total	42	100.0			

There are no statistically significant differences among the options (None, Local, Global, and Local & Global). The Chi-square value is 0.286 and non-significant (i.e., all options have the same level of challenge according to the responses) [94]. The percentages include 23.8% as 'None', 28.6% as 'Local' (largest value), 23.8% as 'Global', and 23.8% as 'Local & Global'. Figure 4.31 shows the distribution.

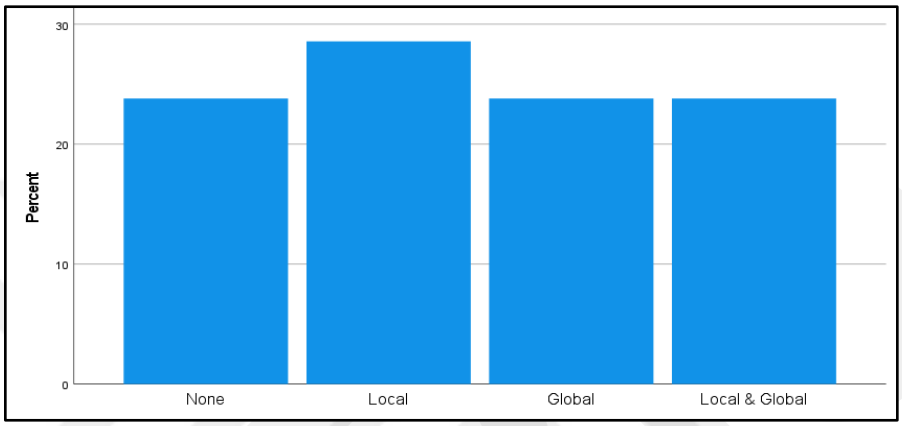


Figure 4.31: Challenges related to traceability mechanism for project artifacts in agile software maintenance

P20 Challenges related to the client's monitoring the progress of software maintenance in Agile projects

Table 4.45 represents the Chi-square test of Challenges related to viewing the client's progress of software maintenance in agile projects.

Table 4.45 : Challenges related to viewing the client's progress of software maintenance in Agile projects

		Frequency	Percent	Chi-square	Df	Sig
Valid	None	14	33.3	4.286	3	0.232
	Local	12	28.6			
	Global	11	26.2			
	Local & Global	5	11.9			
	Total	42	100.0			

As can be seen, there are no statistically significant differences among the options (None, Local, Global, and Local & Global). The Chi-square value is 4.286 and non-significant (i.e., all options have the same level of challenge according to the results) [94]. The breakdown is 33.3% as 'None', 28.6% as 'Local' (largest value), 26.2% as 'Global', and 11.9% as 'Local & Global'. Figure 4.32 shows the results.

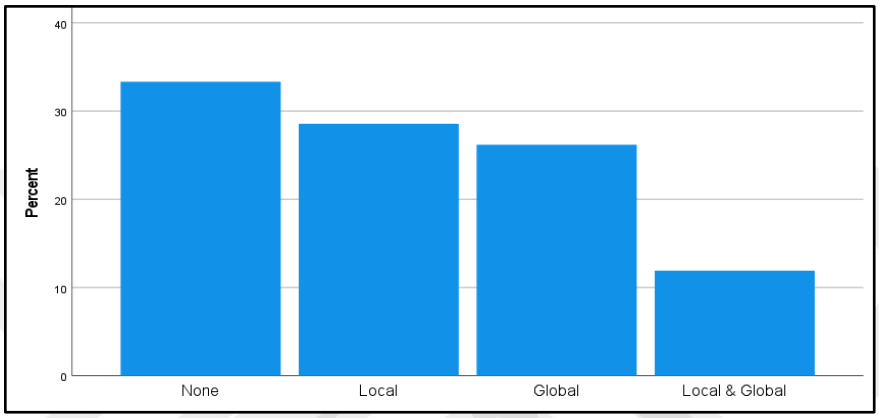


Figure 4.32: Challenges related to viewing the client's progress of software maintenance in agile projects

4.7. General questions

1) What are your views on adopting agile maintenance in the Cloud?

The responses of the participants for this question varied, and the answers were referred to many silos at different hierarchical levels of management that obstruct smooth information flow. This caused many problems in terms of transparency. For instance, many times the budget of the entire software project was rendered as unknown at all levels.

2) Do you think Agile maintenance in the Cloud has a good impact or a bad impact on your work?

The responses of the participants to this question varied as well, and the answers were as follows:

- Adoption of Agile has reduced time effort which in return effected the reduction of cost factor

- Better way to have a full utilization of a tool and its capabilities to get maximum ROI for the organization.
- fast, economical, performance, efficient, result-oriented business ability
- Agile is the way to go ahead but people need to have an Agile mindset.
- There is nothing like "Hybrid-Agile".
- You need high expertise and competencies to apply the methodology correctly, as it is not easy to do all the steps of the methodology except through an experienced person.

3) In your opinion, what is the good impact or adverse impact of using Agile maintenance in the Cloud on your work?

The responses of the participants were as follows:

- Adverse: Additional burden of locally maintained but a lot can be learnt and built.
- Better ROI and cost optimization.
- Continuous feedback from client helps improve the solution and also react quickly to customer needs.
- Easy to access. don't have to take care of machine maintenance.
- It definitely has good impact of Agile maintenance.
- It involves the customer in this process, and adverse impact is that you required expert team for using Agile maintenance.
- Is that working on the Cloud enables us to communicate early and respond early.

CHAPTER 5

EXTENDING THE ADCC FRAMEWORK AND CASE STUDY

5.1. Introduction

There are many studies that have adapted Agile in Cloud Computing, such as [32], [15], [16], [33], [12], and [13] ; however, only one has proposed Agile Development Cloud Computing (ADCC) framework [12], as shown in Figure 5.1, which has satisfactory characteristics and achieves good results upon evaluation as indicated in [13], and [14]; though, in case of Agile maintenance, some shortcomings have also been reported. In this chapter, we will extend the ADCC framework by including maintenance as a part in it, and evaluate the framework using a case study in actual settings.

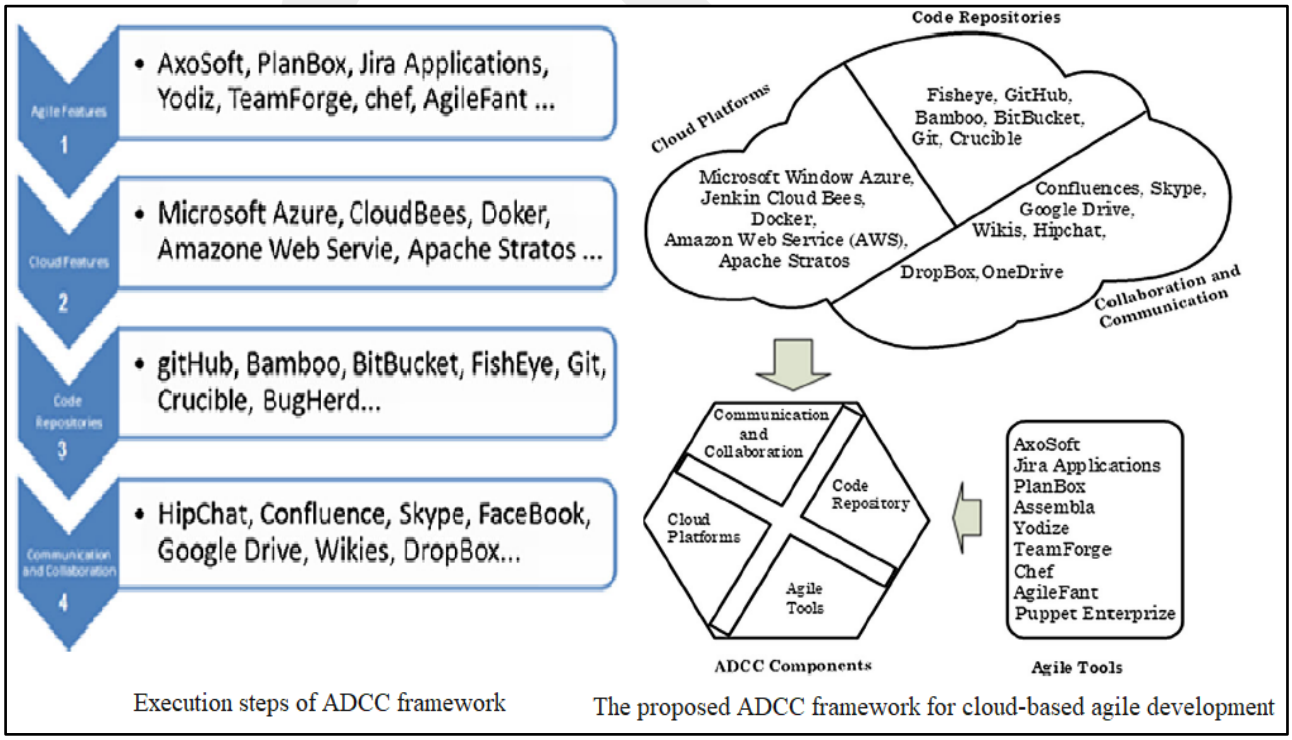


Figure 5.1: ADCC Framework [14].

5.2. Agile Software Maintenance and Development Cloud Computing Framework

Agile Software Maintenance and Development Cloud Computing (ASMDCC) framework, which is based on the ADCC framework and is different from it by providing a suitable environment for managing projects using Agile methodologies for the development and maintenance process. It consists of four main steps to determine the following:

1. The Cloud platform based on the organization size and business needs;
2. The required process for development and/or maintenance;
3. The Agile methodology that will be followed; XP, Scrum, Kanban, etc., and the respective Agile practices; and
4. The Agile tools for code repository, management, documentation, collaborations, and testing. See Figure 5.2, which illustrates the extensions in different colors.

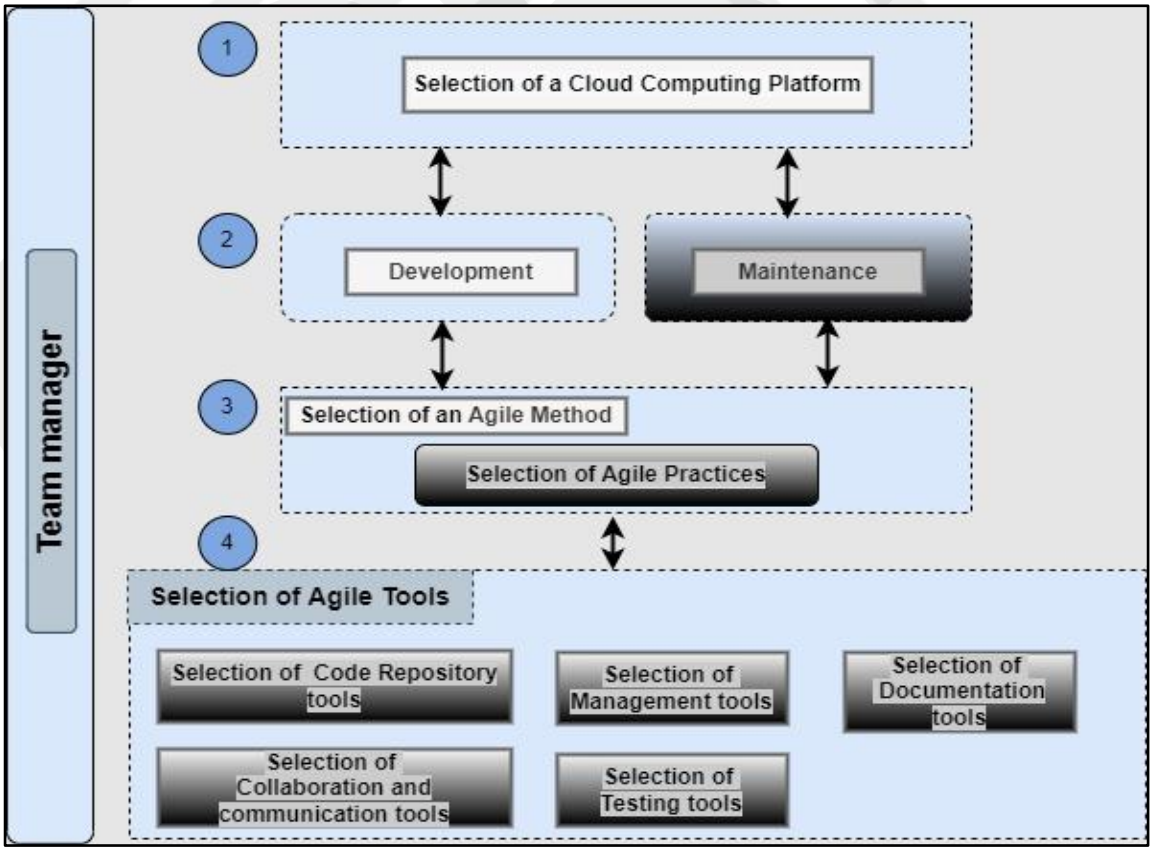


Figure 5.2: ASMDCC Framework

5.2.1. Cloud Computing Service Providers

In the first step, the Agile team selects the desired Cloud Computing platform according to their needs. There are several platforms available, and the major ones are highlighted here.

5.2.1.1. Amazon Web Services (AWS)

This is a popular Cloud service provider launched in 2006, offering different Cloud services such as: Amazon EC2, Amazon S3, Amazon RDS (Amazon Relational Database Services), Amazon Simple DB, Amazon Route 53, Amazon CloudFront, and Amazon Elastic MapReduce [96], [97].

5.2.1.2. Google Cloud Platform (GCP)

This Cloud Computing platform was launched in 2008 by Google, and it provides different Cloud services such as: Compute Engine, App Engine, Cloud Storage, Cloud SQL, Cloud Datastore, and Big-Query [96], [97].

5.2.1.3. Microsoft Azure

In 2010, the Microsoft Company launched Microsoft Azure, offering several Cloud services for customers such as: Infrastructure services, Web development, Mobile development platform, Media services, Storage, Big data Cloud, and Identity and access management services [96], [97].

5.2.1.4. Oracle Cloud

The Oracle Cloud services launched were in 2012, and they spread over 40 regions, providing many services, such as (IaaS), (PaaS), (SaaS), and (DaaS). Customers can use these services to develop, deploy, and integrate applications in the Cloud [96], [98].

5.2.1.5. IBM Cloud

The IBM Cloud services were launched in 2011 to offer many services, such as Compute Infrastructure, Compute Services, Storage, Network, and Mobile [96].

More details are provided here by comparing certain characteristics between these Cloud platforms. See Table 5. 1.

Table 5. 1: Comparison among Cloud Platforms [96], [99], [100].

Cloud Platforms / Services	Amazon Web Services	Google Cloud Platform	Microsoft Azure	Oracle Cloud	IBM Cloud
Services	IaaS, PaaS, Storge, Database.	IaaS, PaaS, Storage,mobile, database,Big Data	Iaas, Paas, mobile,Media, Database, Big Data	IaaS, PaaS, SaaS, and DaaS.	IaaS, PaaS, SaaS, Storage, Network, and Mobile
Cost Model	Pay-As-You-Go model.	Pay-As-You-Go model.	Pay-As-You-Go model.	Pay-As-You-Go model.	Pay-As-You-Go model.
Locations	25 regions	25 regions	54 regions	29 regions.	Six multi-zone regions.
Virtual Server	Amazon EC2	Compute Engine	Azure Virtual Machine	Oracle Cloud Infrastructure Compute	-Classic Virtual Server -Virtual Server for VPC -Power Systems Virtual Servers -Hyper Protect Virtual Server (LinuxONE) -Quantum Services
Maximum Processors in VM	128	160	M128ms	Cores: 80 (160 vCPUs)	56
Storage	-Simple Storage Service (S3) -Elastic Block Storage (EBS) - Elastic File System (EFS) - FSx for Windows File Server - FSx for Lustre - FSx for NetApp ONTAP - FSx for OpenZFS\ - S3 Glacier -Storage Gateway	- Cloud Storage -Persistent Disk -File Store	-Azure Blob Storage -Azure Page Blobs / Premium Storage -Managed Disks -Azure Files -Azure NetApp Files -Azure Archive Storage -Azure Cool Storage -Azure Storsimple	- Oracle Cloud Infrastructure Object Storage - Oracle File Storage Services - Oracle Cloud Object Storage Infrequent Access Tier - Oracle Cloud Infrastructure Storage Gateway	-Cloud Object Storage - Block Storage -Block Storage for VPC -File Storage -Object Storage-ColdVault -Archive
Database	-Amazon Aurora -Amazon RDS -Amazon DynamoDB -Amazon Timestream -Amazon Neptune -Amazon ElastiCache -Amazon MemoryDB for Redis -Amazon Redshift	-Cloud Datastore -Cloud Firestore -Cloud Bigtable -Cloud MemoryStore -BigQuerya	-Azure CosmosDB -Table Storage -Azure Time Series Insights -Azure Time Series Insights -Azure Cache for Redis -Azure Synapse Analytics	-Oracle NoSQL Database Cloud Service -Oracle Autonomous JSON Database -Graph Database -Exadata Cloud Service -Autonomous Data Warehouse Cloud	-Cloudant NoSQL DB -Compose for JanusGraph -Databases for MongoDB -Databases for etcd -Databases for DataStax -Hyper Protect DBaaS for MongoDB -Informix -IBM Graph -Databases for Redis

5.2.2. Agile Methods and Practices

In the second step, the team selects a suitable Agile method. Here we will compare the most popular methods in Table 5.2.

Table 5. 2: Comparison among Different Agile Methods [101], [102], [103], [104], [105].

Agile Methods / Parameters	**XP**	**Scrum**	**Kanban**	**Crystal**	**DSDM Dynamic Systems Development Method**
Process Type	Development	Development /Management	Development	Developme-nt	Development
Team Size	2-10	5-9	5 – 9 members with Scrum Master and Product Owner	Any size	2-10
Team Management	Self-organized	Loosely organized	Self-organized	Self-organized	Self-organized
Iteration/ Sprint duration	1-3 weeks	1-4 weeks	No specific period	1-2 weeks	2-4 weeks
Daily meeting	Yes	Yes	Yes	No	No
Face to Face meeting	No	No	No	Yes	No
Requirement Gathering	User stories	Product backlog	User stories	Use cases	User stories
User Interaction	Very high	High	-	Medium	Medium
Project Coordinator	XP Coach	Scrum Master	Team Work	Considered as sub-roles in Crystal	Technical coordinator
Methods Process	1.Exploration 2. Planning 3. Iterations to first release 4. Production 5.Maintenance	1. Pre-game 2.Development 3. Release	No defined Process	No defined Process	1. Feasibility 2.Business study 3.Functional Model Iteration 4.Design and build iteration 5.Implementation

Table 5.2 (cont'd). Comparison among Different Agile Methods [101], [102], [103], [104], [105].

Agile Methods / Parameters	XP	Scrum	Kanban	Crystal	DSDM Dynamic Systems Development Method
Roles and Responsibilities	1. Coach 2.Programmer 3. Tester 4.Customer 5. Consultant	1.Product Owner 2.Scrum Team 3.ScrumMaster 4. Chickens	1.Service Delivery Manager 2.Service Request Manager	1. Sponsor 2.Designer-Programmers 3. Coordinator 4.Business Expert 5.Ambassador User 6.Lead designer 7. Tester 8. Writer	1.Project manager 2.Chief architect 3.Domain experts 4.Development Manager 5.Chief Programmers 6. Class owners 7. Environment Manager 8. Tester
Practices	1.Planning game 2.Small/short releases 3.System Metaphor 4.Simple design 5.Testing 6.Frequent Refactoring 7.Pair programming 8.collective ownership 9.Continuous integration 10. 40- hour week (sustain-able pace) 11.On-site customer 12.Coding standards 13.Open workspace 14. Just rules	1.Self-directed and self-organizing team 2.No external addition of work to an iteration, once chosen 3.Daily standup meeting with special questions 4.Usually 30-calendar day iterations 5.Demo to external stakeholders at end of each iteration 6.Each iteration, client-driven adaptive planning	1.Visualize 2.Limit work in progress 3.Manage flow 4.Make policies explicit 5.Implement feedback loops 6.Improve collaboratively, evolve experimentally	1.Frequent delivery 2.Close communication 3.Reflective improveme-nt 4.Personal safety 5. Focus 6.Easy access to experts' users 7.Technical environmen-t	1.Domain Object Modeling 2.Developing by Feature 3.Individual Class (Code) Ownership 4.Feature Teams 5. Inspections 6.Regular Builds 7.Configuration Management 8.Reporting/ Visibility of Results

5.2.3. Agile Tools

In this step, the team determines which Agile tools are needed for the project to simplify the development and maintenance process. There are many tools are available in the market, and we shall review the most popular ones used by Agile professionals to date.

1. **Management tools**

Project management tools help teams to plan, track, and support the project. There are many tools currently available to manage the development and maintenance process in the Agile methods. According to [106], the five tools which are commonly used in Agile management are:

A. Assembla: One of Agile management tools which provides a set of project management services compatible with Git (for tracking changes), Apache Subversion SVN (for revision control), and Perforce Helix Core for projects in the Cloud platform. Assembla accommodates between 5 and 5000 users [106].

B. Taiga: An open-source Agile management tool, it is simple to use and has some features to support Scrum projects. This tool can be combined with a group of other tools such as GitHub, Gitlab, Bitbucket, Gogs, Chats, and Slack [106].

C. Version One: A professional project management tool that is easy to use and enables Agile teams to work at all levels to deliver a good-quality software Taiga [106].

D. Asana: A project management tool characterized with high ratings by customers and covering most of the project management tasks and being available on both iOS and Android systems [106], [107].

E. Atlassian Jira: A comprehensive tool characterized with direct support and availability on iOS and Android systems. It can be integrated with a number of tools, such as PDF View Plugin, Script Runner, and Draw.io Diagrams [108]. It has been highly rated by users and is seen to have a major market share. Atlassian Jira has a set of tools for project management, such as Jira software, which is used by Agile teams to plan, track, and release high-quality software. Jira Align is also applied to connect teams for strategic alignment at the enterprise level [106], [107].

2. Collaboration tools

Cooperation among members is an essential practice to ensure project workflow, and it is one of the Agile principles. In the Global development and maintenance process, collaboration requires a number of tools to keep all team members up-to-date about the project progress. In below, most popular collaboration tools are listed according to the literature.

A. Confluence: A collaboration tool that can work in one place from virtually anywhere, and which teams can use to create and discuss project workflow. This tool is compatible with Jira, Trello, Dropbox, and office [14].

B. Trello: A collaboration tool based on the Kanban approach, it is compatible with other tools, such as Jira, Bitbucket, and Confluence and available on mobile platforms [107].

C. Bitbucket: A code collaboration tool to create and deploy code among team members. It can also be used to manage Git repositories. Bitbucket is compatible with Jira [14].

D. Bamboo: A free, open-source code collaboration tool that provides flexibility, scalability, and reliability among team members. It is compatible with Bitbucket and Jira, leading to increased traceability in projects [14].

E. FishEye: One of the collaboration tools suitable for searching and tracking code changes in repositories [14].

F. Github: A tool to create, deliver, and maintain software, Github provides the team with the ability to track the code changes and maintain synchronization among team [109].

G. Wikis: A social networking tool for collaboration among team members, characterized with ease-of-use no need for skills to use it, thus allowing users to communicate and cooperate among one another [14], [110].

H. Dropbox: A hosting service for storing, syncing, and sharing files. It is a collaborative tool that manages tasks, tracks files, and keeps teams synchronized [14].

3. Communication tools

Communication among team members is one of the most important factors to enhance collaboration in Agile methods, not to mention one of the most important principles in Agile [12].

Here is a review of the most important tools used in communication among members, especially in global development and maintenance.

A. Hipchat is a free tool developed by Atlassian to support communication among team members by contacting and sharing information and files. Hipchat can be installed on multi-operating systems, such as Linux, Windows, Mac, iOS, and Android [14].

B. Skype is a telecommunication app that provides voice and video communication, as well as text messages. It can be installed on Windows, Mac, and Linux systems, as well as mobile platforms such as iOS and Android [14].

C. Zoom is a communications software available on operating systems as Windows, mac, Linux, iOS, and Android. The team can communicate by either voice or video call in this way [111].

D. G-Suite: it is also known as 'Google Apps' or 'Google Workspace'; it is a collaboration and communication tool consisting of a set of apps, such as Gmail, Contacts, Calendar, Meet and Chat for communication; along with Currents for employee engagement; Drive for storage; and the Google Docs suite for content creation [112].

4. Testing tools

The testing in the Agile method depends on the application of Agile principles in the development process. One of the essentials of Agile testing is involving customers in the early stages of the project. One of the benefits of using automated testing tools is to speed up the productivity and reduce the total cost [113]. Below is a review of most common testing tools used in Agile project.

A. Selenium is an open-source tool for functional tests to be written for many programing languages such as: C#, Java, PHP, Python, and Ruby. This tool can be installed on many operating systems, such as Windows, Linux, and Mac [114] .

B. NUnit is a unit test framework tool for all NET Framework family and Mono, supporting Data Driven Test (DDT). NUnit is an open-source software [115].

C. Junit is a simple framework for Java used for test units, supporting Test Driven Development. Junit is compatible with Git Hup, and it is an open-source software [116].

D. Worksoft is one of the most popular Agile testing tools, easy-to-use and enabling non-technical users to automate the testing process [117].

E. **PractiTest** is one of test management tools in Agile methods that is easy-to-learn and use at a reasonable price. This tool is compatible with many other tools, such as JIRA, Jenkins, Selenium and TestComplete [118].

F. **JunoOne** is a multitasking tool used for Agile test-case management processes. This tool is compatible with JIRA and provides a number of features that facilitate the testing process [119].

5. Documentation tools

Documentation is one of the most neglected aspects in Agile projects as the focus is more on the project itself rather than its comprehensive documentation. However, this does not mean that there is no need for documentation, as there are many tools available to simplify and manage this process, as reviewed in the following for the most important ones [120].

A. Confluence is one of the tools previously defined in the Collaboration Tools section. This is one of the most popular tools for Agile documentation methods [14].

B. Dropbox Paper is a web-based tool available within the Drobox tools. It allows teams to organize and view text, media, and files in one place. Professionals can access their file through the Internet in this way [121].

C. GitHub is a comprehensive tool that provides many features that enable developers and maintainers to document the process [14].

D. Office 365 is a project documentation tool for managers and other team members to find and search for files and other resources online through the Internet [122].

E. Jira is one of the popular tools in Agile projects which provides many features for project management. Jira is also used for documentation purposes. Table 5.3 summarize all tools.

Table 5. 3: Most popular Agile tools used for Agile project management.

Management tools	**Collaborations tools**	**Communications tools**	**Testing tools**	**Documentation tools**
Assembla	Confluence	Hipchat	Selenium	Confluence
Taiga	Trello	Skype	NUnit	Dropbox Paper
Version One	Bitbucket	Zoom	JUnit	GitHub
Asana	Bamboo	G-Suite	Worksoft	Office 365
Atlassian Jira	Github		PractiTest	Jira
	Wikis		JunoOne	
	Dropbox			

5.3. Experimental Setup and Case Study

We will review the experimental setup of ASMDCC framework after expanding it and adding the maintenance process.

5.3.1. Experimental Setup for the ASMDCC Framework

As explained in the previous sections, the comprehensive framework consists of four stages containing the necessary infrastructures and tools to manage Agile projects. These tools are installed with the help of the Cloud Computing platform. The comprehensive framework has different roles for various individuals – developers, maintenance practitioners, testers, and users. Each will have their own role and tools.

We followed the steps we identified in the ASMDCC framework to set up the environment, starting with selecting the Cloud Computing platform, where the Contabo platform was chosen, this platform provided the necessary infrastructure and tools. In the second step, the project's purpose was determined; in our case study, the maintenance team will maintain the existing system, which will be explained later. In the third step, the Scrum method was followed as one of the Agile methods by the maintenance team, and the necessary practices were selected for the maintenance process. In the fourth step, many necessary tools were selected to complete the maintenance project. The above description can be seen in Figure 5.3. The environment is created using one of the Cloud Computing platforms and launched on two servers. The Windows server is installed on each server along with the necessary tools to go with.

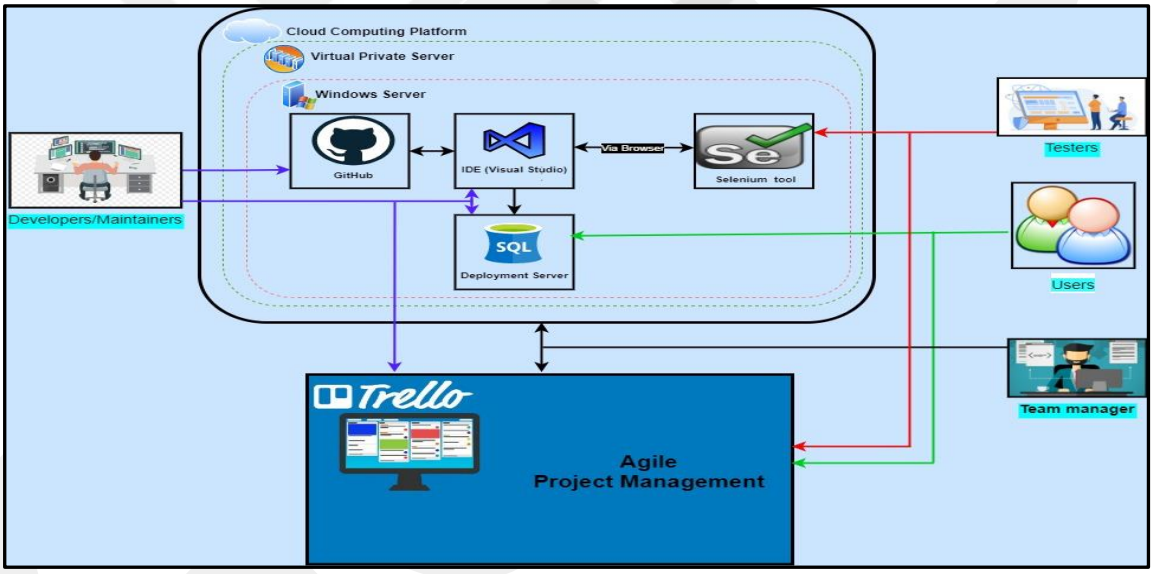


Figure 5.3:Conceptual Diagram for ASMDCC Framework.

As a pilot test, a conceptual diagram was shared with several Agile professionals, who gave valuable feedback about the environment.

The main components and roles of the ASMDCC framework are described in what comes next before proceeding with the case study.

A. Team roles

- **Team manager:** She or he has full access to all resources and tools within the environment. The black lines represent the team manager's role in the conceptual diagram.
- **Testers:** They have access to the testing tool and the project management tool Trello. Their role can be seen in red in the framework's conceptual diagram.
- **Developers/ Maintainers:** They have access to IDE, GitHub, SQL server, and Trello. Their role can be seen in blue inside the framework's conceptual diagram.
- **Users:** They have limited access to the Trello and SQL Server, seen in green in the diagram.
- **Zoom:** An application used by members to communicate among themselves.

B. The tools and components

- **Visual Studio Community:** It is a free integrated development environment provided by Microsoft.
- **Trello:** It is a project management tool provided by Atlassian.
- **Selenium:** It is a testing tool available using the browser.
- **SQL Server:** this is used for deeply the project.
- **GitHub:** It is used for storing, tracking, and collaborating on a software project.

5.3.2. Case Study Description

The case study is conducted on an already existing system – The Electronic Store – which is simply an online platform or website through which products can be sold for clothes.

A. Software Company Overview

It is a small Turkish software company building and maintaining systems. They designed and programmed many applications and systems on various platforms and operating systems, such as Web-based, Android, Apple Store, and Cloud-based apps.

B. Maintenance Team Overview

The case study involves a team working for a small software company and comprising:

- A team manager with more than ten years of experience; and

- Four maintainers with more than two years of experience.

C. Electronic Store System Overview

The Electronic Store provides a complete automation system - that is, materializing the sale process automatically and without the store owner's manual intervention – through:

- Displaying products, their specifications, and prices to the customer using pictures, texts, and perhaps videos;
- Online payment; and
- Linking with shipping companies that deliver the product to the customer.

This system was built using the ASP.NET core, Version 6. The customer has required Perfective Maintenance and extending the Electronic Store for multi-vendors.

D. Maintenance Process Description

In order to explain the Cloud environment to the team, two meetings were held with the team manager, and the environment was clarified in detail to carry out the maintenance tasks for the project.

The team initiates the maintenance process using the Agile Scrum method, through which the team collects the Product Owner's requirements (whose role was represented by one of the team members). Then, the Product Owner set up the Product Backlog, and the team arranged for Sprint Planning.

The Sprint contains the new functionality that the team should develop and during which the team meets daily (Daily meeting practice) to see the project workflow. The person responsible for monitoring the effectiveness of the team during the workflow is the Scrum

Master. Figure 5.4, shows the Scrum process used by the team and more details about Scrum practices.

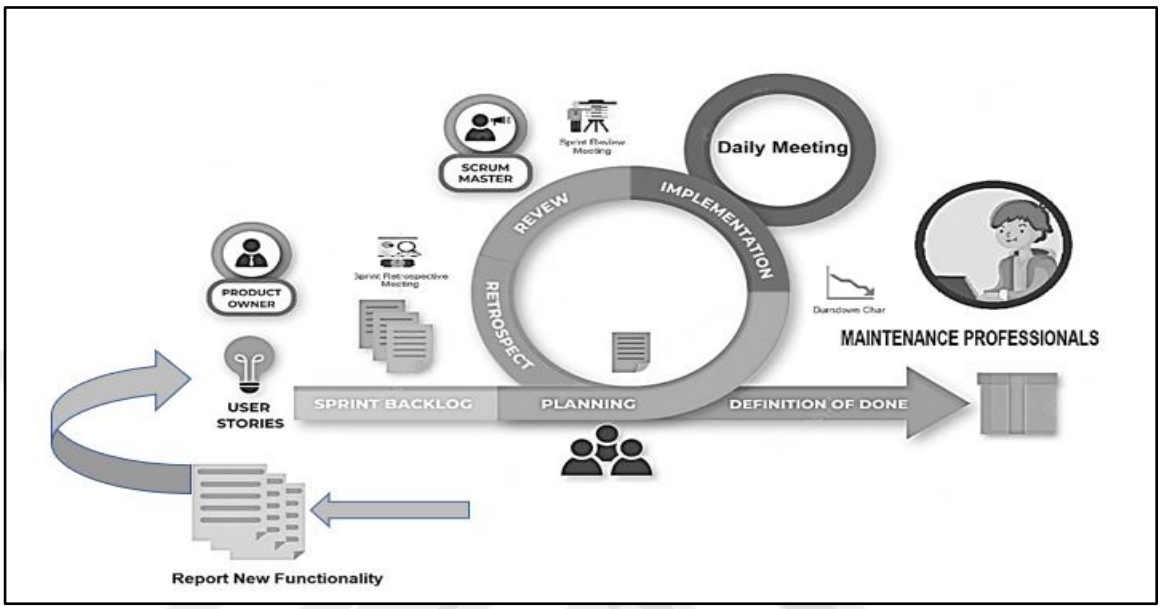


Figure 5.4 :Scrum Process

5.3.3. ASMDCC Framework Evaluation

In order to evaluate the ASMDCC framework in the maintenance process, the team will use the environment to maintain the system described earlier according to the user requirements. After maintaining the system, the team will describe a particular event or experience and explain the advantages and disadvantages of the environment according to pre-defined factors; namely, manageability, infrastructure, collaboration, communication, transparency, and lastly the impact of using tools on Agile practices.

A semi-structured interviews are carried out to collect data from the team by preparing in advance the questions to be asked [123].

More than four meeting was conducted using Zoom and other communication tools, and several questions were asked of the team about the impact of using the Cloud environment on the maintenance process. See Appendix B.

5.3.3.1. Comparing Scrum-based project maintenance with and without Cloud Computing

This section will explain the results of the case study conducted using two scenarios. The first scenario is performing maintenance in the traditional environment, and the second one is performing maintenance in the ASMDCC framework. The evaluation focuses on five primary factors identified earlier in Chapter 3 based on the challenges faced by teams in the local and global environment. These factors are manageability, scalability, infrastructure, collaboration and communication, and transparency.

- **Manageability:** Managing software projects is one of the essential factors in the Scrum method, and many practices require good management to be completed well.

  The questions were asked in an interview regarding manageability and which environments provide tools to support project management. The team's answer, as reported indirectly, was that the Cloud environment provides necessary tools that greatly support the management process and contribute to facilitating it. The team mentioned that using the Trello tool facilitated the project management process during the maintenance process.

- **Scalability:** It is necessary for maintenance, as software optimizations require additional resources, such as storage capacity and servers.

  The team manager was asked about which environments are easier for scaling resources. The answer, reported indirectly, was that Cloud Computing provides resources according to the user's request with little time and effort compared to the traditional environment, which is costly and requires more time and effort to scale up resources.

- **Infrastructure:** The infrastructure for developing and maintaining the system needs devices and resources, such as servers and networks, along with experts to configure and set them up. The team manager mentioned that, in traditional environments, the infrastructure preparation needs more time and effort than preparing such infrastructure in the Cloud platform, where the user needs to have

an account on the platform and to specify the required specifications. In this way, the Cloud Service Provider (CSP) can take the necessary action quickly.

- **Communication and Collaboration:** The most critical Agile practices are based on communication and collaboration among team members and communication with the customer.

  The team has explained the usefulness of the ASMDCC framework in supporting communication and collaboration, confirming that the environment provides the appropriate tools for collaboration among the team members through Trello and GitHub. It was further reported that the communication tools are available both in the traditional environment and in the Cloud environment.

- **Transparency:** It is an essential factor in an Agile project, taken into account in many tasks, such as daily meetings, code reviews, and Sprint reviews.

  The team mentioned that using Trello, GitHub, and Zoom tools within the ASMDCC framework improved the transparency among team members because each member can see their tasks and the project's progress.

Table 5.4 presents a comparison between the traditional environment and a Cloud-based environment according to the point of view of the maintenance team interviewed for the present work.

Table 5.4: Comparison between traditional environment and ASMDCC

Factors	**Sub- Factors**	**Traditional environment**	**Cloud-based environment**	**Team Comments**
1. Manageability	Provide task management, file sharing, and scheduling	☒	☑	Trello simplified the project management process.
2. Scalability	Increase the number of resources	☒	☑	-

Table 5.4 (cont'd). Comparison between traditional environment and ASMDCC

Factors	Sub- Factors	Traditional environment	Cloud-based environment	Team Comments
3. Infrastructure	Automatic infrastructure maintenance and updates	☒	☑	-
	Software hosted in the cloud	☒	☑	-
	No software licensing costs	☒	☑	-
	Easy to implement and use	☒	☑	-
4. Communication & Collaboration	Simplify team collaboration	☒	☑	Both environments provide communication tools.
	Provide Communication tools	☑	☑	
5. Transparency	Providing necessary tools that increase transparency	☒	☑	The proposed environment provided the necessary tools to increase the transparency of the project.
6. Additional factors	Reduced implementation costs	☒	☑	-The team manager mentioned that connecting to Cloud Computing requires Internet availability. -According to the team members, there are security issues regarding the Cloud.
	Access the environment anytime and anywhere	☒	☑	
	Data Recovery	☒	☑	
	Security	☒	☑	

The team manager indicated that the team prefers to use the hybrid method (i.e., using IDE on a local device and other tools on the Cloud) due to the need for high-speed internet.

In addition, the team was asked about the impact of the Cloud-based environment on Scrum practices, and the results are shown in Table 5.5.

This evaluation of the ASMDCC framework represents only the current case study. It can only be generalized if it is verified by more than one system and used by many maintenance professionals.

Table 5.5: The impact of using Agile tools on Scrum practices according to team members

Scrum Practices	**Tools**
1- Sprint Planning meeting 2- Sprint backlog 3- Product backlog 4- Separate backlogs for each team 5- Sprint review 6- Task Prioritization	Trello simplified applying these practices by the team.
1-Daily meeting 2- Retrospective meeting	The use of Zoom contributed to applying these practices.
1-Pair Programming 2- Code Reviews	The use of GitHub contributed to applying these practices. Also, the team can use the live-share extension in visual studio to achieve pair programming.
1-Continuous integration	The use of GitHub contributed to applying these practices.
1-Unit test 2-Automated testing	Selenium and Junit (an extension of visual studio) were used to apply these practices.

CHAPTER 6

DISCUSSION

The methods applied in the Agile approach [124], have several characteristics that make the development process simply more “Agile”, as the term itself implies. These characteristics are: cooperativeness (i.e., focusing on people satisfaction rather than process), adaptiveness (response to rapid change in the business environment), iteration (small and frequent deliveries to the customers), and light documentation. Additionally, each development cycle requires only two to four weeks; the design is simple and can be carried out by a small team; there is direct and face-to-face communication; collective ownership is possible related to the code; and, lastly, the code quality itself is improved.

Abrahamsson et al. [53], highlight different kinds of Agile approaches, such as Extreme Programming (XP), Scrum, Crystal family of methods, Feature-Driven Development, the Dynamic System Development method, and Adaptive Software Development. Each one of these approaches offers different benefits to developers.

The Agile method has twelve principles [3], that involve certain practices to simplify the software development processes. However, this method has faced some challenges; namely, the need for face-to-face communication [125], availability of maintainer experts [126], and the ability to develop and maintain the software in the distributed environment [29]. One of the proposed solutions to overcome these problems is the use of Cloud Computing, which facilitates software development by reducing the total cost of development by utilizing features as sharing data, distributed system applications, and re-prioritization of tasks [29]. In addition, the Cloud Computing environment eliminates the need for installation and configuration processes related to software patches [29]. Another advantage of the Cloud is the utilization of a pay-per-use approach [29], Cloud Computing improves software development using the Agile methods by providing increased productivity, reducing development costs, and improving software quality [7], [127].

In the following, we present the application of the Agile methods in the Cloud Computing environment along with some of the benefits realized as per the available literature:

Cloud Computing:

- improves the quality factors;
- facilitates cooperation among team members;
- increases transparency among the teams;
- provides the infrastructure for the development process; and
- facilitates the traceability in the distributed environment.

All these benefits set up the motivation behind this thesis to study the possibility of using the software maintenance process along with the Cloud Computing environment [15], [13], [29], [51], [71]. As it is well-known, the maintenance process is one of the longest stages of the software life cycle [9]. Therefore, the main motivation for conducting this study is to address the other aspects of this approach by constructing several research questions that have not been considered by earlier studies. Put differently, this study attempts to explore the benefits and challenges of Agile methods for both local and global environments to investigate the possibility of applying it in the Cloud environment to simplify the maintenance process.

To do so, four questions were formulated related to the research objectives. In order to answer the research questions, a series of scientific procedures were followed. In the first step, systematic mapping was applied where nine questions are answered by investigating 48 articles related to Agile maintenance. The primary purpose of this practice is to answer the first, second, and third questions.

In the second step, a survey was conducted in which 56 Agile professionals participated, and through the survey results, the third research question was answered and verified.

In the third step, the maintenance team conducted a case study to evaluate the ASMDCC framework to answer the fourth question of this research. Each question is reviewed and discussed below.

RQ1: What are the different Agile practices that can be used to facilitate the maintenance process in the Cloud environment?

The Agile methods involve many practices devoted to 12 principles, some of which can be used locally while others need to be adopted for the global environment. Twelve practices were identified for Scrum and XP [7], [58], [35], [8], [54], [59], [55], [56], [60], [57]. Due to the differences between the on-premises environment and Cloud, Agile practices need to adapt to Cloud Computing environments accordingly.

In distributed and global software development, there are numerous challenges, such as communication among team members [128], [129], [130], [131], lack of physical proximity [132], [133], team cohesion [134], shared context and knowledge [129], [133], and unavailability of team members [133]. For establishing collaborative relationships among the team members, frequent visits are made [135]. There are two types of visits: seeding and maintaining. Seeding visits occur during the early stages in the project. Their aim is to build connections [135], [55] and are held during the early stages of the development cycle [129]; whereas, maintaining visits are shorter and aim to build collaboration connections [135].

Ramesh et al. [129] suggest that collaboration can be established by visits of members from onshore and offshore sites. Developers, customers, and managers should meet one another to build good collaboration relationships. The challenge is the travelling of members for meeting on distributed locations. To cope, Paasivaara et al. [60] report that Agile practices using Scrum use agility supporting practices for distributed projects. These practices include frequent visits, unofficial distributed meetings, and regular gatherings. Some other agile methodologies, such as Scrum [136] and XP [137] , have successfully customized distributed projects. Scrum uses frequent and open communication; for this, they use multiple communication tools, such as video-conferencing, Internet telephony, desktop sharing, and chat for formal and informal meetings [59].

Paasivaara, et al. [59] state that frequent visits are needed, especially at the beginning of a project and in critical project phases such as testing or planning. For offshore environments, Agile development methodologies use Web-conferencing for real-time and interactive collaboration. Furthermore, white-boarding and code-sharing are used for

kick-off meetings, daily Scrum meetings, and pair-programming activities during Agile development [55].

RQ2: What are the advantages of using Agile and its practices during Software Maintenance?

According to the studies [6], [11], [35], [44], Scrum and XP practices are directed towards the maintenance process to improve customer satisfaction, increase interaction among team members, simplify testing, facilitate code review, increase code quality, and other tasks as stated in RQ7, Chapter 2. Conducting studies on the adaptation of Agile practices in the Cloud environment and their negative and positive effects is expected to facilitate future software maintenance processes.

RQ3: What are the challenges in using Agile maintenance in local and global environments?

Many challenges for using the Agile in local and global environments are stated in RQ8, Chapter 2.

After identifying the challenges through systematic mapping, these challenges were classified according to quality factors to simplify conducting the survey and obtain the Agile professionals' opinions about these challenges. The survey results were explored based on literature studies.

Agile development is iteration-based, and common changes and task lists are absent in maintenance [7]. What is more, the Agile maintenance process involves multi-tasking, and each task must be assigned to one of the team members. Hence, many management challenges appear, such as iterative development, focusing on work objectives, and team working closely. Some studies [5], [7], [9], [11], indicate these challenges under the management category. This was also confirmed by the conducted survey here as the responses referred to the existence of management problems, where the Chi-square value for management challenges was 2.762, 7.714, and 4.857, respectively for (P1, P2, and P4), are non-significant, and 13.429* for P3, which is considered significant. Additionally, the mean (2.07) and standard deviation (0.79). Clearly, the degree of Agile management

challenge is low, thus indicating homogeneity in the opinions of the study sample members on the Manageability challenge.

The Agile method is effective in small projects; however, maintenance teams face some challenges concerning scalability in a large project [12], [127], [126], [138], [139], and [140]. The challenges were further confirmed by professionals participating in the survey, showing that the mean is 2.21. The standard deviation is 0.84, and Chi-square values are 5.048, 5.048, and 3.143, respectively for (P5, P7, and P8), which are non-significant, and 7.905* for P6, which is significant.

One of the most critical challenges facing the Agile teams is the availability of infrastructure as reported by [5], and [11], and confirmed by the survey. According to the results, the mean is 2.33, the standard deviation 0.816, and the degree is medium; while the Chi-square value are; 1.238, 0.857, 4.095, 0.857, and 7.714, respectively for (P9, P10, P11, P12 and P13), are non-significant.

The challenges related to 'Communication' were confirmed by the survey, where the results showed that the degree of communication challenges is medium with a mean value of (2.17). The standard deviation is (0.87), and the Chi-square values are 8.476* for P14, which is significant, and 5.619, 3.143, respectively for P15, and P16 as non-significant. The results confirm the existence of problems in the distributed environment regarding Communication and Collaboration.

Apart from this, previous studies [5], [11], [62], [63], [56], [68], [60], [65], and [66] reported challenges regarding transparency, such as trust, control, and knowledge transfer through openness. The survey results confirmed these challenges, where the results of the Chi-Square are 2, 2, 0.286, and 4.286, respectively for (P17, P18, P19, and P20), as non-significant, the mean was 2.24, the standard deviation was (0.821), and the degree was medium.

RQ4: What are the outcomes of involving an Agile maintenance in ADCC?

In order to evaluate the performance of the ASMDCC framework and learn about the benefits of involving maintenance within the Cloud Computing, a team conducted a case study to do Perfective Maintenance and to extend the Electronic Store for multi-vendors.

The process was conducted under two scenarios: in the first one, the system was maintained using the traditional environment, and in the second one, it was maintained using the ASMDCC framework.

A semi-structured interview was conducted with the project manager, and the questions focused on the five factors identified before – that is, manageability, scalability, infrastructure, collaboration and communication, and transparency.

Concerning manageability, many studies [5], [7], [9], and [11], indicated that there are many issues related to managing maintenance projects using the Agile, such as iterative development, focusing on work objectives, and team working closely. All challenges were confirmed by the conducted survey; though, the interview results indicated that the use of the ASMDCC framework contributed to increasing the ability to manage maintenance projects through the tools provided by the environment.

The literature studies [12], [127], [126], [138], [139], and [140] indicate that there are issues related to scalability which are also verified here using the survey. The team was asked whether or not the ASMDCC framework contributed to resolving these challenges. They confirmed that using the Cloud Computing environment facilitates scaling up the resources required for the maintenance process with less effort and cost than in the traditional environment.

Many literature studies as [12], [140], [75], and [141] state that the Cloud provides the necessary infrastructure for software projects. The team further illustrated that the Cloud environment provides the necessary infrastructure for the project with minimal time and effort compared to the traditional environment.

Cloud Computing provides communication and collaboration tools [15], [16], [75], and [125]. The team explained that the Cloud environment provides collaboration tools as GitHub. In contrast, though, it was also stated that the communication tools are available in both environments.

The studies [5], [11], [37], [44], [46], [56], [60], [62], [63], [65], [66], and [68], report that there are challenges related to transparency, such as light documentation, collective ownership, and knowledge transfer through openness, control, and trust.

The interview results indicated that using the available Cloud-based tools, such as Trello and Jira, contributed to increasing project transparency by showing and accessing project progress, tasks, tracking, customer feedback, sharing data, and code review for all team members. All these practices help to increase project transparency.

CHAPTER 7

CONCLUSION

This study was conducted to include the maintenance process in Agile Development Cloud Computing (ADCC) framework, and to add necessary tools to this framework to simplify the maintenance process.

In order to achieve this goal, a systematic mapping was carried out about Agile maintenance to answer nine research questions related to the use of Agile maintenance and its benefits, Agile practices used for the maintenance processes in the local and global environment, and the challenges for Agile maintenance and their solutions. The main finding thereof reveals that only 48 studies focused on Agile maintenance during 2000-2020. Software maintenance is more complicated than software development, and this difficulty is significantly multiplied when the software is deployed in globally distant locations.

The challenges faced are communication, lack of close physical proximity, lack of team cohesion, lack of shared context and knowledge, and unavailability of team members. Agile software maintenance differs from traditional maintenance in terms of requirements, customer role, and software design.

In order to verify these challenges, a survey was conducted containing twenty questions and comprising five headings: Manageability, Scalability, Infrastructure, Communication and Collaboration, and Transparency. The results of this survey show that there are challenges regarding these factors and conforming to the challenges stated in the systematic mapping study.

According to the survey and the results obtained here, the performance of Agile maintenance models increases by adopting the Cloud Computing environment, such as computing, to facilitate the maintenance process. It is observed that the Scrum software

maintenance model increases client satisfaction. Apart from this, Extreme programming resolves the majority of issues faced during the software maintenance. The Agile maintenance models increase the degree of agility; however, the degree of agility varies across the phases: It is higher in the implementation phase and lower in the pre-implementation phase. For this reason, the Agile Development Cloud Computing framework was expanded to include the maintenance process. The ASMDCC framework was evaluated using a case study. The team performed Perfective maintenance for "The Electronic Store" in the Cloud environment. The results showed that maintenance in the Cloud contributed better than the traditional environment. The results of adopted Agile maintenance in the Cloud environment contributed to simplify the maintenance process and would help maintainers to do their work with minimum effort. So, according to the case study conducted, the ASMDCC framework contributed to increasing the level of management, simplifying infrastructure configuration, increasing scalability, improving collaboration among team members, and improving the level of transparency by showing the maintenance team of the workflow and increasing trust among members. Additionally, the tools that were used in the framework were seen to facilitate the use of Scrum practices.

Other observations are noted as follows:

- Many advantages of applying Agile practices for maintenance are stated in the systematic mapping;
- The use of the Agile methods for software maintenance can contribute to simplifying the maintenance process due to the characteristics and advantages of this methodology; and
- Several advantages are gained by adopting software development using the Agile methods in the Cloud computing environment.

7.1. Research limitations

Despite achieving the goals initially set for this research, there are certainly some limitations in different aspects, which are discussed in the following.

Regarding the systematic mapping analysis, there may be some limitations in covering all the Agile maintenance articles. Although the researcher attempted to search and collect all the relevant articles, there may still be missing ones here.

Another limitation of the research appears in the survey. The sample size could be larger; especially, if we consider the spread of the Agile methods among software companies. As it is known, a larger sample positively affects the objectivity of the results and gives an accurate idea of the community's opinion.

Lastly, regarding the case study, it would have been better to test the ASMDCC framework with more than one software company to gain more realistic ideas; nevertheless, as it is known, this matter requires cooperation from software companies – a topic that is both debatable and challenging due to time constraints, staff availability, etc. Also, one major limitation in a case study is that only a few organizations are adopting Cloud computing environments, specifically in Agile software maintenance activities.

7.2. Future Work

For future work, certain initiatives can be taken, namely:

1. Conducting several case studies using different teams to learn more about the efficiency of the ASMDCC framework from different viewpoints;

2. Conducting a survey in several companies to experiment, express an opinion about the framework, and learn more about its benefits and drawbacks for software maintenance; and

3. Increasing the number of respondents in surveys regarding Agile maintenance challenges.

4. Studying the impact of more attributes in case studies and survey to increase knowledge of the impact of using Agile methods in the Cloud environment concerning quality and productivity.

REFERENCES

[1] T. Mens and S. Demeyer, *Software evolution*. 2008. doi: 10.1007/978-3-540-76440-3.

[2] "The new new product development game," *Journal of Product Innovation Management*, vol. 3, no. 3, 1986.

[3] K. Beck *et al.*, "Manifesto for Agile Software Development," *The Agile Alliance*, 2001.

[4] R. Malhotra and A. Chug, "Comparative analysis of agile methods and iterative enhancement model in assessment of software maintenance," in *Proceedings of the 10th INDIACom; 2016 3rd International Conference on Computing for Sustainable Global Development, INDIACom 2016*, 2016.

[5] K. S. K. Ibrahim, J. Yahaya, Z. Mansor, and A. Deraman, "The Emergence of Agile Maintenance: A Preliminary Study," in *Proceedings of the International Conference on Electrical Engineering and Informatics*, 2019, vol. 2019-July.

[6] G. K. Devulapally, "Agile in the context of software maintenance: a case study." 2015.

[7] C. Poole and J. W. Huisman, "Using extreme programming in a maintenance environment," *IEEE Softw*, vol. 18, no. 6, 2001.

[8] M. Kajko-Mattsson and J. Nyfjord, "A model of agile evolution and maintenance process," in *Proceedings of the 42nd Annual Hawaii International Conference on System Sciences, HICSS*, 2009.

[9] S. Abdullah, M. Subramaniam, and S. Anuar, "Improving the Governance of Software Maintenance Process for Agile Software Development Team," 2018. [Online]. Available: www.sciencepubco.com/index.php/IJET

[10] M. M. Lehman, “Programs, Life Cycles, and Laws of Software Evolution,” *Proceedings of the IEEE*, vol. 68, no. 9, 1980.

[11] L. T. Heeager and J. Rose, “Optimising agile development practices for the maintenance operation: nine heuristics,” *Empir Softw Eng*, vol. 20, no. 6, 2015.

[12] M. Younas, I. Ghani, D. N. A. Jawawi, and M. M. Khan, “A Framework for Agile Development in Cloud Computing Environment,” *Journal of Internet Computing and Services*, vol. 17, no. 5, 2016.

[13] M. Younas, D. N. A. Jawawi, A. K. Mahmood, M. N. Ahmad, M. U. Sarwar, and M. Y. Idris, “Agile Software Development Using Cloud Computing: A Case Study,” *IEEE Access*, vol. 8, 2020.

[14] M. Younas, D. N. A. Jawawi, I. Ghani, M. A. Shah, M. M. Khurshid, and S. H. H. Madni, “Framework for Agile Development Using Cloud Computing: A Survey,” *Arabian Journal for Science and Engineering*, vol. 44, no. 11. 2019.

[15] S. Kalem, D. Donko, and D. Boskovic, “Agile methods for cloud computing,” in *2013 36th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2013, pp. 1079–1083.

[16] I. Inayat, S. S. Salim, and Z. M. Kasirun, “Agile-based software product development using cloud computing services: findings from a case study,” *Sci Int*, vol. 25, no. 4, pp. 1065–1069, 2013.

[17] M. A. Branch, M. C. Jackson, M. C. Laviolette, and E. Frankel, “Software maintenance management,” in *The Institute of Electrical and Electronics Engineers, Inc on Conference on software maintenance--1985*, 1986, pp. 62–68.

[18] T. M. Pigoski, *Practical software maintenance: best practices for managing your software investment*. Wiley Publishing, 1996.

[19] Ieee, *IEEE Standard Glossary of Software Engineering Terminology*, vol. 121990, no. 1. 1990.

[20] U. Akhlaq and M. U. Yousaf, “Impact of software comprehension in software maintenance and evolution.” 2010.

[21] J. A. McDermid, *Software engineer’s reference book*. Elsevier, 2013.

[22] N. C.; Roberts, R. T. Bradley, N. C. Roberts, and R. T. " Bradley, “Calhoun: The NPS Institutional Archive DSpace Repository Research methodology for new public management,” 2002. [Online]. Available: http://hdl.handle.net/10945/40398.

[23] P. Grubb and A. A. Takang, *Software maintenance: concepts and practice*. World Scientific, 2003.

[24] S. Mamone, “The IEEE standard for software maintenance,” *ACM SIGSOFT Software Engineering Notes*, vol. 19, no. 1, pp. 75–76, 1994.

[25] I. Sommerville, *Sommerville Software Engineering*, vol. 291. 2011.

[26] Y. Singh and B. Goel, “A step towards software preventive maintenance,” *ACM SIGSOFT Software Engineering Notes*, vol. 32, no. 4, 2007.

[27] P. Mell and T. Grance, “The NIST definition of cloud computing: Recommendations of the National Institute of Standards and Technology,” in *Public Cloud Computing: Security and Privacy Guidelines*, 2012.

[28] S. Bulusu and K. Sudia, “A Study on Cloud Computing Security Challenges,” *School of Computing Blekinge Institute of Technology*, 2012.

[29] M. Younas, D. N. A. Jawawi, I. Ghani, T. Fries, and R. Kazmi, “Agile development in the cloud computing environment: A systematic review,” *Information and Software Technology*, vol. 103. 2018.

[30] T. Dillon, C. Wu, and E. Chang, “Cloud computing: Issues and challenges,” in *Proceedings - International Conference on Advanced Information Networking and Applications, AINA*, 2010.

[31] W. Kim, “Cloud computing architecture,” *International Journal of Web and Grid Services*, vol. 9, no. 3, pp. 287–303, 2013.

[32] T. Dybå and T. Dingsøyr, “Empirical studies of agile software development: A systematic review,” *Information and Software Technology*, vol. 50, no. 9–10. 2008.

[33] W. T. Tsai, W. Wu, and M. N. Huhns, “Cloud-based software crowdsourcing,” *IEEE Internet Computing*, vol. 18, no. 3. 2014.

[34] P. Brereton, B. A. Kitchenham, D. Budgen, M. Turner, and M. Khalil, “Lessons from applying the systematic literature review process within the software engineering domain,” *Journal of Systems and Software*, vol. 80, no. 4, 2007.

[35] H. Svensson and M. Host, “Introducing an agile process in a software maintenance and evolution organization,” in *Proceedings of the European Conference on Software Maintenance and Reengineering, CSMR*, 2005.

[36] F. U. Rehman, B. Maqbool, M. Q. Riaz, U. Qamar, and M. Abbas, “Scrum Software Maintenance Model: Efficient Software Maintenance in Agile Methodology,” in *21st Saudi Computer Society National Computer Conference, NCC 2018*, 2018.

[37] B. Kumar, “The Sway of Agile Processes over Software Maintainability,” *Int J Comput Appl*, vol. 109, no. 1, 2015.

[38] M. Cardoso de Mello, “Agile processes for the maintenance cycle,” *A smarter work cycle for a Smarter Planet. IBM DeveloperWorks*, 2012.

[39] J. Choudhari and U. Suman, “Iterative maintenance life cycle using extreme programming,” in *Proceedings - 2nd International Conference on Advances in Recent Technologies in Communication and Computing, ARTCom 2010*, 2010.

[40] J. Choudhari and U. Suman, “Extended iterative maintenance life cycle using eXtreme programming,” *ACM SIGSOFT Software Engineering Notes*, vol. 39, no. 1, 2014.

[41] J. Choudhari and U. Suman, “An Empirical Evaluation of Iterative Maintenance Life Cycle Using XP,” *ACM SIGSOFT Software Engineering Notes*, vol. 40, no. 2, 2015.

[42] N. Jain, “Offshore agile maintenance,” in *Proceedings - AGILE Conference, 2006*, 2006, vol. 2006.

[43] G. K. Hanssen, A. F. Yamashita, R. Conradi, and L. Moonen, “Maintenance and agile development: Challenges, opportunities and future directions,” in *IEEE International Conference on Software Maintenance, ICSM*, 2009.

[44] J. Prochazka, “Agile support and maintenance of IT services,” in *Information Systems Development - Business Systems and Services: Modeling and Development*, 2011.

[45] S. Tarwani and A. Chug, “Agile methodologies in software maintenance: A systematic review,” in *Informatica (Slovenia)*, 2016, vol. 40, no. 4.

[46] Y. Yamato, S. Katsuragi, S. Nagao, and N. Miura, “Software maintenance evaluation of agile software development method based on open stack,” *IEICE Trans Inf Syst*, vol. E98D, no. 7, 2015.

[47] W. Reyes, R. Smith, and B. Fraunholz, “Agile approaches to software maintenance: an exploratory study of practitioner views,” *Managing worldwide operations …*, 2012.

[48] M. Agarwal and R. Majumdar, “Software Maintainability and Usability in Agile Environment,” *Int J Comput Appl*, vol. 68, no. 4, 2013.

[49] F. J. Pino, F. Ruiz, Félix García, and M. Piattini, “A software maintenance methodology for small organizations: Agile-MANTEMA,” *Journal of software: Evolution and Process*, vol. 24, no. 8, 2012.

[50] R. S. Pressman, *Software Engineering A Practitioner’s Approach 7th Ed - Roger S. Pressman*. 2009.

[51] L. Yu and A. Mishra, “Risk analysis of global software development and proposed solutions,” *Automatika*, vol. 51, no. 1, 2010.

[52] J. Rudzki, I. Hammouda, and T. Mikkola, “Agile experiences in a software service company,” in *Conference Proceedings of the EUROMICRO*, 2009.

[53] P. Abrahamson, Outi Salo, Jussi Ronkainen, and Juhani Warsta, “Agile software development methods: Review and analysis,” *VTT Publications*, 2002.

[54] S. Beecham, J. Noll, and I. Richardson, “Using Agile Practices to Solve Global Software Development Problems - A Case Study,” in *Proceedings - International Computer Software and Applications Conference*, 2014, vol. 18-21-August-2014.

[55] A. Danait, “Agile offshore techniques - A case study,” in *Proceedings - AGILE Confernce 2005*, 2005, vol. 2005.

[56] M. Kircher, P. Jain, A. Corsaro, and D. Levine, “Distributed extreme programming,” *Extreme Programming and Flexible Processes in Software Engineering*, no. September 2013, 2001.

[57] A. Qumer and B. Henderson-Sellers, “An evaluation of the degree of agility in six agile methods and its applicability for method engineering,” *Inf Softw Technol*, vol. 50, no. 4, 2008.

[58] A. Dagnino, “An evolutionary lifecycle model with Agile practices for software development at ABB,” in *Proceedings of the IEEE International Conference on Engineering of Complex Computer Systems, ICECCS*, 2002, vol. 2002-January.

[59] M. Paasivaara, S. Durasiewicz, and C. Lassenius, “Using scrum in distributed agile development: a multiple case study,” in *Proceedings - 2009 4th IEEE International Conference on Global Software Engineering, ICGSE 2009*, 2009.

[60] M. Paasivaara, S. Durasiewicz, and C. Lassenius, “Distributed agile development: Using Scrum in a large project,” in *Proceedings - 2008 3rd IEEE International Conference Global Software Engineering, ICGSE 2008*, 2008.

[61] S. Shaw, “Using Agile Practices in a Maintenance Environment,” 2007.

[62] O. McHugh, K. Conboy, and M. Lang, “Agile practices: The impact on trust in software project teams,” *IEEE Softw*, vol. 29, no. 3, 2012.

[63] S. Lee and H. S. Yong, “Distributed agile: Project management in a global environment,” *Empir Softw Eng*, vol. 15, no. 2, 2010.

[64] J. Highsmith and A. Cockburn, “Agile software development: The business of innovation,” *Computer*, vol. 34, no. 9. 2001.

[65] S. Jalali and C. Wohlin, “Agile practices in global software engineering - A systematic map,” in *Proceedings - 5th International Conference on Global Software Engineering, ICGSE 2010*, 2010.

[66] M. Talluri and H. M. Haddad, “Best managerial practices in agile development,” in *Proceedings of the 2014 ACM Southeast Regional Conference, ACM SE 2014*, 2014.

[67] M. A. Akbar, S. Mahmood, H. Alsalman, A. Razzaq, A. Gumaei, and M. T. Riaz, “Identification and prioritization of cloud based global software development best practices,” *IEEE Access*, vol. 8, 2020.

[68] K. B. Awar, M. S. I. Sameem, and Y. Hafeez, “A model for applying Agile practices in Distributed environment: A case of local software industry,” in *Proceedings of 2017 International Conference on Communication, Computing and Digital Systems, C-CODE 2017*, 2017.

[69] D. Batra, “Modified agile practices for outsourced software projects,” *Commun ACM*, vol. 52, no. 9, 2009.

[70] S. Aziz Butt, “Study of agile methodology with the cloud,” *Pacific Science Review B: Humanities and Social Sciences*, vol. 2, no. 1, 2016.

[71] M. Manuja and M. Manisha, “Moving agile based projects on Cloud,” in *Souvenir of the 2014 IEEE International Advance Computing Conference, IACC 2014*, 2014.

[72] L. Cocco, K. Mannaro, and G. Concas, “A model for global software development with cloud platforms,” in *Proceedings - 38th EUROMICRO Conference on Software Engineering and Advanced Applications, SEAA 2012*, 2012.

[73] S. A. Butt, M. I. Tariq, T. Jamal, A. Ali, J. L. D. Martinez, and E. De-La-Hoz-Franco, “Predictive variables for agile development merging cloud computing services,” *IEEE Access*, vol. 7, 2019.

[74] S. Singh and I. Chana, “Introducing Agility in Cloud Based Software Development through ASD,” *International Journal of u- and e- Service, Science and Technology*, vol. 6, no. 5, 2013.

[75] V. E. Jyothi and K. N. Rao, “Effective implementation of agile practices in collaboration with cloud computing,” *International Journal of Current Engineering and Technology*, vol. 4, no. 3, 2014.

[76] A. Bryman and E. Bell, “Business research methods: Oxford University Press,” *American journal of sociology*, 2015.

[77] S. Chapman, P. McNeill, and P. Mcneill, *Research methods*. Routledge, 2005.

[78] C. Cassell and G. Symon, “Qualitative methods in organizational research: A practical guide,” 1994.

[79] T. Dybå, R. Prikladnicki, K. Rönkkö, C. Seaman, and J. Sillito, “Qualitative research in software engineering,” *Empirical Software Engineering*, vol. 16, no. 4. 2011.

[80] M. Q. Patton, “Qualitative research and evaluation methods. Thousand Oaks,” *Cal.: Sage Publications*, 2002.

[81] H. K. MOHAJAN, "QUALITATIVE RESEARCH METHODOLOGY IN SOCIAL SCIENCES AND RELATED SUBJECTS," *Journal of Economic Development, Environment and People*, vol. 7, no. 1, 2018.

[82] L. T. Hoshmand, "Narratology, cultural psychology, and counseling research," *Journal of Counseling Psychology*, vol. 52, no. 2. 2005.

[83] D. Coghlan, "Action research: Exploring perspectives on a philosophy of practical knowing," *Academy of Management Annals*, vol. 5, no. 1, 2011.

[84] E. Weyant, "Research Design: Qualitative, Quantitative, and Mixed Methods Approaches, 5th Edition," *Journal of Electronic Resources in Medical Libraries*, vol. 19, no. 1–2, 2022.

[85] J. W. Creswell, *Qualitative Inquiry & Research Design: choosing among five approaches*, vol. 2. 2007.

[86] M. Špiláčková, "Historical Research in Social Work - Theory and Practice," *ERIS Web Journal*, vol. 3, no. 2, 2012.

[87] "Leedy, P. D., & Ormrod, J. E. (2015). Practical research. Planning and design (11th ed.). Boston, MA: Pearson.," *Journal of Applied Learning & Teaching*, vol. 1, no. 2, 2018.

[88] H. U. Chih-Pei and Y.-Y. Chang, "John W. Creswell, research design: Qualitative, quantitative, and mixed methods approach," *Journal of Social and Administrative Sciences*, vol. 4, no. 2, pp. 205–207, 2017.

[89] B. A. Kitchenham and S. L. Pfleeger, *Chapter 3 Personal Opinion Surveys - Guide to advanced empirical software engineering*. 2008.

[90] T. F. Burgess, "A general introduction to the design of questionnaires for survey research.," in *Guide to the Design of Questionnaires*, no. May, 2003.

[91] M. Selvanathan, N. Jayabalan, and N. Hussain, "Employee Productivity in Malaysian Private Higher Educational Institutions," *Journal of Archaeology of Egypt/ Egyptology*, vol. 17, no. 3, 2020.

[92] A. Garcia-Santillan, E. Moreno-Garcia, J. Carlos-Castro, J. H. Zamudio-Abdala, and J. Garduno-Trejo, "Cognitive, Affective and Behavioral Components That Explain Attitude toward Statistics," *Journal of Mathematics Research*, vol. 4, no. 5, 2012.

[93] H. F. Kaiser, "An index of factorial simplicity," *Psychometrika*, vol. 39, no. 1, 1974.

[94] C. Beckett, L. Eriksson, E. Johansson, and C. Wikström, "Multivariate Data Analysis (MVDA)," in *Pharmaceutical Quality by Design: A Practical Approach*, 2017.

[95] J. L. Pimentel, "A note on the usage of Likert Scaling for research data analysis," *USM R&D Journal*, vol. 18, no. 2, pp. 109–112, 2010.

[96] N. Bhatte, V. Prajapati, S. Varia, and J. More, "Comparison of Different Cloud Providers".

[97] P. Kaushik, A. M. Rao, D. P. Singh, S. Vashisht, and S. Gupta, "Cloud Computing and Comparison based on Service and Performance between Amazon AWS, Microsoft Azure, and Google Cloud," in *Proceedings of International Conference on Technological Advancements and Innovations, ICTAI 2021*, 2021.

[98] "Oracle | Cloud Applications and Cloud Platform." https://www.oracle.com/ (accessed Dec. 16, 2022).

[99] P. Dutta and P. Dutta, "Comparative study of cloud services offered by Amazon, Microsoft & Google," *International Journal of Trend in Scientific Research and Development*, vol. 3, no. 3, pp. 981–985, 2019.

[100] “Public Cloud Services Comparison | A simple cloud comparison chart of all the cloud services offered by the major public cloud vendors globally.” https://comparecloud.in/ (accessed Dec. 16, 2022).

[101] S. O. Barraood, H. Mohd, and F. Baharom, “A Comparison Study of Software Testing Activities in Agile Methods,” in *Knowledge Management International Conference (KMICe) Virtual Conference*, 2021, pp. 130–137.

[102] G. S. Matharu, A. Mishra, H. Singh, and P. Upadhyay, “Empirical Study of Agile Software Development Methodologies,” *ACM SIGSOFT Software Engineering Notes*, vol. 40, no. 1, 2015.

[103] A. Margini, G. Cutrona, and C. Fantuzzi, “Comparison of different agile methodologies and fit assessment in an industrial context,” 2017.

[104] F. Kanwal, K. Junaid, and M. A. Fahiem, “A hybrid software architecture evaluation method for FDD - An agile process model,” in *2010 International Conference on Computational Intelligence and Software Engineering, CiSE 2010*, 2010.

[105] S. Merzouk, S. Elhadi, H. Ennaji, A. Marzak, and N. Sael, “A Comparative Study of Agile Methods: Towards a New Model-based Method,” *International Journal of Web Applications*, vol. 9, no. 4, 2017.

[106] M. Manole and M. Avramescu, “A comparative analysis of agile project management tools,” *Economy Informatics*, vol. 17, no. 1, 2017.

[107] D. Özkan and A. Mishra, “Agile Project Management Tools: A Brief Comprative View,” *Cybernetics and Information Technologies*, vol. 19, no. 4, 2019.

[108] O. C. Buturugă, V. M. Gogoi, and I. A. Prodan, “Agile project management tools,” *Academy of Economic Studies. Economy Informatics*, vol. 16, no. 1, pp. 19–26, 2016.

[109] “GitHub: Let’s build from here · GitHub.” https://github.com/ (accessed Dec. 16, 2022).

[110] K. R. Parker, J. Chao, and R. F. Houghton, “Wikis as a Collaboration Tool,” in *Encyclopedia of Education and Information Technologies,* Springer International Publishing, 2019, pp. 1–10.

[111] “One platform to connect | Zoom.” https://zoom.us/ (accessed Dec. 16, 2022).

[112] “Google Workspace | Business apps and collaboration tools.” https://workspace.google.com/intl/en_uk/ (accessed Dec. 16, 2022).

[113] R. A. Razak and F. R. Fahrurazi, “Agile testing with Selenium,” in *2011 5th Malaysian Conference in Software Engineering, MySEC 2011,* 2011.

[114] “About Selenium | Selenium.” https://www.selenium.dev/about/ (accessed Dec. 16, 2022).

[115] “NUnit.org.” https://nunit.org/ (accessed Dec. 16, 2022).

[116] “JUnit 5.” https://junit.org/junit5/ (accessed Dec. 16, 2022).

[117] “Automated Software Testing & Business Process Testing.” https://www.worksoft.com/ (accessed Dec. 16, 2022).

[118] “PractiTest - Test Management Platform to Manage all Your QA Efforts.” https://www.practitest.com/ (accessed Dec. 16, 2022).

[119] “Juno.” https://juno.one/ (accessed Dec. 16, 2022).

[120] S. Voigt, D. Huttemann, and A. Gohr, “SprintDoc: Concept for an agile documentation tool,” in *Iberian Conference on Information Systems and Technologies, CISTI,* 2016, vol. 2016-July.

[121] “dropbox.com.” https://www.dropbox.com/ (accessed Dec. 16, 2022).

[122] “Home | Microsoft 365.” https://www.office.com/?auth=1 (accessed Dec. 16, 2022).

[123] K. E. Newcomer, H. P. Hatry, and J. S. Wholey, “Conducting semi-structured interviews,” *Handbook of practical program evaluation*, vol. 492, p. 492, 2015.

[124] D. Moher *et al.*, “Preferred reporting items for systematic review and meta-analysis protocols (PRISMA-P) 2015 statement,” *Revista Espanola de Nutricion Humana y Dietetica*, vol. 20, no. 2, 2016.

[125] M. R. J. Qureshi and I. Sayid, “Scheme of global scrum management software,” *International Journal of Information Engineering and Electronic Business*, vol. 7, no. 2, p. 1, 2015.

[126] A. Nazir, A. Raana, and M. Fahad Khan, “Cloud Computing ensembles Agile Development Methodologies for Successful Project Development,” *International Journal of Modern Education and Computer Science*, vol. 5, no. 11, 2013.

[127] A. Tuli, N. Hasteer, M. Sharma, and A. Bansal, “Empirical investigation of agile software development: cloud perspective,” *ACM SIGSOFT Software Engineering Notes*, vol. 39, no. 4, pp. 1–6, 2014.

[128] S. Berczuk, “Back to basics: The role of agile principles in success with an distributed scrum team,” in *Proceedings - AGILE 2007*, 2007.

[129] B. Ramesh, L. Cao, K. Mohan, and P. Xu, “Can distributed software development be agile?,” *Communications of the ACM*, vol. 49, no. 10. 2006.

[130] M. Farmer, “DecisionSpace Infrastructure: Agile development in a large, distributed team,” in *Proceedings of the Agile Development Conference, ADC 2004*, 2004.

[131] E. Therrien, “Overcoming the Challenges of Building a Distributed Agile Organization,” 2008.

[132] H. Holmstrom, E. Ó. Conchúir, P. J. Ågerfalk, and B. Fitzgerald, "Global software development challenges: A case study on temporal, geographical and socio-cultural distance," in *Proceedings - 2006 IEEE International Conference on Global Software Engineering, ICGSE 2006*, 2006.

[133] M. Yap, "Follow the sun: Distributed extreme programming development," in *Proceedings - AGILE Confernce 2005*, 2005, vol. 2005.

[134] M. Seyyedattar, S. Zendehboudi, and S. Butt, "Technical and Non-technical Challenges of Development of Offshore Petroleum Reservoirs: Characterization and Production," *Natural Resources Research*, vol. 29, no. 3. 2020.

[135] K. Braithwaite and T. Joyce, "XP expanded: Distributed extreme programming," in *Lecture Notes in Computer Science*, 2005, vol. 3556.

[136] K. Schwaber and M. Beedle, *Agile software development with scrum. Series in agile software development*, vol. 1. Prentice Hall Upper Saddle River, 2002.

[137] K. Beck, *Extreme Programming Explained: Embrace Change*. 1999.

[138] A. Dumbre, S. P. Senthil, and S. S. Ghag, "Practising agile software development on the windows azure platform," *Infosys Whitepaper*, 2011.

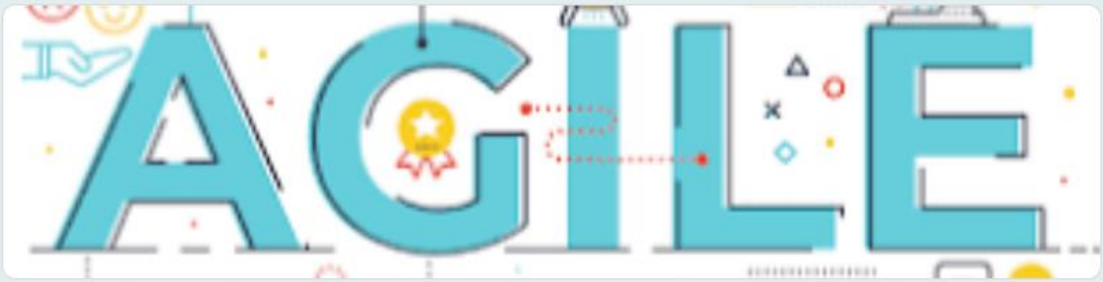
[139] F. Almudarra and B. Qureshi, "Issues in adopting agile development principles for mobile cloud computing applications," in *Procedia Computer Science*, 2015, vol. 52, no. 1.

[140] B. P. Gopularam, C. B. Yogeesha, and P. Periasamy, "Highly scalable model for tests execution in cloud environments," in *2012 18th Annual International Conference on Advanced Computing and Communications, ADCOM 2012*, 2012.

[141] S. Karunakaran, "Impact of cloud adoption on agile software development," in *Software Engineering Frameworks for the Cloud Computing Paradigm*, Springer, 2013, pp. 213–234.

APPENDIX A

Survey Questions



Agile Maintenance challenges in Software Development Companies

mhycs91@gmail.com (not shared) Switch account

* Required

Section A: Demographics(General Information)

Email address

Your answer

Country: *

Choose

Job title industry / Academic:

Your answer

How many years of industry/academia experience do you have in your field?

- ◯ 0-4 (years)
- ◯ 5-7 (years)
- ◯ 8-10 (years)
- ◯ more than 10 (years)
- ◯ Other: _____

Could you specify the organization size?

- ◯ Micro (<10 employees)
- ◯ Small (<50 employees)
- ◯ Medium (<250 employees)
- ◯ 1000+
- ◯ 2200
- ◯ 30000+

Could you specify the organization's nature?

- ◯ National
- ◯ International
- ◯ Others

Does your organization use Agile for software maintenance? *

- ◯ Yes
- ◯ No

Is your organization use Cloud with Agile maintenance?

- ◯ Yes
- ◯ No

How long has your organization used Agile for software maintenance? *

- ◯ 0-4 (years)
- ◯ 5-7 (years)
- ◯ 8-10 (years)

Canon MG3600 series Print

Could you specify the organization's nature?

- ◯ National
- ◯ International
- ◯ Others

Does your organization use Agile for software maintenance? *

- ◯ Yes
- ◯ No

Is your organization use Cloud with Agile maintenance?

- ◯ Yes
- ◯ No

How long has your organization used Agile for software maintenance? *

- ◯ 0-4 (years)
- ◯ 5-7 (years)
- ◯ 8-10 (years)

Canon MG3600 series Print

Which method is used to carry out the maintenance process?

- ◯ On-site (Local).
- ◯ By remote access to the customer servers (Global).
- ◯ Depending on the nature of the fault, use (Local or Global).
- ◯ Support
- ◯ Local, company-remote or cloud provider remote
- ◯ Other: _____

Back Next Clear form

Section B - Agile software maintenance challenging.

This section aims to specify the challenging factors in agile maintenance that faced software maintainers. Please rank each challenge factor according to your understanding and experience.

B1 Manageability (HR) (Please, state in global or local) *

Please, mark the challenges that have faced maintenance for local or global or both of them. If there is no challenge, mark "None".

	Local	Global	None
P1: Challenges related to the availability of experts for configuring software and hardware resources and Agile maintenance experts	☐	☐	☐
P2: Lack of technical support for the improvement of agile maintenance practices.	☐	☐	☐
P3: Weakness in managing maintenance team projects	☐	☐	☐

P4: Challenges related to lack of management commitment to support agile maintenance members	☐	☐	☐

Any more challenges that related to Manageability in Agile Software maintenace?

Your answer

B2 Scalability and Interoperability in Agile Software Maintenance. *

Please, mark the challenges that have faced maintenance for local or global or both of them. If there is no challenge, mark "None".

	Local	Global	None
P5: Challenges related to software's ability to interact with other systems in organization	☐	☐	☐
P6: Challenges related to frequent planning of interactions among team members	☐	☐	☐
P7: Challenges related to organizing large-size projects in agile software maintenance	☐	☐	☐
P8: Challenges related to documentation of large-size projects in agile software maintenance	☐	☐	☐

Any more challenges that related to Scalability and Interoperability in Agile Software Maintenance?

Your answer

P6: Challenges related to frequent planning of interactions among team members	☐	☐	☐
P7: Challenges related to organizing large-size projects in agile software maintenance	☐	☐	☐
P8: Challenges related to documentation of large-size projects in agile software maintenance	☐	☐	☐

Any more challenges that related to Scalability and Interoperability in Agile Software Maintenance?

Your answer

B3 Software Infrastructure *

Please, mark all the challenges that have faced maintenance for local or global or both of them. If there is no challenge, mark "None".

	Local	Global	None
P9: Challenges related to configuration infrastructure for implementing software maintenance	☐	☐	☐
P10: Challenges related to the availability of infrastructure and resources in Agile software maintenance projects	☐	☐	☐
P11: Challenges related to necessary software tools for artifacts management in agile software maintenance	☐	☐	☐
P12: Challenges related to the necessary testing server for both frequent and automated tests in for Agile software maintenance	☐	☐	☐
P13: Challenges related to the necessary server for frequent delivery of software	☐	☐	☐

Any more challenges that related to Infrastructure in Agile Software Maintenance?

Your answer

B4 Communication and Collaboration *

Please, mark the challenges that have faced maintenance for local or global or both of them. If there is no challenge, mark "None".

	Local	Global	None
P14: Challenges concerning geographically distributed teams to reduce the issues related to communication, coordination. Collaboration, etc. among team members	☐	☐	☐
P15: Challenges related to using advanced tools and techniques for continuous communication between team members	☐	☐	☐
P16: Challenges related to obtaining customer feedback and involvement in projects	☐	☐	☐

B5: Transparency *

Please, mark the challenges that have faced maintenance for local or global or both of them. If there is no challenge, mark "None".

	Local	Global	None
P17: Challenges related to providing source code management in agile software maintenance	☐	☐	☐
P18: Challenges related to sharing data, code, built results and test reports in agile software maintenance	☐	☐	☐
P19: Challenges related to traceability mechanism for project artifacts in agile software maintenances	☐	☐	☐
P20: Challenges related to the client's monitoring the progress of software maintenance in Agile projects	☐	☐	☐

Any more challenges that related to Transparency in Agile Software Maintenance?

Your answer

Back Next Clear form

APPENDIX B

Interview Questions

Factors	Questions	Traditional environment	Cloud-based Environment
1. Manageability	Which environment is easier to install the necessary applications and tools to manage maintenance projects?	☒	☑
2. Scalability	Which environment is easier and cheaper in scaling up resources (Storage, CPU, network, etc.)?	☒	☑
3. Accessibility	Which environment is more accessible from different locations?	☒	☑
4. communication and collaboration between team	Which is the best environment for providing Code collaboration tools?	☒	☑
	Which is the best environment for providing communication tools?	☑	☑
5. Infrastructure	Which environment is easier to set up and configure the resources used for software development and maintenance?	☒	☑
	Which environment needs more staff to configure it?	☑	☒
6. Transparency	Which environment supports integration transparency and traceability tools within a project?	☒	☑
7. Additional Comments	The team manager mentioned that he prefers to use the hybrid environment (install IDE on the local machine and all other tools on Cloud) due to internet speed problems.		