

Agile-Based Approach for Efficient and Quality-Oriented Development of DApps.



MCS

Author

Muhammad Haroon Alam

Registration Number

00000362161

Supervisor

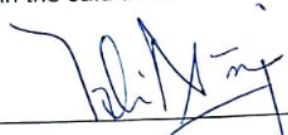
Assoc Prof Dr. Fahim Arif


A thesis submitted to the faculty of the Department of Computer Software Engineering, Military College of Signals, National University of Sciences and Technology (NUST), Rawalpindi in partial fulfillment of the requirements for the degree of MS in Software Engineering.


(January, 2024)

THESIS ACCEPTANCE CERTIFICATE

Certified that final copy of MS Thesis written by **Mr. Muhammad Haroon Alam**, Registration No. **00000362161**, of **Military College of Signals** has been vetted by undersigned, found complete in all respects as per NUST Statutes/Regulations is free of plagiarism, errors, and mistakes and is accepted as partial fulfillment for award of MS degree. It is further certified that necessary amendments as pointed out by GEC members of the scholar have also been incorporated in the said thesis.

Signature: 
Name of Supervisor Assoc Prof Dr. Fahim Arif ✓
Date: 19-01-2024

Signature (HOD):  Brig
Head of Dept of CSE
College of Sigs (NUST)
Date: 22/1/24

Signature (Dean/Principal)  Brig
Dean, MCS (NUST)
(Asif Masood, Phd)
Date: 25/1/24

Declaration

I, **Muhammad Haroon Alam**, declare that this thesis titled "Agile-based Approach for Efficient and Quality-oriented Development of DApps" and the work presented is my own and has been generated by me as a result of my own original research.

I confirm that:-

1. This work was done wholly or mainly while in candidature for a Master of Science degree at NUST.
2. Where any part of this thesis has previously been submitted for a degree or any other qualification at NUST or any other institution, this has been clearly stated.
3. Where I have quoted from the work of others, this is always clearly attributed.
4. Where I have quoted from the work of others, the source is always given. Except for such quotations, this thesis is entirely my work.
5. I have acknowledged all main sources of help.
6. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Author Name: **Muhammad Haroon Alam**

Signature: _____



Dedication

This research work is dedicated to my father (late) who continues to be my source of strength, and to my brave mother, siblings, friends, and fellows, who have all been my endless source of love, encouragement, and strength. Your unwavering belief in my abilities, countless sacrifices, and relentless support have been the foundation upon which I built my academic pursuits. This research work would not have been possible without their love and support.

Acknowledgments

All worship and glory be to the All-Magnificent and All-Merciful Allah Almighty. I am deeply grateful to Allah Almighty for granting me the capability and determination to pursue and complete this research. Allah Almighty's divine blessings and guidance have been instrumental in overcoming obstacles and achieving success. I humbly acknowledge that no words can fully express my gratitude for the countless blessings bestowed upon me. I dedicate this thesis as a humble tribute to Allah Almighty, recognizing His infinite wisdom and benevolence. I pray that my work may benefit others and be pleasing to Allah Almighty.

I would also like to express my heartfelt appreciation to my thesis supervisor, **Assoc Prof Dr. Fahim Arif**, for his unwavering support and guidance throughout my thesis. His knowledge, expertise, and dedication to his field have been a source of inspiration to me, and I am grateful for the time and effort he invested in my success. From the beginning of my journey until the end, he has been an embodiment of kindness, motivation, and inspiration towards me.

In addition, I extend my gratitude to my GEC committee members, **Assoc Prof Dr. Tauseef Ahmad Rana** and **Asst. Prof. Dr. Muhammad Sohail**, for their continuous availability for assistance and support throughout my degree, both in coursework and thesis. Their expertise and knowledge have been invaluable to me, and I am grateful for their unwavering support and guidance.

Abstract

Blockchain Oriented Software Engineering (BOSE) introduces new research directions for the systematic, disciplined, and quantifiable approaches to the development of blockchain-based software systems i.e., Decentralized Applications (DApps). The development lifecycle of DApps is different and complex as compared to traditional software systems. BOSE develops new software engineering approaches to manage the development of DApps, these approaches are hybrid and based on standards that are being followed in software engineering to manage the development of traditional software systems with collaborations new trends, and architecture of DApps. This thesis contributes to this area of research by introducing software engineering approaches such as agile software development and project management methods to provide efficient and quality-oriented development of blockchain-enabled smart contracts-based DApps systems. This thesis proposes a hybrid framework to ensure specified, simple, and standardized agile-based management of efficient and quality-oriented development of DApps. The proposed framework facilitates efficiency and quality through specified project goals, management of resource consumption of DApps systems, agile testing, simplicity in software requirements, and design process. SMART (Specific, Measurable, Achievable, Realistic, Time-bound, or Traceable) objectives, Class Responsibility Collaborator modeling, Pair Programming, Acceptance Test Driven Development, Prioritization of Requirements, and Project Velocity are practices and methods introduced in the proposed framework. A questionnaire survey in the community of software industry professionals and a case study is also conducted to implement the proposed framework in a real-time industry environment. Responses and results of the survey and case study establish the effectiveness and usability of all the practices and methods of the proposed framework for efficient and quality-oriented development of DApps systems.

Keywords – blockchain-oriented software engineering, agile software development, SMART analysis, decentralized applications, smart contracts

Table of Contents

| | | |
|----------|---|-----------|
| 1 | Introduction | 1 |
| 1.1 | Overview | 1 |
| 1.2 | Motivation | 5 |
| 1.3 | Problem Statement | 5 |
| 1.4 | Research Objectives | 6 |
| 1.5 | Contribution of Research | 6 |
| 1.6 | Thesis Outline | 8 |
| 2 | Literature Review | 10 |
| 2.1 | Introduction | 10 |
| 2.2 | Agile based Software Engineering | 10 |
| 2.3 | Blockchain and DLT | 12 |
| 2.4 | Blockchain based Smart Contracts | 14 |
| 2.5 | Smart Contracts based DApps | 18 |
| 2.6 | Quality Attributes of Development of DApps | 19 |
| 2.6.1 | Simplicity in Development of DApps | 19 |
| 2.6.2 | Agile Testing in Development of DApps | 23 |
| 2.6.3 | Efficient Development of DApps | 25 |
| 2.6.4 | Quality-oriented Sustainable Development of DApps | 27 |
| 2.7 | Blockchain Oriented Software Engineering (BOSE) | 29 |
| 2.8 | Agile based Approaches for Development of DApps | 33 |

TABLE OF CONTENTS

| | | |
|----------|--|-----------|
| 2.8.1 | Test Driven Development (TDD) | 33 |
| 2.8.2 | Acceptance Testing | 34 |
| 2.8.3 | Planning for new Release and Iteration | 35 |
| 2.8.4 | Pair Programming | 35 |
| 2.8.5 | Small Incremental Releases with Review Meetings and Project Velocity | 36 |
| 2.8.6 | Class Responsibility Collaborator (CRC) Modeling | 36 |
| 2.9 | SMART Analysis of Project Goals | 37 |
| 2.10 | Summary | 37 |
| 3 | Proposed Methodology | 38 |
| 3.1 | Introduction | 38 |
| 3.2 | Rationale and Motivation | 38 |
| 3.3 | Proposed Framework | 40 |
| 3.4 | Vision (Define Vision) | 40 |
| 3.5 | Design and Implementation | 43 |
| 3.6 | Maintenance | 46 |
| 3.7 | Key Attributes of the Proposed Framework | 47 |
| 3.7.1 | Simplicity of the Proposed Framework for the Development of DApps | 47 |
| 3.7.2 | Quality Oriented Development of DApps using the Proposed Framework | 48 |
| 3.7.3 | Agile Testing in the Proposed Framework for the Development of DApps | 49 |
| 3.7.4 | Efficient Development of DApps in the Proposed Framework | 49 |
| 3.8 | Summary | 50 |
| 4 | Validation of Proposed Framework | 51 |
| 4.1 | Introduction | 51 |
| 4.2 | Questionnaire Survey | 51 |
| 4.3 | Implementation Example - Development of DApp | 53 |
| 4.4 | Summary | 54 |

| | | |
|----------|---|-----------|
| 5 | Case Study Implementation of Proposed Framework | 55 |
| 5.1 | Introduction | 55 |
| 5.2 | Case Study for Implementation | 55 |
| 5.2.1 | Objective | 55 |
| 5.2.2 | Scope and Criteria | 56 |
| 5.2.3 | Level of Assurance | 56 |
| 5.2.4 | Description of the Project | 56 |
| 5.3 | Implementation and Analysis | 56 |
| 5.4 | Vision (Define Vision) | 57 |
| 5.5 | SMART Analysis | 58 |
| 5.6 | Split the System | 59 |
| 5.7 | Design and Implementation | 59 |
| 5.7.1 | Smart contracts Development | 59 |
| 5.7.2 | App System Development | 65 |
| 5.8 | Maintenance | 67 |
| 5.9 | Case Study Concluding Remarks | 67 |
| 5.10 | Summary | 68 |
| 6 | Results and Analysis | 69 |
| 6.1 | Introduction | 69 |
| 6.2 | Analysis of Questionnaire Survey | 69 |
| 6.2.1 | Challenges in Agile SDLC of DApps | 70 |
| 6.2.2 | In practice Agile Methodologies for the Development of DApps | 70 |
| 6.2.3 | Requirements Prioritization in the Development of DApps | 71 |
| 6.2.4 | Define Project goals using SMART Analysis | 73 |
| 6.2.5 | Use of CRC Cards in the Design Phase of Agile DApp Development | 73 |
| 6.2.6 | Influence of Pair Programming in Code Quality DApp Systems | 74 |
| 6.2.7 | ATDD to Validate and Acceptance Phase in Agile Development of DApps | 75 |

TABLE OF CONTENTS

| | | |
|----------|--|-----------|
| 6.2.8 | Simple Design Leads to Efficient Design and Requirements Elicitation in Agile Development of DApps | 76 |
| 6.2.9 | Agile Practices for Quality-oriented Development of DApps | 77 |
| 6.3 | Central Tendencies of Survey Responses on Statements of Questions | 78 |
| 6.4 | Comparison of Proposed Framework with Related Studies | 79 |
| 6.5 | Summary | 80 |
| 7 | Conclusion and Future Work | 81 |
| 7.1 | Conclusion | 81 |
| 7.2 | Future Work | 82 |
| | References | 83 |
| | Annex A | 89 |

List of Tables

| | | |
|-----|--|----|
| 2.1 | A structured comparison of financial and non-financial applications of blockchain technology [23] | 14 |
| 2.2 | Applications of smart contracts with their benefits and use cases [27] | 17 |
| 2.3 | Comparison of traditional web applications and DApps [7] | 18 |
| 2.4 | Practical advantages of different types of DApps [29] | 19 |
| 2.5 | Factors lead to simplicity with their benefits in ASD [31] | 22 |
| 2.6 | Advantages and disadvantages of Agile testing [34] | 24 |
| 2.7 | Solutions of Scalability problems and trilemma trade-offs of these solutions in blockchain system [40] | 27 |
| 6.1 | Central tendencies of survey responses on statements of questions | 79 |
| 6.2 | Comparison of the proposed framework with related studies in this area of research | 80 |

List of Figures

| | | |
|-----|---|----|
| 1.1 | A typical architecture of DApps. Ethereum blockchain on the left, DApps UI on the bottom right, and Backend server (centric server or distributed file system i.e., IPFS) on top right [4]. | 3 |
| 1.2 | Contributions of the proposed framework for the development of DApps | 7 |
| 1.3 | Thesis Chapter wise Outline | 9 |
| 2.1 | Agile Software Development Lifecycle [22] | 11 |
| 2.2 | A process of functioning from a new transaction request to the completion and execution of the transaction [25] | 13 |
| 2.3 | Phases of smart contracts life cycle, from creation to completion [27] | 16 |
| 3.1 | A framework for efficient and quality-oriented development of DApps | 41 |
| 3.2 | Key Attributes of the Proposed Framework | 47 |
| 5.1 | UML Use case diagram based on user stories. | 58 |
| 5.2 | CRC Modeling, arranged with respect to collaborators | 59 |
| 5.3 | The Standard UML Class Diagram of Supply Chain Management DApp System of Grapes Juice | 60 |
| 5.4 | UML Class diagram, based on .sol files of smart contracts of Supply Chain Management DApp System of Grapes Juice | 61 |
| 5.5 | Sequence Diagram of the grapes fruit juice supply chain management DApp system | 61 |
| 5.6 | User Interface of Grapes Fruit Supply chain DApp System - Register New User with Order Details | 66 |

LIST OF FIGURES

| | | |
|------|--|----|
| 5.7 | User Interface of Grapes Fruit Supply chain DApp System – Farm and Product Details | 66 |
| 6.1 | Identified Challenges in Agile SDLC of DApps | 70 |
| 6.2 | In practice Agile Methodologies for the Development of DApps and Software . . . | 71 |
| 6.3 | Results of how much essential prioritization of requirements in SDLC of DApps . | 72 |
| 6.4 | Graphical representation of survey results to show the effectiveness of prioritization of requirements for efficient agile development of DApps | 72 |
| 6.5 | Results to validate the effectiveness of defining and specifying goals of the project using SMART analysis | 73 |
| 6.6 | Analytical graph of survey responses to verify the effectiveness of CRC cards in the comprehension of UML diagrams in the agile development of DApps | 74 |
| 6.7 | Analytical graph to validate the influence of pair programming in code quality of agile development of DApps. | 75 |
| 6.8 | Analytical graph to verify that ATDD enhances the validation and acceptance in agile development of DApps. | 76 |
| 6.9 | Graphical representation of responses validate that simple design leads to efficient design and requirements elicitation. | 77 |
| 6.10 | Identified and validated through survey crucial agile practices for quality-oriented development of DApps. | 78 |

List of Abbreviations

SMART — Smart Measurable Achievable Realistic Traceable/Time bound.

BOSE — Blockchain Oriented Software Engineering

BBS — Blockchain Based Software

DApps — Decentralized Applications

ATDD — Acceptance Test Driven Development

TDD — Test Driven Development

CRC — Class Responsibility Collaborator

DLT — Distributed Ledger Technology

SCs — Smart Contracts

XP — Extreme Programming

UML — Unified Modeling Language

IPFS — Inter Planetary File System

SDLC — Software Development Life Cycle

ABCDE — Agile Blockchain DApp Engineering

DeFi — Decentralized Finance

AI — Artificial Intelligence

EVM — Ethereum Virtual Machine

UI — User Interface

P2P — Peer to Peer

ASD — Agile Software Development

Introduction

1.1 Overview

The Fourth Industrial Revolution (4IR) introduced modern and state-of-the-art technologies i.e., Artificial Intelligence (AI), Blockchain, Cloud Computing, Big Data and Analytics, Augmented Reality (AR). Blockchain is one of the pillars of 4IR. It is called the new internet i.e., Web 3.0. It has a key role in the digitalization of the global economy. It is an old technology, however, after the introduction of Bitcoin in 2008, it is considered a suitable environment for decentralized computing [1]. Now blockchain is the primary technology for most of the cryptocurrency platforms i.e., Bitcoin, Ethereum, and Litecoin, these platforms are commonly in use for transactional purposes [2]. Blockchain is based on the peer-to-peer network, which is transparent, immutable, and auditable. Blockchain is an append-only “Distributed Ledger Technology” (DLT). All the processes carried out in it are through using proper consensus mechanisms, though not all the DLTs are based on consensus mechanisms [4].

The introduction of Bitcoin initiated a new domain for the cryptocurrency market and acquired the attention of research and development enthusiasts to introduce the Ethereum blockchain. Ethereum blockchain reinstated the concept of “Smart Contracts” (SCs), which are computer programs, where an event occurs when some specific conditions meet, related to that event [1][3]. SCs of Ethereum and other cryptocurrency platforms use blockchain technology to perform transactional events. As blockchain is a decentralized network of nodes, there is no need for any central trust authority for transactions using blockchain-based cryptocurrency platforms using SCs [4][3].

Blockchain is a core technology of “Decentralized Applications” (DApps) as well, which is a trending area of research in software engineering. DApps are significantly in use for transactional purposes, using blockchain-based SCs. DApps is a complete software system typically based on blockchain. Blockchain-based SCs are used to manage transactional and other information of DApps. Typically, DApps are not life-critical applications, however, they can be used as mission-critical apps [4].

Several platforms to develop SCs in the market are used in DApps, Ethereum is presently the most in-use platform to develop SCs on the public blockchain. It is difficult to manage the data of DApps based on permissioned (private) blockchains, however, Ethereum is also suitable for DApps running on permissioned blockchains [5]. All these technologies i.e., blockchain, Ethereum, and SCs are the building blocks of the architecture of DApps as shown in Figure 1.1 [4]. In Figure 1.1, a typical architecture is illustrated, where on the bottom left is the inner structure of the Ethereum blockchain, where a node is composed of storage, “Ethereum Virtual Machine” (EVM), and SCs byte code. DApps have a “User Interface” (UI) as shown in Figure 1.1, an end user can interact with DApps through the UI of DApps. It is designed on Web3.js and uses an “Operating System” (OS) based web browser. A third part of DApps architecture is the backend server. The backend server stores and manages data, because blockchain nodes have storage capacity only for SCs bytecode, this server performs business computations. Hashes of blockchain-based SCs store large files of a specific size that can affect the performance of blockchain-based SCs and this data is secured with the hash facility at the “InterPlanetary File System” (IPFS) [7][4].

DApps architecture, in Figure 1.1 illustrates, has three main components i.e., blockchain platform (Ethereum), DApps UI, and DApps backend server, The blockchain part is an addition if it is compared with traditional software (applications) architecture.

Subsequently, the development process of DApps needs simple, responding to change approaches that can provide the facility to handle the complex structure of blockchain-based SCs with simplicity and manage the change in requirements and design. The most effective and frequently adopted approaches for such systems are Agile-based approaches. Responding to change is always a priority and the simplicity of these approaches can facilitate the development of blockchain-based SCs along with the other two architectural components of DApps [4][6].

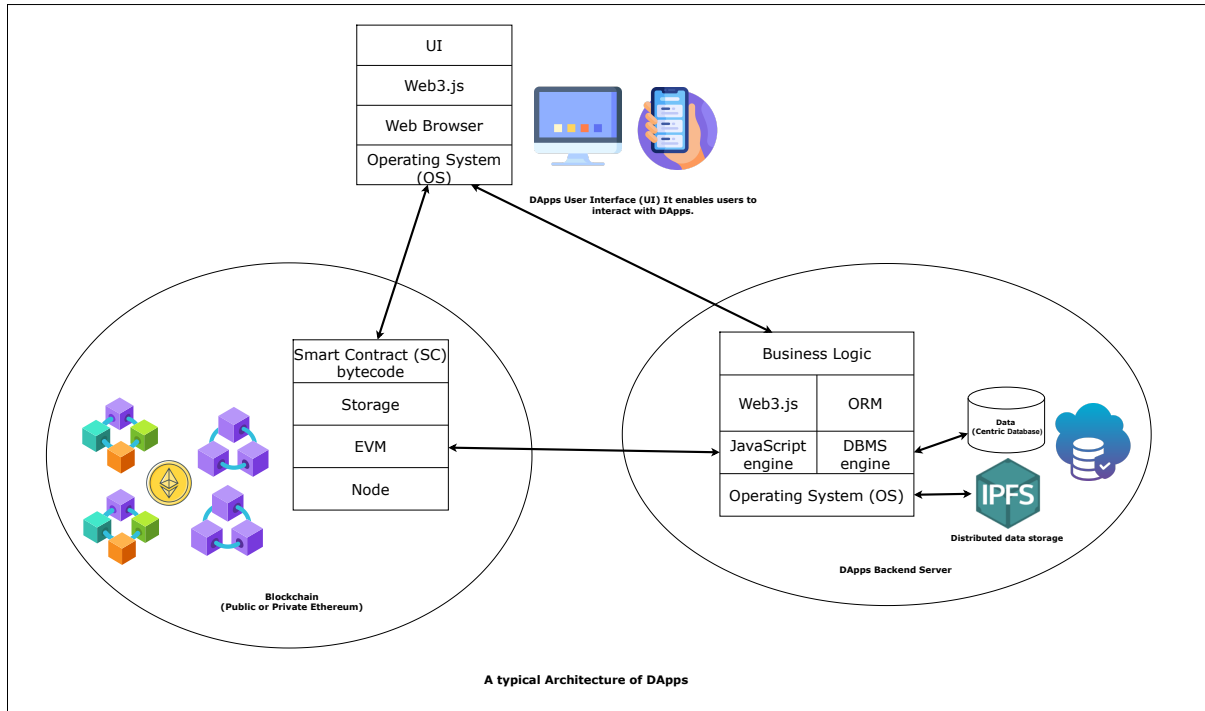


Figure 1.1: A typical architecture of DApps. Ethereum blockchain on the left, DApps UI on the bottom right, and Backend server (centric server or distributed file system i.e., IPFS) on top right [4].

Different quality attributes of the development of DApps, i.e., simplicity of development, efficiency in development, agile testing, and quality-oriented development; such quality attributes are important for the productive development of software. Agile-based approaches can be adopted to achieve these quality attributes as follows:

- Simplicity of Development** Agile-based approaches provide simplicity in design and development. Extreme Programming (XP) is one of the agile methodologies where designs are simple and small, and development is carried out with high-priority features. XP follows a simple approach where all the high-priority tasks are designed and performed at first. This simplicity brings down the need for documentation, and developers focus on high-priority features. In the Agile Manifesto, the agile-based approaches focus on simplicity stating that “Simplicity, the art of maximizing the amount of work not done, is essential”. Agile-based approaches are a proven set of methods for promoting simplicity [8][9].

- **Efficiency in Development** Efficiency in development i.e., resource consumption, is considered a real challenge in the large-scale and complex development of DApps. Considerable practices to mitigate development challenges in the development process are as follows:
 - Adapt the agile-based requirements engineering practices i.e., XP practices can be tailored for large scale and complex projects. In XP, designing can be upfront and customer appearance can be surrogate [10].
 - Use of appropriate technology following the requirements of projects i.e., for large-scale and mission-critical systems IPFS are recommended as database or file storage systems.
- **Agile Testing** Testing is an important phase of every software development process, it does not matter if it is a traditional development approach i.e., Waterfall, or under the umbrella of modern agile methodologies i.e., Scrum, XP, Kanban. A verified and validated product delivery and deployment is the foremost task of every software development approach. However, agile-based approaches emphasize quick and on-demand delivery and deployment of products. This is only possible with continuous customer involvement in the development and testing process. For quality-oriented and efficient products, testing is mandatory. XP, an agile methodology, proposes a practice of Test Driven Development (TDD), where software testing is performed based on prioritized user stories before moving to development and acceptance testing [11][12][13].
- **Quality** After practicing all the above agile-based approaches for simplicity, efficiency, and agile testing, a quality-oriented, efficient, and high-performance product is delivered. Agile-based approaches are adaptable, they can be adjusted as per the requirements of the projects, if a project is mission-critical and high risk then we can perform scalability prioritizing, by prioritizing the user stories with efficiency [14]. Scalability can be achieved by using appropriate technology i.e., IPFS with blockchain [7]. Agile-based approaches also emphasize the simplicity of the design and development phases. “Simplicity, the art of maximizing the amount of work not done, is essential”, a principle from the Agile manifesto [8][15]. Testing is an integral part of the software development process, whereas agile testing follows the agile-based continuous, incremental, and iterative approaches to verify a product that is in development [16]. Unit tests are performed to test a developing system at granularity [17]. Agile-based approaches deal with change throughout the process, so

agile testing at every change in requirements i.e., user stories, provides quality-oriented and efficient products [18]. From small to medium and then large-scale projects agile testing practices are different and adaptable [19]. Simplicity, efficiency, and agile testing play an important role in the development of DApps. These quality attributes are performance metrics to measure the development of DApps.

1.2 Motivation

In the development lifecycle of DApps systems efficient and quality-oriented approaches are deficient. BOSE can mitigate the challenges for efficient and quality-oriented development of blockchain-based systems. Agile-based approaches in BOSE are suitable for the development of BBS (Blockchain Based Software) or DApps because in DApps the requirements are not completely understood and there is always a possibility of change in the requirements throughout the development lifecycle [4]. Change is constant and inevitable, a concept that is frequently practiced in the software engineering community, so whenever a new framework or concept is devised for the development of modern software systems this concept of change is always under consideration. The proposed framework for efficient and quality-oriented development of blockchain-enabled smart contracts-based DApps considers the software engineering concept of change. A hybrid framework based on XP and Scrum methodologies of agile software engineering is proposed, which adopts practices like simplicity, TDD, pair-programming, release planning, and prioritizing the user stories with iterative and incremental development for efficient and quality-oriented DApps. For simplicity in the design phase, we found CRC modeling a useful practice that can lead to a mature Unified Modeling Language (UML) diagram from simple to detailed architecture. We found agile testing i.e., acceptance testing with collaboration with stakeholders and TDD, an effective practice for quality, efficiency, and rapid development of a product.

1.3 Problem Statement

DApp systems are complex as compared to traditional software systems. Blockchain technology is the reason for its complexity because of its immutable nature and working with sensitive data i.e., transactions, it is a critical task to manage such systems. The architecture and nature of blockchain disturb resource management which leads to disruption in efficiency and quality

of the development lifecycle of DApps. Other problems also exist in the development lifecycle of DApps i.e., Changing requirements from stakeholders. A well-organized and standardized framework of the software development life cycle is required to deal with the immutable and complex nature of blockchain technology in the DApps system. Software engineering owns such approaches and practices that can mitigate the risks associated with the efficient and quality-oriented development of DApps.

BOSE is a concept that promotes software engineering practices for the development of blockchain-enabled smart contracts based on DApps. Agile methodologies i.e., Scrum, and XP, are suitable for the continuously changing and immutable nature of blockchain-enabled smart contracts-based DApps.

1.4 Research Objectives

The following research goals are intended to be accomplished.

- **RO1:** To evaluate the current state of DApps development and identify key challenges and inefficiencies in the development process.
- **RO2:** To analyze the principles and best agile practices and identify how they can be applied to DApps development to improve efficiency and quality.
- **RO3:** To propose an agile-based framework for DApps development that incorporates efficient and quality-oriented products.
- **RO4:** To conduct a case study to examine the proposed agile-based process for the development of DApps.
- **RO5:** To identify key factors from the case study that affect the performance or outcome of the proposed agile-based process for the development of DApps.

1.5 Contribution of Research

This research thesis proposes a framework for the efficient and quality-oriented development of DApps. The proposed framework is based on agile approaches where development is carried out in continuous iterations and increments. Efficient and quality-oriented development are two core factors that are considered as priorities for the development of DApps. Other factors that

are associated with the development of DApps for efficiency and quality in development are illustrated in Figure 1.2:

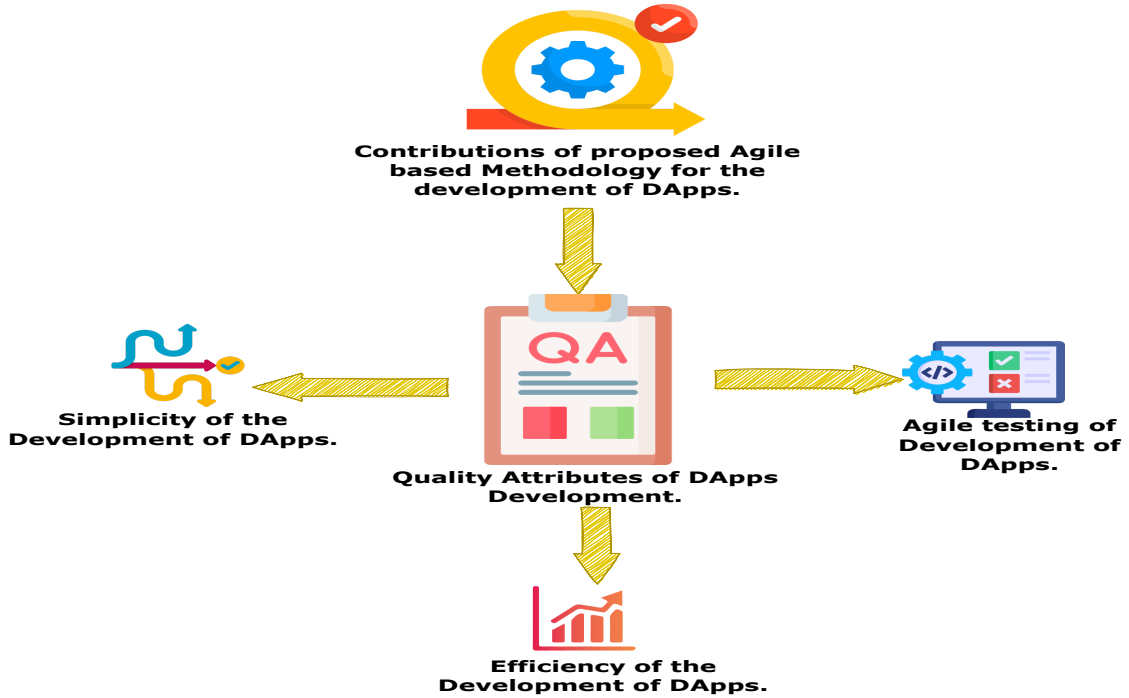


Figure 1.2: Contributions of the proposed framework for the development of DApps

- **Simplicity of the Development of DApps** - The architecture of DApps systems is complex as compared to traditional software systems and the immutable nature of blockchain technology increases this complexity. In the proposed framework simple and specified agile-based approaches are adopted for the development of complex DApps systems.
- **Efficiency of the Development of DApps** - The immutable nature of blockchain technology in DApps reduces efficient development and resource management. In the proposed framework appropriate and user-centric agile-based approaches are adopted for efficient development of DApps systems.
- **Agile Testing of the Development of DApps** - For efficient and quality-oriented development of DApps an organized and efficient way of testing and agile testing is the most suitable approach for this purpose.

1.6 Thesis Outline

The research work has been structured and distributed as follows:

- **Chapter 1 - Introduction** An overview of the whole research work is explained, and the motivation to work on this topic is briefly described, followed by the problem statement and research objectives of this study. Furthermore, the contributions of this research work are highlighted.
- **Chapter 2 - Literature Review** Study the related literature and organize it in a structured form to understand the need of the proposed framework, from start to end and from basic to advanced every aspect of literature is strived to cover.
- **Chapter 3 - Proposed Methodology** The Rationale and motivation for the research work are explained, followed by the proposed framework illustrated with proper description. Key attributes of the proposed framework are also highlighted to strengthen the point of rationale and motivation.
- **Chapter 4 - Validation of Proposed Framework** A questionnaire survey is designed to get the comments and reviews of industry experts on the proposed framework. A practical validation is also performed to validate the different phases of the proposed framework.
- **Chapter 5 - Results and Analysis** Discussion and analysis of the results of the questionnaire-based survey and practical validation of the proposed framework.
- **Chapter 6 - Conclusion and Future Work** A brief conclusion is drawn with some research gaps for future research work.

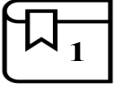



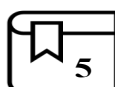
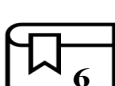
| | |
|---|---|
|  | Introduction |
|  | Literature Review |
|  | Proposed Methodology |
|  | Validation of Proposed Framework |
|  | Results and Analysis |
|  | Conclusion and Future Work |

Figure 1.3: Thesis Chapter wise Outline

CHAPTER 2

Literature Review

2.1 Introduction

Chapter 2 provides the related research and development work that has already been carried out on Agile-based approaches for efficient and quality-oriented development of DApps. In this chapter, firstly a brief description of agile software development lifecycle with different features of agile software development is described. After that, different concepts, and terminologies i.e., Smart contracts, DApps, BOSE, Pair Programming, and others, are discussed one by one with the help of relevant literature. Moreover, a brief analysis of how quality concerns like simplicity, efficiency in development, and testing can contribute to the quality-oriented development of DApps. All these topics are covered with the help of appropriate references from the research and development that has already been carried out.

2.2 Agile based Software Engineering

“Agile Movement” was first introduced, from a research phase to industry and development, after the publication of the Agile Software Development Manifesto in 2001, by Beck et al. (2001). The agile manifesto is established on the core belief that “We are uncovering better ways to develop software by doing it and helping others do it” [20][21]. Individuals and interactions, Customer collaboration, working software, and responding to change are valued over Process and tools, contract negotiation, comprehensive documentation, and following a plan respectively. These are core beliefs that are always considered while adopting agile-based approaches for software development [20][21]. The recognition of people as a core contributor to project success,

with an intense focus on effectiveness and flexibility, is the principal practice of agile-based software development practices. Cockburn et al. (2002a), defined the use of light and sufficient rules of project behavior with the use of human-centric communication-oriented rules which are basic units of agile software development practices [21]. Typically, there are six core phases of development in the agile software development lifecycle. It is an iterative and incremental development process, where at each release phase, the system is updated with a new increment as a typical agile software development is shown in Figure 2.1 [22].

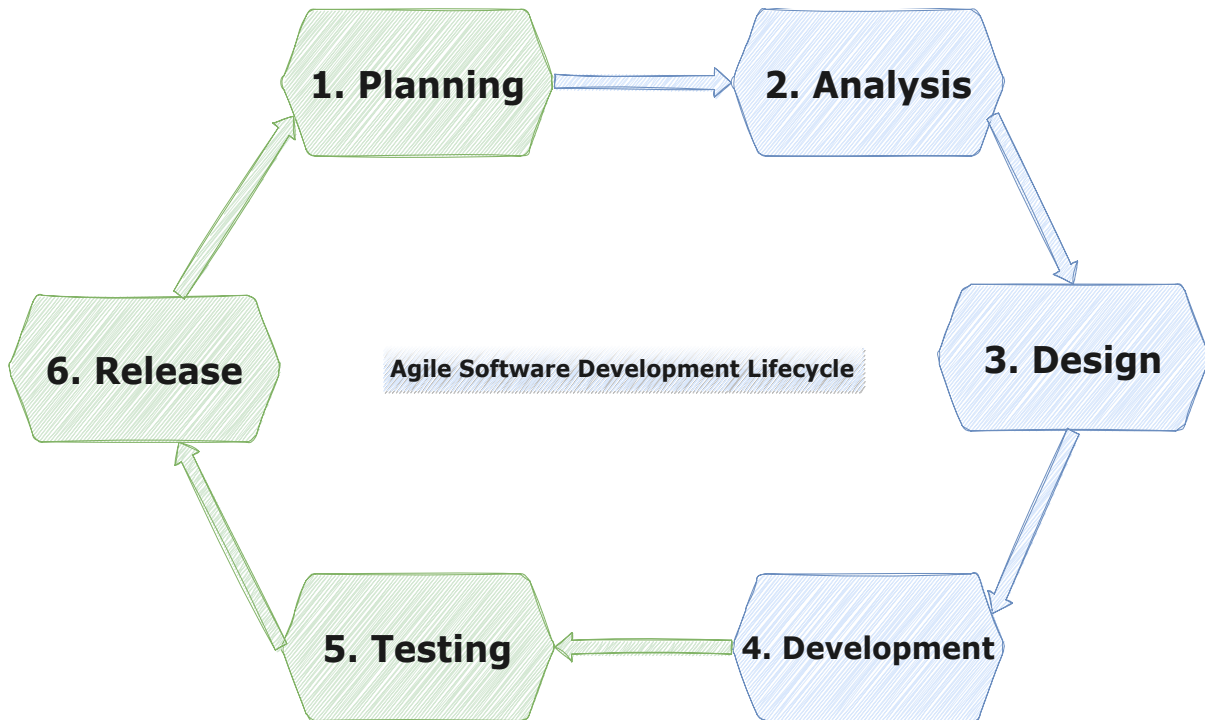


Figure 2.1: Agile Software Development Lifecycle [22]

Several agile development approaches lie under the umbrella of agile software development manifesto i.e., Scrum, XP, Kanban, Lean development, Crystal, etc. Key features of agile-based software engineering practices are as follows [20] [21]:

- Fast development with quality, with rapid feedback, to win the confidence of all the stakeholders. Maximum delivery time is 3 weeks in agile-based practices.
- Always try to avoid complexity, and go to simplicity, this will bring ease in change and fewer requirements will be in the change process at a time.

- Simplicity in the design process, improve the design quality with time to make the next user story cost-effective.
- Agile testing after regular intervals will help early and cost-effective detection of defects, without any extra effort at the maintenance or deployment phase.
- Always prefer the people over process and tools, involve human efforts in the development process as much as possible.
- Agile practices are preferred for no or less documentation during development, ready to use software, is preferable as compared to lengthy documentation and software deployment at once.
- The most critical part of agile-based software development approaches is the communication among stakeholders.

In this research thesis, we will propose a customized hybrid approach based on Scrum and Extreme Programming (XP). These two agile approaches are the most adopted methodologies, according to VersionOne's 2009 Agile Methodology Survey, Scrum is adopted by 50% of the software development, whereas XP is 24%. Both can be applied as customized hybrid software development processes [20].

2.3 Blockchain and DLT

DLT is based on several peer-to-peer (P2P) technologies that are enabled by the internet. Transactions and transfers through the internet are considered elusive because everything on the internet is at risk and the same in the case of transactions and transfers i.e., document, and media files transfer [24]. In 2008, a novel approach was proposed in a P2P manner, for funds transfer, in a research paper written by an as-yet unidentified person, who used the pseudonym Satoshi Nakamoto. The approach was based on blockchain technology, which is a particular type of immutable data structure. Cryptographic and algorithmic methods are employed in it to record and synchronize data across a network [24]. Blockchain and DLT are considered similar technologies, both technologies are famous in different projects of industry. Note that, not all distributed ledgers are based on blockchain technology [24].

Blockchain stores identical information across different physical addresses i.e., nodes, peers, or computers. Immutability, data integrity, transparency, and tokenization are the unique features

of blockchain technology that make it secure, traceable, and trustworthy [25]. Blockchain follows a well-structured process of functioning, from a transaction request to the completion and execution of the transaction as follows in Figure 2.2 [25]:

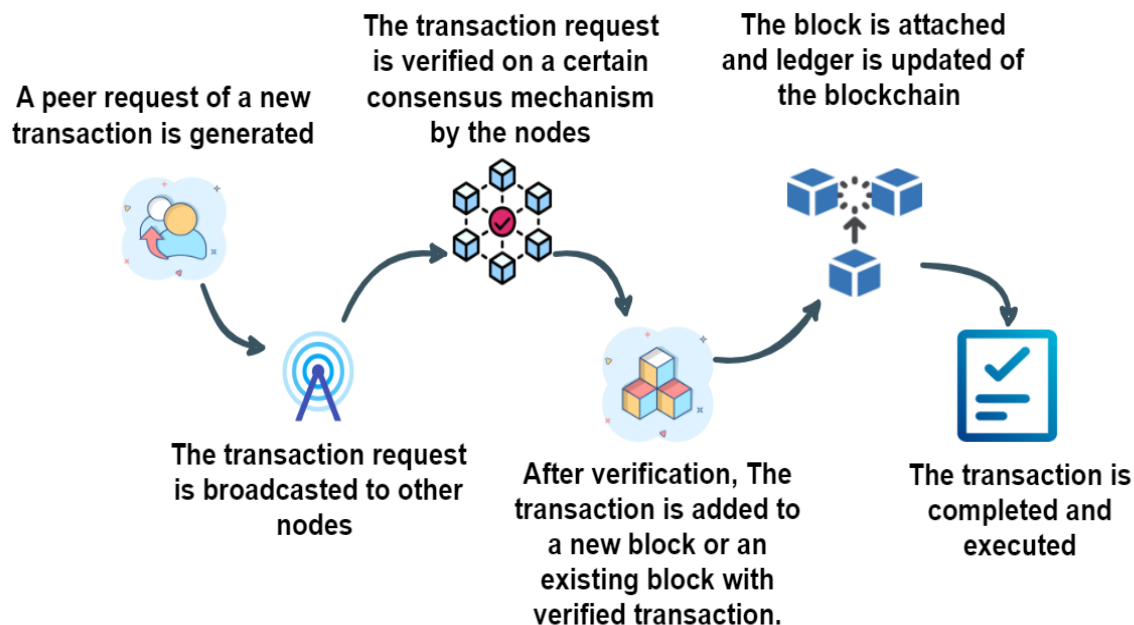


Figure 2.2: A process of functioning from a new transaction request to the completion and execution of the transaction [25]

Michael Nofer et al. (2017) discuss blockchain and its potential in finance and other projects of industry. Applications of blockchain technology in finance and other sectors of industry are elaborated in the research, however, the finance industry is the primary application of it [23]. The Michael Nofer et al. (2017) research provides a structured comparison of financial and non-financial applications of blockchain, as follows in Table 2.1 [23]:

| Business Domain | Area of Application of Blockchain | Examples |
|---|---|--|
| <i>Applications of Financial Domain</i> | <i>Cryptocurrencies</i> | Bitcoin, Litecoin, Ripple, Monero |
| | <i>Insurance, Security, Trading, and settlement</i> | Everledger, NASDAQ private equity, Medici, Coin setter |
| <i>Applications of Non-Financial Domain</i> | <i>Notary Public</i> | Stampery, Viacoin, Ascribe |
| | <i>Music Industry</i> | Imogen heap |
| | <i>Decentralized storage proof of documents</i> | POEX.io |
| | <i>Decentralized Storage</i> | Storj |
| | <i>Decentralized Internet of Things (IoT)</i> | Filament ADEPT (Developed by IBM and Samsung) |
| | <i>Anti-counterfeit solutions</i> | Blockverify |
| | <i>Websites and Apps</i> | Namecoin |

Table 2.1: A structured comparison of financial and non-financial applications of blockchain technology [23]

2.4 Blockchain based Smart Contracts

Blockchain is a revolutionary technology, the key aspect of this technology is smart contracts. Smart contracts are computer programs, and the correctness of their execution is guaranteed by the security of the blockchain technology to make them tamper-proof. Smart contracts are secure, decentralized, economically beneficial, and legal, however, all these properties are not straightforward to achieve [26]. Capocasale V. et al. (2022) discuss misconceptions and also provide guidelines. The key guidelines on misconceptions about smart contracts are discussed in [26] as follows:

- Smart contracts use two types of data internal or ledger data and external or transaction data, The first one is reliable and the second one must be verified, so smart contracts move the system from one state to the other and are state transition functions.
- Must be a possibility to verify output generated by smart contracts at every node in the future.
- The output generated by smart contracts must be the same on all the nodes executing it which makes smart contracts deterministic.
- Smart contracts must be deterministic and equivalence classes, the same input state and same input symbol must generate the same output state.
- No need to store smart contracts on-chain. However, it can vary in some contexts.
- Smart contracts are not immutable, however updating smart contracts is not an easy task.
- Smart contracts are not just legal contracts, but they have a good range of possible applications.
- Independent coding, testing, and execution of smart contracts should be preferred.

The concept of smart contracts gained popularity after the invention of blockchain technology because this technology makes smart contracts immutable or difficult to modify. Both technologies can exist independently, however, they are considered a combination of technologies as both can broaden the scope of one another [25].

Smart contracts have a complete lifecycle, from creation to completion there are four major phases: Creation, Deployment, Execution, and Completion phase. Every phase has certain steps to complete [27]. Zheng, Z. et al. (2020) illustrate the four phases of the smart contracts life cycle with different challenges and proposed advances to mitigate each of the challenges in every lifecycle phase, the complete lifecycle of smart contracts, with four phases, is illustrated in Figure 2.3:

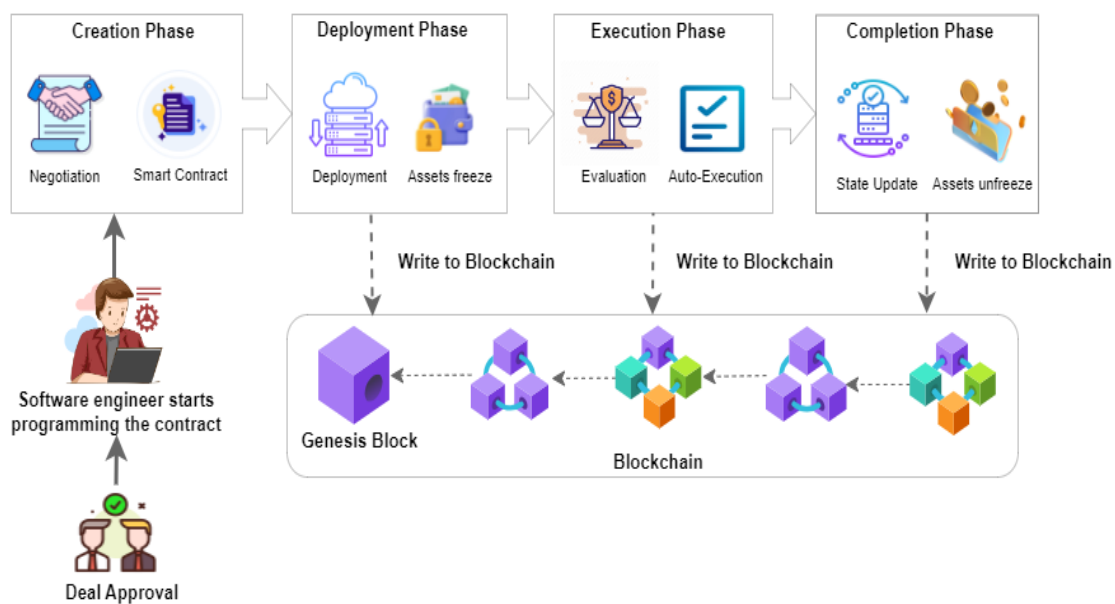


Figure 2.3: Phases of smart contracts life cycle, from creation to completion [27]

Zheng, Z. et al. (2020) also discuss the applications of smart contracts with the help of use cases for better understanding. These applications are categorized into six types; Internet of Things (IoT), Security Distributed Systems, Finance, Data Provenance, Sharing Economy, and Public Sector, details of these types are described in Table 2.2, with benefits and use cases [27]:

| Application Categories | Benefits of Smart Contracts | Use Cases of Smart Contracts |
|---|--|--|
| 1. Internet of Things | <ul style="list-style-type: none"> • Reduce maintenance cost of the central server. • Automation of P2P business trading • Cost reduction for trusted third parties | <ol style="list-style-type: none"> 1) Auto-update of IoT device firmware 2) Speed up the supply chain |
| 2. Security of Distributed Systems | <ul style="list-style-type: none"> • Quick and reliable listing of share attack • Verification of cloud service providers • Avoidance of brokers | <ol style="list-style-type: none"> 1) Mitigation of DDoS in computer networks 2) Cloud computing |
| 3. Finance | <ul style="list-style-type: none"> • Reduce financial risks. • Administration and service cost reduction. • Improvement in efficiency of financial services. | <ol style="list-style-type: none"> 1) Investment banking and Capital Markets 2) Retail and commercial banking 3) Insurance |
| 4. Data Provenance | <ul style="list-style-type: none"> • Capture malicious data falsification. • Improve data reliability. • Protection of privacy | <ol style="list-style-type: none"> 1) Scientific research 2) Public health 3) Provenance of cloud data |
| 5. Sharing Economy | <ul style="list-style-type: none"> • Customer cost reduction • Trusted third parties' cost reduction. • Protection of privacy | <ol style="list-style-type: none"> 1) Item sharing 2) P2P automatic payment systems 3) Platforms of currency exchange |
| 6. Public Sector | <ul style="list-style-type: none"> • Protection from data fraudulence • Transparency of data of public information • Protection of privacy. | <ol style="list-style-type: none"> 1) E-voting systems 2) Personal reputation systems 3) Platforms of smart property exchange |

Table 2.2: Applications of smart contracts with their benefits and use cases [27]

2.5 Smart Contracts based DApps

“Decentralized Applications” or “DApps” use smart contracts as backend and standard web technologies i.e., JavaScript and React JS, for frontend or web user interface. Smart contracts are not suitable to store and process large amounts of data, so off-chain data storage services i.e., IPFS and Swarm, are used to store bulky data. In DApps, all possible aspects i.e., backend software or application logic, frontend software, data storage, message communication, and name resolution are mostly or entirely decentralized [28]. Resilience with no downtime, Transparency, and Censorship resistance are unique features that distinguish DApps from traditional web applications. A comparison of traditional web applications and DApps is briefly described in Table 2.3 [7]:

| Features for Comparison | Traditional web Apps | DApps |
|--------------------------------|---|---|
| 1. Architecture | Client-server architecture | Blockchain-based P2P architecture using smart contracts |
| 2. Authority | Central authority for roles and permissions setup | Decentralization with no central authority |
| 3. Privacy | No or low-level privacy | Less privacy issues |
| 4. Transparency | Lack of transparency | Transparency guaranteed |
| 5. Down Time | Mostly face downtime | Resilience with no downtime |
| 6. Secure | Security issues | More secure and autonomous |
| 7. Immutability | No immutability | Immutable |

Table 2.3: Comparison of traditional web applications and DApps [7]

Different architectures of smart contracts blockchain or smart contracts based DApps are present, i.e., Native Client as a DApp, Smart Contract as a DApp, Web and Contract as a DApp, and Fully decentralized DApp. A typical architecture of DApps is illustrated in Figure 1.1. Every architecture is used for a specific type of DApps, and different types of DApps are present in the industry, where unique features of DApps are in practice. These types of DApps are described with their practical advantages in Table 2.4 [29]:

| Types of DApps | Practical Advantages of DApps |
|--|---|
| <i>1.Financial Sector i.e., DeFi</i> | <ul style="list-style-type: none"> • Improve efficiency • Time cost reduction • Automatic execution of transactions |
| <i>2.Online Gaming Platforms i.e., Game-Fi</i> | <ul style="list-style-type: none"> • Permanent game assets • Improve asset mobility • Uniqueness of roles is ensured |
| <i>3.Provence and Data Storage i.e., Non-Fungible Tokens (NFT)</i> | <ul style="list-style-type: none"> • Permanent and immutable storage • Privacy is preserved • Reliability is improved |
| <i>4.Privacy Protection</i> | <ul style="list-style-type: none"> • Privacy is preserved • Improvement in user ownership of data |
| <i>5.Sharing Platforms i.e., P2P</i> | <ul style="list-style-type: none"> • Improvement in platform efficiency • Sharing without trusting a third party • Promotion of resource sharing |
| <i>6.Gambling and Prediction Platforms</i> | <ul style="list-style-type: none"> • No need for trusted third-party • Low costs with less fees • Automatic execution of transactions |

Table 2.4: Practical advantages of different types of DApps [29]

2.6 Quality Attributes of Development of DApps

Development of DApps is different and complex, as compared to traditional web applications development, because of blockchain-based smart contracts. Quality aspects must be considered in the development process of DApps. Simplicity, Testing, efficiency, and quality-oriented development are key attributes that must be considered in the development of DApps. Such attributes with related research are discussed here.

2.6.1 Simplicity in Development of DApps

The emerging challenges and opportunities in software development have driven the project management approaches to be flexible. Agile software development methodologies promote

simplicity through collaboration and flexibility. Agile values are based on some principles that are described in the Agile Manifesto. Simplicity is one of the principles that emphasizes quick and continuous feedback and collaboration with customers [15]. XP is an agile method of software development, XP practices focus on five XP values i.e., simplicity, respect, communication, courage, and feedback. XP practices are kept simple, from planning to design and implementation simple solutions are preferred [30].

Santos, W.B. (2018) illustrates the factors in Agile Software Development (ASD) that lead to simplicity in the software development process. The factors are found after conducting primary studies, as follows [31]:

- **Reduction of Time** – The Development process is divided into different phases, depending on time and cost reduction.
- **Development of the Team** – Teams are developed to work on different phases of the development process.
- **Simple Design** – Design is always simple and unambiguous for every phase of development.
- **Innovation and Creativity** – Simple Designing produces creative and innovative development processes.
- **Lightweight Management Tools** - Simple designing, innovation, and creativity are achieved by using lightweight management tools.
- **Less or Necessary Documentation** – Agile methodologies promote less documentation and focus on working products with less or necessary documentation.
- **Adaptation of Methodology** – Methodologies that lie under the umbrella of Agile software development are adapted.
- **Model Driven Development** – Different modules are designed to develop software.
- **Refactoring** – Iterative and incremental development is achieved by refactoring at every iteration or increment.
- **Reuse of Code** – Code is reused by different modules and iterations.
- **Test Driven Development** – Test cases are generated based on user stories and before implementation.

Simplicity is considered a best practice in defensive programming because complexity creates security issues. Other best practices of defensive programming for the secure development of smart contract-based DApps are emphasized as follows [28]:

- **Simplicity** – Complex code and practices can create security issues in smart contracts-based DApps development.
- **Code Reusability** – Reusability is always encouraged to adopt traditional and agile methodologies [27] [30].
- **Quality of Code** – Rigorous quality-oriented software development methodologies should be in practice to develop smart contracts based on DApps.
- **Auditability or Readability of Code** – Code should be simple, clear, and easy to understand. This will lead to an auditable code of smart contracts.
- **Test Coverage** – Every module of smart contracts-based DApps should be tested, agile testing would be an ideal testing methodology for this purpose.

Santos, W.B (2018) also discussed the benefits of the factors that lead to simplicity in ASD. All the previously discussed factors are illustrated in Table 2.5, with their benefits of simplicity in ASD [31].

| Sr. No. | Benefits of simplicity in ASD | Factors lead to simplicity in ASD |
|---------|---|---|
| 1. | <i>Avoid over-development</i> | Development of the team, Adaptation of methodology, Refactoring. |
| 2. | <i>Minimize the development time</i> | Reduction of time, Innovation and Creativity, Lightweight management tools, Less or necessary documentation, and Adaptation of methodology. |
| 3. | <i>Maintenance</i> | Development of the team, Adaptation of methodology, Refactoring, Simple design, Reuse of code. |
| 4. | <i>Quality of Code</i> | Development of the team, Adaptation of methodology, Refactoring, Reduction of time, Simple design, Lightweight management tools, Less or necessary documentation, Model-driven development, Test-driven development, and Reuse of code. |
| 5. | <i>Lightweight Process</i> | Reduction of time, Simple design, Lightweight management tools, Less or necessary documentation, Adaptation of methodology, Model driven development, Test-driven development, Innovation and Creativity. |
| 6. | <i>Product with value</i> | Adaptation of Methodology, Development of the team, Refactoring, Simple design, Innovation and Creativity, Lightweight management tools, and Less or necessary documentation. |
| 7. | <i>Simplification of the selection of Methodology</i> | Adaptation of methodology, Simple design, Lightweight management tools, Less or necessary documentation, Innovation and Creativity. |
| 8. | <i>Reusability and Simplicity of the Design</i> | Simple design, Adaptation of methodology, Refactoring, Lightweight management tools, Fewer or necessary documents, Innovation, and Creativity. |

Table 2.5: Factors lead to simplicity with their benefits in ASD [31]

2.6.2 Agile Testing in Development of DApps

When a testing or quality assurance team performs different agile-based approaches i.e., TDD, to test the product and maintain quality in a software development process, such activities are called “Agile Testing”. Agile testing methodology is different from traditional development and testing methodologies. In traditional development methodologies i.e., waterfall method, testing is carried out at the end, right before release. Agile is an iterative and incremental approach, so each increment of coding is tested as it is finished [32] [34]. Agile testers follow core principles which include [32]:

- Continuous feedback.
- Valuable output to customers.
- Encourage face-to-face communication.
- Can perform refactoring.
- Keep the testing procedure simple.
- Practice for continuous improvement.
- Ready to respond to change.
- Maintain yourself and be self-organized.
- Focus on people, not process.
- Enjoy your work.

Puleio, M (2006) discusses challenges in several areas i.e., communication, estimation, and automation, that a testing team faces during the agile testing process. In agile testing, all agile principles are followed so these challenges are also related to agile methodologies [33]. Mohanty, H. et al. (2017) discuss trends in software testing with the advantages and disadvantages of agile testing. A summary with a comparison of these advantages and disadvantages of agile testing is in Table 2.6 as follows [34]:

| Sr. No. | Advantages of agile testing | Disadvantages of agile testing |
|---------|--|--|
| 1. | Knowledge transfer between developers, testers, and junior team members. | Increase in intra-team communication over inter-team communication. |
| 2. | Less documentation and excellent communication with small teams. | Personality clashes of the team members. |
| 3. | Requirements volatility is reduced because of continuous customer feedback. | Challenges in scaling agile methods to large groups or organizations. |
| 4. | Small manageable tasks with continuous integration for process control and transparency. | Generation and maintenance of a priority list of all the projects and small stories and teams. |
| 5. | Early detection of bugs. | Inside dependencies management of implementation. |
| 6. | Improvement in quality of work life. | Less time spent leads to poor design, architecture, and rework. |

Table 2.6: Advantages and disadvantages of Agile testing [34]

Testing of smart contracts is a costly phase in the development of DApps because of the immutability concept, if a bug is found after deployment, then there has to be developed a new smart contract. Ethereum transaction requires a payment of a fee, depending on the size of the smart contract, whenever a deployment of smart contracts is performed [35]. The factor of payment of fees for Ethereum transactions increases the importance of the test phase before the deployment of smart contracts. So, it should be managed through the adoption of the best approaches i.e., agile methodologies and specific tools of testing [35]. Agile testing is based on iterative and incremental approaches and is suited to testing quickly with quality in small to medium self-organizing teams, working together with continuous communication and customer collaboration [35].

2.6.3 Efficient Development of DApps

In any software development methodology, requirement engineering is a critical phase. Several requirements engineering techniques are in practice for efficient requirements gathering in an agile software development process. Traditional requirements engineering techniques are well adapted to flexible agile software development methodologies [36]. Malik, M.U. (2013) discusses requirements engineering techniques for agile methodologies of software development such as Direct discussion and interviews, JAD (Joint Application Development Sessions), Collective as well as individual brainstorming, User stories, and use case scenarios as requirements elicitation and Onsite or online surveys questionnaires are some efficient requirements engineering techniques that are well suited for flexible agile methodologies [36]. Malik, M.U. (2013) also discusses six evaluation criteria, to evaluate these requirement engineering techniques, based on cost, time, and resource effectiveness along with final impact, audience reach, and feasibility for agile development methodologies [36].

The main objective of agile software development methodologies is to remove the obstacles to efficient software development by following the principles of agile manifest [37]. Lee, G. et al. (2010) analyze the quantitative and qualitative field data on software development agility and discuss several issues related to the relationship between software development agility, performance, efficiency, agility, and autonomy agility. The research also illustrates some findings relevant to these issues as follows [38]:

- **Research findings for relationships between agility of software development and performance [38]**
 - Requirements change is evaluated by its impact on business, time, cost, scope, and technical difficulty.
 - Highly impactful business changes are prioritized regardless of constraints.
 - Responses should not be interrupted but responding extensively in the early stage can be time and cost-efficient.
- **Research findings for relationships between efficiency and extensiveness of response [38]**
 - Extensiveness of response is in direct relation with lower response efficiency.
 - Extensive responses to changes result in work overload and lack of focus.

- Agility in efficiency and extensiveness of response can be increased by effective management of time and cost.
- **Research findings for autonomy and diversity on agility [38]**
 - Autonomy on agility increases response efficiency whereas diversity on agility is helpful in effective solutions of complex problems.
 - Autonomy on agility limits the responses to change to control the speed of completion of overall project goals whereas diversity on agility provides quick solutions by better translation and understanding of complex changes in requirements.

Liu, Y. (2021) proposes efficient and effective scaling-up solutions for blockchain, where general subcategories in blockchain scaling are offloading of workloads, enabling the interoperability of chain for free token transfers, Accelerate the time-consuming executions, partitioning of networks i.e., Sharding, and Lighten the resource-intensive mechanism, these general subcategories have related solutions to scale up the blockchain-based systems [40].

Decentralized Finance (DeFi) is primarily based on blockchain technology. DeFi has characteristics of blockchain i.e., Transparency, Permissionless, and Trustless because of decentralized and distributed ledger technology, Interconnectivity, Governance is decentralized and establishing self-sovereignty in the system. The purpose of DeFi is to provide all the advantages of blockchain technology in the financial sector [39]. Amler, H. et al. (2021) discuss several major challenges that users and developers face in the development of DApps for the DeFi sector such as Security, Limited Scalability, Oracles, and Geographical or international rules and regulations i.e., GDPR [39]. Pierro, G.A. (2022) describes scalability problems and their solutions along with a trilemma view of blockchain which says three important attributes of blockchain technology i.e., decentralization, security, and scalability cannot perfectly co-exist, a brief overview of this trilemma is illustrated in Table 2.7 as follows [40]:

| Sr. No. | Solutions of Scalability Problems of Blockchain System | Trilemma trade-off of Solutions |
|---------|--|--|
| 1. | Blockchain interval shortening of a blockchain to increase the throughput. | Transactions' throughput increases however it creates security concerns for the whole system. |
| 2. | Increase the block size in blockchain, by maintaining the original block time. | Transfer of Large blocks is not easy because of the limited bandwidth of the intra-blockchain. |
| 3. | Perform the compression of transactions in each block to save the network bandwidth. | Security concerns due to the hash collisions in short hashes. |
| 4. | Sharding increases the transaction process of blockchain per second. | Decentralization is disturbed by the blockchain system. |
| 5. | Use of Proof of Stake (PoS) consensus as an alternative mechanism to avoid the computational overhead produced by Proof of Work (PoW) consensus. | Security and decentralization concerns of blockchain arise. |

Table 2.7: Solutions of Scalability problems and trilemma trade-offs of these solutions in blockchain system [40]

2.6.4 Quality-oriented Sustainable Development of DApps

A smart contract is executed when a consensus of the whole network and multiple other contracts of the same block are executed in series, this results in some overheads i.e., excessive time and poor performance of execution. In Ethereum, the miner collects a fee on every transaction emission as an incentive to include new transactions in blocks. The transaction fee can be calculated as follows [41]:

$$\text{Transaction Fee} = \text{Gas Cost} \times \text{Gas Price}$$

Where:

Gas Cost is the sum of the cost of the gas of each instruction of Ethereum Virtual Machine

(EVM). *Gas Price* is the price defined by the caller of the function of the smart contract.

It is not possible to know what function of the smart contract is being called, only with the cost of the gas. For this purpose, smart contract interfaces are required that are provided in DApps. Blockchain testing and smart contracts testing are two categories where extensive research is being conducted [41]. The execution of smart contracts is sequential in blockchain systems e.g., execution of one contract at a time. Sequential execution would affect the blockchain-based systems by limiting the execution of the smart contracts per second, and with the growth in the number of smart contracts, this can also affect the scalability of blockchain-based systems. Parallel execution of smart contracts is a proposed solution and an alternative in [42].

Scherer, M. (2017) provides comprehensive results on techniques to improve the performance and scalability of blockchain systems or networks i.e., Bitcoin, Ethereum, and Hyperledger [43]. Some effective techniques that are described to improve performance and scalability are as follows [43]:

- Raise the limit of block size cap in Bitcoin using the blockchain forks technique i.e., hard fork or soft fork.
- Segregated Witness, or Seg Wit, in which data from Bitcoin transactions related to signatures is removed to reduce the block size.
- Sharding is a technique in which the global state of accounts/nodes in Ethereum are divided into smaller segments or chunks also known as shard, where each shard possesses its history of transactions.
- Creating channels is a technique adopted in Hyperledger blockchain networks, in which a big blockchain is split into many private blockchains, to make data isolation and confidentiality possible.

All these techniques are practiced for sustainable quality-oriented development of blockchain-based systems. The main aspects that are considered for the sustainable development of blockchain-based systems are social which is about customers and other stakeholders' encouragement, economic factors are about financial aspects and environmental factors are related to efficient and effective use of resources [44].

2.7 Blockchain Oriented Software Engineering (BOSE)

Implementation of blockchain technology in the development of software is defined as Blockchain Based Software (BBS), blockchain is an integral part of such software systems. Blockchain technology in BBS is considered of distinctive importance, the development of BBS can be efficient and quality-oriented by following the software engineering practices. Practice the standard software engineering methodologies and approaches for the development of BBS is defined as Blockchain Oriented Software Engineering (BOSE). From requirement elicitation and analysis to design, development, and deployment of BBS is practiced by adopting the standard software engineering methodologies [45][46]. BOSE leads to improved development processes and techniques by following the standard software engineering approaches.

Blockchain technology is influential with its continuous evaluation and research, in its early days, it was considered the backbone of currency i.e., digital currency, all cryptocurrencies fall into the domain of blockchain (Blockchain 1.0). After the introduction of Ethereum and smart contracts, it has become a new computing paradigm after mainframes, Personal computers, the internet, and social networks. Now it has been adopted in the financial sector for security and verification purposes (Blockchain 2.0). Now other public sectors like health, science, art and culture, literacy, and government are also adopting this computing paradigm (Blockchain 3.0) [47]. Reddivari, S. et al. (2022) provide a generic detail of all the available and future variants of blockchain technology i.e., Distributed Frameworks like public, private or consortium, Consensus algorithms like PoW and PoS, and Transaction Payloads like simple, stateful, or private transactions. These variants are affected by different non-functional requirements, as listed below, that must be in consideration while working on variants of blockchain, under practices of blockchain oriented requirement engineering from BOSE [47].

1. **Decentralized Governance** – A social contract of the predefined and codified process of decision-making on known vetted participants.
2. **Integrity** – Hash and signature-protected linked blocks.
3. **Transparency** – Data and transactional records are open for all nodes of the private blockchain.
4. **Security** – Integrity and transparency combine to provide security from anonymous actors.

5. **Privacy** – Reduce the transparency to make a blockchain system more privacy-oriented by eradicating the flow of information in blocks.
6. **Scalability** – Increase the size of blocks to increase throughput and other attributes of scalability.
7. **Performance** – Manage different attributes i.e., throughput, size, and creation time of blocks for a targeted performance of blockchain systems.
8. **Availability/Reliability** – Resilience against cyber-attacks.
9. **Speed** – Control the creation and transaction speed of blocks.
10. **Customizability/Extensibility** – Adaptability of core protocol or algorithm to a certain situation or requirement.
11. **Testability** - Test and verify code before deployment.
12. **Reversibility** – Reverse the transactional records on blockchain systems, on desire, or because of human or software errors.
13. **Regulatory Compliance** – Follow all the rules and regulations set by regulatory bodies at international and geographical levels.

All the above-discussed non-functional requirements of blockchain variants need proper software requirement engineering approaches for different aspects of blockchain-oriented software engineering life cycle. In blockchain-oriented software requirements, the development aspect is two types of development first one is classic requirement engineering practices, approaches, or frameworks i.e., the waterfall model. The second one is modern software requirement engineering practices i.e., agile development or model-driven development [48]. Infrastructure aspects of blockchain-oriented requirement engineering, consider generic frameworks and architectures for user engagement [48]. Whereas the quality aspect is achieved by requirement validation through requirements traceability and requirements negotiation under user scenarios and guidelines [48]. Integration of software engineering with other technologies is always appreciated, specifically, integration of agile software engineering methodologies with other software engineering practices and approaches is a common practice. The integration is appreciated because of established software engineering concepts present in agile methodologies for usability, requirements engineering, security and safety of critical systems, global software engineering, design, and deployment of

software systems. The same practice is also adopted in the BOSE paradigm, and it is considered the future of agile methodologies [49].

This consideration is because many technological misconceptions need to be clarified under proper management and guidelines [26]. In BOSE, classical and modern software engineering approaches can be adopted, depending on the requirements of the project under development as follows [46][50]:

- **Ontology-driven development** – facilitates the understanding of BOSE domain concepts and enables the identification, representation, discoverability, and actor identification of smart contracts.
- **Architecture based development** – facilitates development and management of complexity in team.
- **Model Driven Development** – enables model traceability for legal and smart contracts and automates the consistency checking between models of requirements, legal and smart contracts. It also reduces vendor lock-in and enables the automation of the generation of models and testing of smart contracts.
- **Pattern-based development** – helps to improve design, interoperability, security, migration of data, and proof of integrity. And is classified as emerging and common software patterns.
- **Agile base Development** – deals with the challenges of change in smart contracts, provides a prototype of a system based on user stories and requirements, and separates the development process into two processes development of smart contract and non-smart contract components with early identification and validation of smart contract.

These software engineering approaches, that can be adopted in BOSE, have several processes and tasks to deal with changing circumstances of Blockchain Based Software (BBS) development lifecycle as follows [50]:

- **The process of designing BBS** – influences the factors related to quality i.e., security, privacy, interoperability, and adoption to change. It concerns design and architecture related to on-chain or off-chain components of blockchain, permissioned and permission-less blockchain, size and type of blockchain transactions to response time, and storage

of data. Designing is related to several tasks such as state management, authentication and authorization, replication and synchronization, iteration, consensus, and incentive mechanism.

- **Process of implementation and testing of BBS** – includes tasks to test the smart contract code and review using conventional and modern testing techniques i.e., automation testing. For the implementation of BBS, the selection of development platforms and tools is based on maturity, ease of development, confirmation of time, security, and API support between nodes of the development environment.
- **Process of maintenance and analysis of BBS** – maintenance imposes issues of difficulty of continuous maintenance and integration e.g., update smart contracts and DevOps. Whereas analysis of BBS consists of several tasks of analysis such as requirements analysis based on user stories, feasibility analysis is performed through common software engineering techniques such as workshops and questionnaires as well as through core characteristics of blockchain such as immutability, transparency, data provenance, etc.

Several software modeling notations and approaches are adaptable in the BBS life cycle which includes models like requirement model, 4+1 architecture, data flow diagram, consensus, and smart contract models along with different modeling languages like Unified Modeling Language (UML). The People, for which the software systems have been developed, play an integral role in the BBS development lifecycle [50].

Agile software development methodologies are considered the most effective and adopted approaches for the software development lifecycle. Agile software development methodologies have several activities, listed below, that are directly related to core features of blockchain systems such as Trust, Decentralization, Traceability, Consensus and faster settlement, Security and privacy, Cost reduction, Individual control, innovation, and confidence, Immutability and Anonymity [51].

- Software projects depend on user stories, so the creation, refinement, and definition of these stories and acceptance by the domain expert or product owner are performed in the Agile software development lifecycle.
- Creation of incremental and iterative deliverables and milestones accepted by all the stakeholders' objectives and goals.

- Calculation of the complexity and incremental deliverable size with collaboration and pairing of frequent reviews.
- Participate and explore the designing of the technical and domain-oriented prototype and architecture activities.
- Keep track and perform testing techniques of incremental changes and generation of test reports before delivery.
- Deploy the product and perform inspections and business support.

These activities to develop BBS with all its capabilities and attributes, BOSE has some challenges that are encountered during the development life cycle of BBS like Need for new professional roles with specialized skill development education, Guidelines for security and reliability for every phase of software development lifecycle by performing testing activities for smart contracts and blockchain, Introduce new software design notations, modeling languages and metrics or update the classic ones [45].

2.8 Agile based Approaches for Development of DApps

In Agile based software development lifecycle, there are various approaches to contribute to every development phase of the agile software development lifecycle. Some most adopted approaches that would be efficient and quality-oriented development of blockchain and smart contracts-based DApps are as follows:

2.8.1 Test Driven Development (TDD)

Test Driven Development, or TDD is a process to write automated tests well before the development of functional code, these tests are executed in small and rapid iterations. If we analyze the concept of TDD, we can divide it into three main aspects that are the core of this process namely the test aspect, the driven aspect, and the development aspect. Firstly, the testing aspect involves the writing of automated tests for individual units of the functional program. These automated tests are small in size and manageable for every iteration. Secondly, the driven aspect of TDD ensures how this process is carried out to make decisions related to analysis, design, and implementation. TDD is useful to analyze the implementation with incomplete or inconsistent and open to change requirements. Lastly, the development aspect of TDD is

to make sure that the testing phase is a vital component of the development process. If an automated test fails, the development team knows the new change and logic of change now development team has to update the production code and automated test suites [52].

TDD process is beneficial in different aspects of the software development lifecycle such as Comprehension of the programming which helps to understand the code during the maintenance phase of the lifecycle of a project. Efficiency in debugging and defect removal of code is increased with TDD because more time is spent on writing and executing test cases. Test assets are generated, and a testable environment is established where programmers write code that is automatically testable. TDD is favorable in reducing defect injection because fixes during maintenance and small code changes may cause 40 times more error-prone, and there are good chances of injecting new faults during maintenance [53][54].

Changes in requirements or any other phase of the software development lifecycle should be understandable for the development team. Refactoring in TDD makes the change in the internal structure of software easier to understand and cheaper to modify. In refactoring change is modified without any external or observable change of behavior [54]. Refactoring helps the developer to add new code, with improved design, and better methods of code cleanup of existing code [54]. TDD team faces several challenges in the process such as difficulty in writing a test, complex applications like complicated algorithms, numeral and parallel computing, difficulty in 100% code coverage, and lack of software engineering methodologies, tools, and standards for TDD [54].

2.8.2 Acceptance Testing

After the development phase in the software development lifecycle acceptance testing is performed before the deployment of the system. Acceptance testing is performed in different scenarios with distinct characteristics i.e., Operational Acceptance Testing (OAT), User Acceptance Testing (UAT), and alpha and beta testing. User Acceptance Testing is considered most suitable for a testing approach where the system is evaluated with the requirements of the end-user or other stakeholders, to perform confirmation that the developed software meets the requirements of end users. Acceptance testing is a type of black-box testing, where only external behavior is tested following the acceptance criteria [55]. UAT prevents software development from failing, three targets are achieved in this process of testing, namely: UAT focuses on functionality and business logic instead of the internal structure of the software. It verifies

the developed software following the end-user requirements. It also checks the level of quality in functionality and how the software system has been completed [56].

2.8.3 Planning for new Release and Iteration

Planning is an initial and important phase in the XP development process. Two fundamental points are in consideration while working on the planning phase. First, is under consideration that “Which business value-oriented modules can be developed within time constraints?” The second point under consideration is “What strategy would be adopted for the upcoming iteration?” The planning phase is carried out in two sections i.e., Iteration planning and Release planning, while the whole planning phase takes merely one to two days to complete [20] [30]. In the first section of release planning the main objective is to discover the key points that need to be completed and a delivery plan is also prepared to achieve these discovered points of this section. In the second section of release planning the developers devised a plan of the tasks to implement the targeted points. The developers plan the activities to carry out and evaluate the development process within the time, cost, and resource constraints [30].

2.8.4 Pair Programming

As the name indicates Pair Programming consists of a pair of two software engineers who code for the development of the same system, at one computer, which leads to the best quality software at a cheaper cost. Pair Programming is time and cost-effective because the time duration is short for identification and rectification of defects [30]. Pair Programming consists of several factors that contribute to the successful implementation of this technique as follows [57]:

- The quality of the software is improved by following the pair programming technique in the software development lifecycle.
- Team spirit is practiced by building and training team and pair management.
- The personality of a pair of software engineers should be mature and strong enough to deal with the conflicts of understanding on the same problem.
- The working environment should be supportive of the limitations and capabilities of any individual of the pair of software engineers.
- Proper project management to deal with the challenges related to planning and estimation.

- Pair programming results in improved design and problem-solving through the exchange of ideas.

2.8.5 Small Incremental Releases with Review Meetings and Project Velocity

The development team in XP methodology releases versions of the software system in iteration. At the end of every iteration, there is an increment of the DApp system which is under development, and a review meeting to evaluate the performance of the latest release and this circle continues [58]. Project velocity measures the progress of the project at the current iteration by calculating the estimation for the completed user stories-based tasks during the current iteration. Project velocity is defined during planning for the new iteration [59]. Estimation at the early stage of the software lifecycle is difficult so project velocity is helpful in the estimation of software cost and completion time [59].

2.8.6 Class Responsibility Collaborator (CRC) Modeling

Class Responsibility Collaborator or CRC card modeling is an object-oriented technique to design and analyze the development of software systems. It is a low-cost, low-technology technique, and easily adaptable technique that people without experience in object-oriented programming concepts can also use. It is based on a simple index card method, where a card is a representation of a single class in the design and analysis session of object-oriented development [60]. It is a useful technique to define objects and the roles of classes in object-oriented development. Each CRC card has three sections that represent related information, i.e., Name represents an appropriate and suitable name for the class. Responsibilities represent the doable tasks of the objects of the relevant class. Whereas Collaborators are other classes carrying out the same responsibility. In every CRC card, a description in the form of notes about the implementation of the class like data types of the data elements, is written on the back side of the card. CRC cards also support working in groups, ancestry details of parent and derived classes are also written in the collaborators section [60][61].

2.9 SMART Analysis of Project Goals

Software engineering projects are solutions to some specific problems that need to be addressed for different aspects i.e., financial or service oriented. Every software engineering project has specific objectives and goals, proper management and understanding of these objectives are crucial for project success. SMART is a method to assist people in setting project goals and objectives, these SMART objectives are as [62][63]:

S – Specific, the goal should be specific to what stakeholders want to achieve from the project.

M – Measurable, the goal should be measurable so that the progress of the project can be trackable.

A – Achievable, the goal should be achievable and challenging in available resources.

R – Realistic, the goal should be realistic to achieve with available resources and time frame.

T – Time bounded, a specific time frame within available resources.

“T” in SMART setting of goal Time-bound is replaced with Traceable with to use SMART objects concept for requirements traceability in software development life cycle [66].

2.10 Summary

In this chapter, a detailed literature review is provided with proper references. Literature is related to agile-based software engineering, Blockchain and DLT, Smart contracts, DApps, and their interrelationships in the field of software development. Applications, advantages, and shortcomings of these software engineering and technological terms are also discussed in this chapter. Literature related to efficient and quality-oriented DApp development using software engineering methodologies and different agile-based approaches with SMART objectives is also reviewed and discussed in this chapter.

Proposed Methodology

3.1 Introduction

In Chapter 3, the rationale, motivation, and proposed framework for efficient and quality-oriented development of DApps are described. Rationale and motivation are illustrated with appropriate and logical points of this research work. The proposed framework for efficient and quality-oriented development of DApps is also illustrated in pictorial representation with a complete description of each phase of the proposed framework. In the last section of this chapter, key attributes of the proposed framework are described.

3.2 Rationale and Motivation

Since the inception of software engineering in the 1960s, it has been an evolving discipline with continuous research and development in this area. The history of software engineering is not older than other engineering disciplines; however, the growth and adoption of this discipline is continuous, rapid, and future prosper. Blockchain technology gained popularity after the introduction of Bitcoin in 2008 by Satoshi Nakamoto because it is the premier technology of Bitcoin. When these two technologies, i.e., software engineering and blockchain, are implemented with each other they introduce a new term Blockchain Oriented Software Engineering or, BOSE. This term is devised to explain and provide solutions to the problems that are associated with the software systems where blockchain technology is implemented as a part of the software systems i.e., DApps.

The Institute of Electrical and Electronics Engineers (IEEE) defines software engineering as a set of approaches for systematic, disciplined, and quantifiable development, operation, and maintenance of software systems. Several software engineering approaches are in practice for a disciplined, systematic, and quantifiable software development lifecycle i.e., Waterfall model, Spiral model, Agile methodologies. These approaches can also be used for the development of complex BBS which are relatively new in the software industry and demand as they are immutable, transparent, and secure solutions [4]. However, many technological misconceptions are present in the market about blockchain technology and its use in software systems which requires some standards for the proper management of BBS systems. Such technological and management problems and misconceptions lead to the concept of BOSE, in which disciplined, and quantifiable approaches are used to develop, operate, and maintain the BBS [4][26].

In the development lifecycle of BBS systems efficient and quality-oriented approaches are deficient. BOSE can mitigate the challenges for efficient and quality-oriented development of blockchain based systems. Agile-based approaches in BOSE are suitable for the development of BBS or DApps, because in DApps requirements are not completely understood and there is always a possibility of change in the requirements throughout the development lifecycle [4]. Change is constant and inevitable, a concept that is frequently used in the software engineering community, so whenever a new methodology or concept is devised for the development of modern software systems this concept of change is always under consideration.

The proposed framework for efficient and quality-oriented development of blockchain-enabled smart contracts-based DApps considers the software engineering concept of change. A hybrid framework based on XP and Scrum methodologies of Agile is proposed, which adopts practices like simplicity, TDD, pair-programming, and release planning while prioritizing the user stories with iterative and incremental development for efficient and quality-oriented DApps. For simplicity in the design phase, we found CRC modeling a useful practice that can lead to a mature UML diagram from simple to detailed architecture. We found agile testing i.e., acceptance testing with collaboration with stakeholders and TDD, an effective practice for quality, efficiency, and rapid development of a testable product.

We found from related research that pair programming can reduce the defects and the chances of failure, moreover, it can also improve the debugging speed. The requirements or planning phase is the core of software development where the percentage of uncertainties about the actual requirements is relatively the highest as compared to other phases of the DApps development.

These are some practices from agile methodologies that can be effective and suitable for the efficient and quality-oriented development of DApps. Sections 3.3. and 3.4 describes the details of the proposed framework for efficient and quality-oriented development of DApps.

3.3 Proposed Framework

The proposed framework is primarily based on Agile approaches, it is a hybrid framework comprised of different practices of mainly two agile methodologies Scrum and XP. The framework is partitioned into three main sections and their relevant steps of different phases of the DApps development lifecycle. The main sections are Vision, Design and Implementation, and Maintenance. The “design and implementation” section is further partitioned into three phases of the DApps development lifecycle namely the Release Planning and Informal Design Phase, the Formal Design Phase, and the Testing and Implementation Phase. All the sections, sub-sections, and phases of development have an identity or tracking number for a better understanding of the proposed framework. The “design and implementation” section is split into two development-based sub-systems namely SCs Development and App System Development, these two sub-systems are designated to the development phases of smart contract and application development respectively, this division of the system is a convenient way to manage the development activities of the complex architecture of DApps. These partitions will provide convenience to practitioners and software engineers who will implement this proposed framework in the practical development of DApps for efficient and quality-oriented products.

3.4 Vision (Define Vision)

The section is comprised of three steps, that define a vision and goal of the project and the stakeholders. The objectives of the project for which the different development lifecycle phases will be implemented are explained and documented in this section by defining the user stories-based UML use cases. The three phases of defining a vision of the project are as follows:

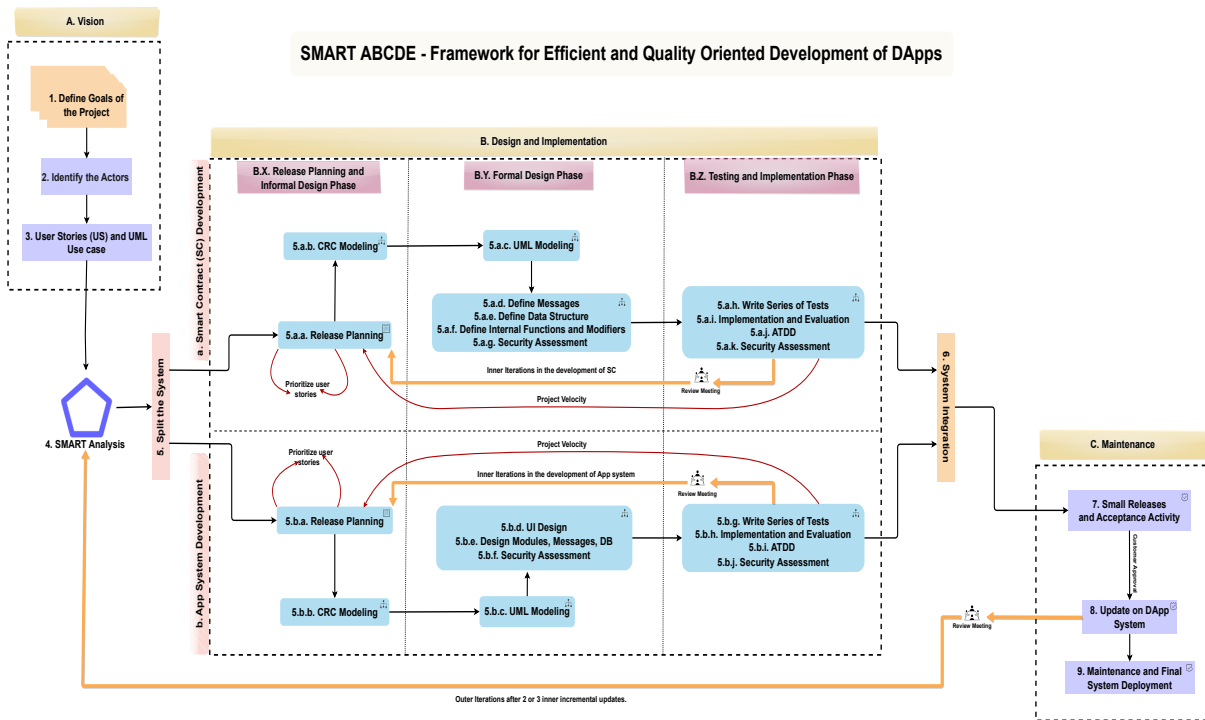


Figure 3.1: A framework for efficient and quality-oriented development of DApps

A complete flow of the proposed framework is illustrated in Figure 3.1, and section-wise complete descriptions of the proposed framework are described in sections 3.4 to 3.6.

1. **Define Goal of the Project** Confirm that a problem exists, for this purpose carry out related research and determine that a system i.e., DApps, is required to solve the problem, and after that produce a feasibility report. Summarize the feasibility report in 20 to 30 words that will sum up the goal of the project to develop a system as a solution to the problem. Display this 20 to 30-word summary of the feasibility report in a place that is visible to all the stakeholders of the project. This is a recognized SCRUM practice to define the goal of the project i.e., the DApp system, and communicate it to the whole team [4].
2. **Identify the Actors** Perform related research to identify the actors that will directly or indirectly associate with the DApp system. The list of the actors could be long as it includes human roles i.e., user, product owner, project manager, programmers, and external systems or physical devices to exchange information with the DApp system [4].

3. **User Stories and UML Use Cases** User stories are a technique to explain the requirements of the system. These user stories are used to design UML use case diagram, users and their actions are under consideration at a high level. The DApp system is considered full of all possible functionalities without the architecture of the DApp system will be considered in the UML use case diagram. The diagram represents the relationships graphically among the actors and user stories, this graphical representation will help to understand the functionalities of the actors and the relationships among the actors with their user stories [4].
4. **SMART Analysis** The Goal of the project at a high level has been defined in step 1 of section 3.4, in a summary of 20 to 30 words, however, there is a need to define specified goals for every increment. Use the SMART method to define specific, measurable, achievable, realistic, and time goals for every increment with 2 to 3 inner iterations, so that in the next phases of the agile software development lifecycle the SMART analysis can help to maintain a relation of development practices with the goal of the project. It is not mandatory to achieve these specified parameters, however, it can be helpful to maintain a record of iterations and analyze them with the progress in development. Analysis will be conducted after 2 to 3 inner iterations and SMART will be updated after every increment.
5. **Split the System** Split the DApp system for development purposes into two sub-systems [4]:
 - **a. Smart Contracts (SCs) Development** to carry out the agile-based software engineering practices specifically for the development of SCs of DApp system on blockchain technology.
 - **b. App System Development** to carry out the agile-based software engineering practices focused on the development of Applications i.e., frontend and backend, of DApp system.

At this point, an analysis of the architecture of the whole system should be conducted and specify the on-chain or off-chain. Normally the data and processing related to transactions and need to be transparent and immutable is managed on-chain [4].

3.5 Design and Implementation

a. Smart Contracts (SCs) Development

5.a.a. Release Planning Release planning for the new iterations and increment of SCs has some steps as follows:

- Domain experts read the user stories related to the on-chain development of SCs and conduct elicitation sessions with stakeholders to understand the requirements written in the form of user stories earlier in step 3.
- Stakeholders assign a certain priority value and prioritize the user stories. Project team members examine the user stories and assign a cost, and weeks of development to each user story, user stories with high costs i.e., long development time, are discussed again with the stakeholders and split into smaller stories.
- After this discussion, stakeholders reassign priority values, whereas project team members assign new costs to these smaller user stories.
- Stakeholders and project team members finalize the tasks for the next release.

5.a.b. CRC Modeling Design CRC cards compliant with the prioritized user stories for modeling the requirements for the development of SCs, step 5.a.a. These cards will assist in UML modeling in the design phase, step 5.a.c., of SCs.

5.a.c. UML Modeling Design the SCs sub-system following the prioritized user stories requirements, with the help of UML modeling. Marchesi, L. et al. (2020) provide a list of stereotypes for UML modeling i.e., class diagrams and sequence diagrams of SCs [4]. Use these proposed stereotypes while designing UML diagrams for SCs.

5.a.d. Define Messages Define and establish a flow of Ether transfers for transactions among on-chain SCs, external SCs, and the Application system. Marchesi, L. et al. (2020) provide UML state charts to document these messages and transactional flows for SCs. Use these state charts to design a UML state sequence diagram and document these interactions.

5.a.e. Define Data Structure Define the data structure of on-chain SCs, the external interface, and events that will directly and indirectly interact with SCs [4].

5.a.f. Define Internal Functions and Modifiers Define the modifiers, a special feature of SCs, special functions with preconditions to test a function before its safe execution, and also define internal and private functions of SCs [4].

5.a.g. Security Assessment Perform security assessments to maintain the special security aspect of SCs. Follow the instructions described in Marchesi, L. et al. (2020) research for security assessment of SCs.

5.a.h. Write Series of Tests Write a series of test cases for the implementation and evaluation stage at 5.a.i. and acceptance scenarios for acceptance test-driven development, following the prioritized user stories of step 5.a.a., that are defined, examined, and related to requirements in the form of user stories with a defined goal of the project, step 1 to step 4. The security aspect of SCs is critical, in these test cases based on user stories and implementation of the architecture of the DApp System.

5.a.i. Implementation and Evaluation Implement the code for the development of SCs and perform automation unit testing using specific testing environments for SCs i.e., Truffle. It is recommended that perform this implementation process in pairs, where one individual will implement the code for SCs and the second one will monitor the implementation, because SCs development involves blockchain which is a separate entity, and using it for testing purposes will require extra resources which will increase cost of development, monitoring the implementation is considered an efficient de-bugging technique at the time of development. This practice will also help in the development of critical software, a better understanding of the DApp system, and the learning of junior-level developers.

5.a.j. Acceptance Test Driven Development (ATDD) Involve a resourceful and reliable domain expert to validate the implementation following the acceptance scenarios defined at 5.a.h.

5.a.k. Security Assessment Perform security assessments to maintain the special security aspect of SCs. Follow the instructions described in Marchesi, L. et al. (2020) research for security assessment of SCs.

Note: After security assessment of SCs, at step 5.a.k, arrange review meetings with stakeholders to discuss the goals of the iteration to improve or maintain the development standards or practices in the next iteration. Moreover, after security assessment send a message of time velocity as a reminder to maintain check and balance in the development life cycle of DApp.

b. App System Development

5.b.a. Release Planning Perform release planning as described at 5.a.a. and update the user stories that are related to App System and users interacting with SCs.

5.b.b. CRC Modeling Design CRC cards, compliant with the prioritized user stories for modeling the requirements for the development of the App System. These cards will assist in designing UML models in the design phase, step 5.b.c., of the App System.

5.b.c. UML Modeling Design the requirements from prioritized user stories, with the help of UML diagrams i.e., class diagrams and sequence diagrams. In UML modeling of the App System includes the interaction of the App System with blockchain and other off-chain storage elements.

5.b.d. UI Design the User Interface (UI) of the App System with related research on market demand choose the best or in budget available resources for design and development. End users prefer a system with a simple and interactive interface that is easy to use and control [4].

5.b.e. Design Modules, Messages, DB Decompose the App System in modules, a modular approach of development is encouraged to adopt. It is recommended to design state and data flow diagrams to represent how messages and data flow occur in different modules of App Systems and how SC transactions are carried out in DApp systems [4].

5.b.f. Security Assessment Perform security assessments to maintain the special security aspect of DApp. Follow the instructions described in Marchesi, L. et al. (2020) research for the security assessment of SCs and their interactions with the App System in the architecture of DApp.

5.b.g. Write Series of Tests Write a series of test cases for the implementation and evaluation stage at 5.b.h. and acceptance scenarios for acceptance test-driven development, following the prioritized user stories of step 5.b.a., that are defined, examined, and related to requirements in the form of user stories with a defined goal of the project, step 1 to step 4. The security aspect of SCs is critical, in these test cases based on user stories and implementation of the architecture of DApp System.

5.b.h. Implementation and Evaluation Implement the code for the development of the App System and perform automation unit testing using testing tools i.e., JUnit. It is recommended that perform this implementation process in pairs, where one individual will implement the code for App System and the second one will monitor the implementa-

tion, because App System development involves blockchain which is a separate entity, and using it in testing purposes will require extra resources which will increase the cost of development, monitoring the implementation is considered an efficient de-bugging technique at the time of development. This practice will also help in the development of critical software, a better understanding of the DApp system, and the learning of junior-level developers.

5.b.i. Acceptance Test Driven Development (ATDD) Involve a resourceful and reliable domain expert to validate the implementation following the acceptance scenarios defined at 5.b.g.

5.b.j. Security Assessment Perform security assessments to maintain the special security aspect of DApp. Follow the instructions described in Marchesi, L. et al. (2020) research for the security assessment of SCs and their interactions with the App System in the architecture of DApp.

Note: After security assessment of the App System, at step 5.b.j, arrange review meetings with stakeholders to discuss the goals of the iteration to improve or maintain the development standards or practices in the next iteration. Moreover, after security assessment send a message of time velocity as a reminder to maintain check and balance in the development life cycle of DApp.

6. **System Integration** Perform DApp system integration in a local environment to test and validate that the integration process is carried out as expected [4].

3.6 Maintenance

7. **Small Releases and Acceptance Activity** After DApp system integration this point is a release point for new updates based on the defined goals, SMART analysis, and release planning performed at earlier stages of the DApp development lifecycle. Now conduct acceptance meetings with all the stakeholders for the approval of the new release.
8. **Update on DApp System** Update the system with new release after acceptance meetings and approval of stakeholders and conduct a final review meeting to demonstrate the new update of the DApp system. After the review meeting, go to step 4 SMART analysis, and redefined the goals for the next increment.

9. **Maintenance and Final System Deployment** If the problem that was defined in the vision section has been resolved or the DApp system has been obsolete then perform final system deployment, otherwise keep working on the maintenance of the deployed DApp system.

3.7 Key Attributes of the Proposed Framework

The proposed framework is based on the agile approaches i.e., Scrum and XP, that are best suited to the efficient and quality-oriented development lifecycle of DApps. The proposed framework possesses key attributes that are helpful for the efficiency and quality of the development of DApps. The four key attributes are illustrated in Figure 3.2 and the description of each attribute is described in sections 3.7.1 to 3.7.4, as follows:

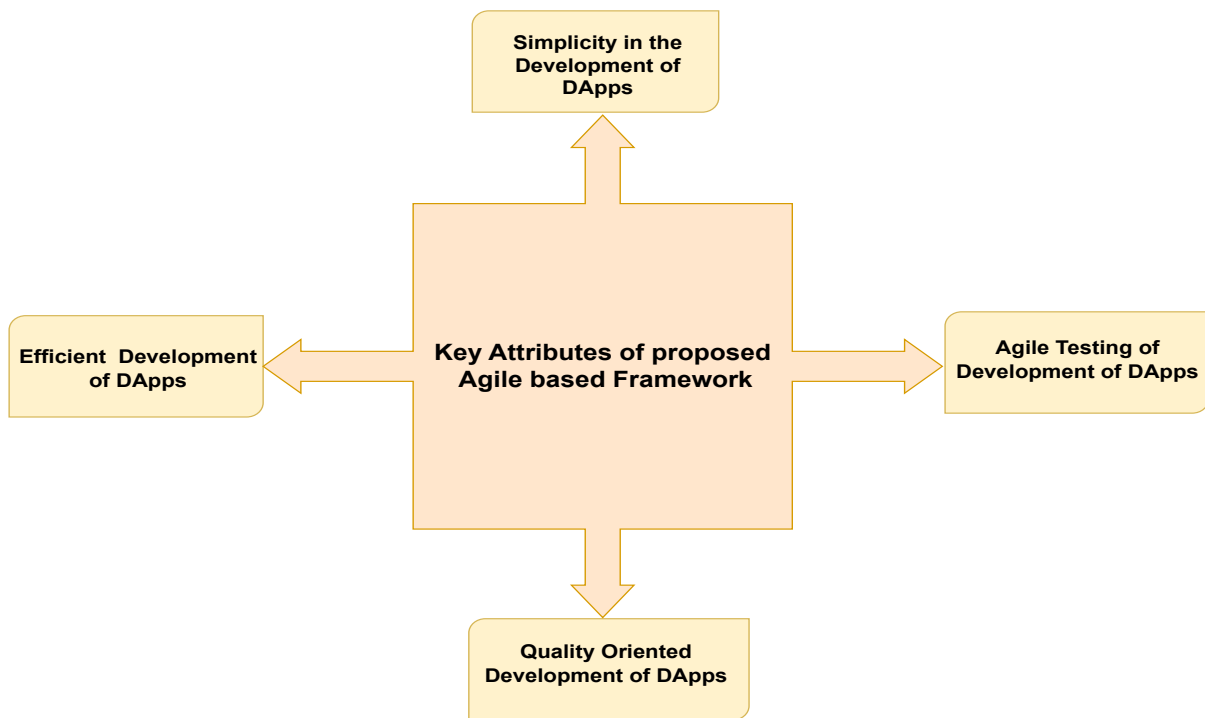


Figure 3.2: Key Attributes of the Proposed Framework

3.7.1 Simplicity of the Proposed Framework for the Development of DApps

The proposed framework provides simplicity in the lifecycle of the development of DApps, the practices that are introduced for simplicity of the development of DApps lifecycle are as follows:

- **Prioritize the DApp system** requirements from user stories to facilitate the focus of the development team on the critical and cost-effective requirements i.e., Prioritize user stories in step 5.a.a. Release Planning.
- **Analyze the project goal** defined in step 1, Define Goals of the Project, at step 4, SMART Analysis, which provides a focused and simplified representation of the project goal.
- **Partitions of the complex DApp system** into sub-systems i.e., SCs Development and App System Development of step 5, Split the System, Sections i.e., Vision, Design and Maintenance and Maintenance, to make the development easy and simple for the development team.
- **CRC Modeling** provides a simple and easy way to design user stories-based requirements of the DApp system, as compared to complex UML diagrams i.e., class diagrams, for the non-technical individuals in the project team.
- Continuous improvement through **small and to-the-point incremental iterations and releases with review meetings.**

3.7.2 Quality Oriented Development of DApps using the Proposed Framework

The proposed framework provides quality in the lifecycle of the development of DApps, the practices that are introduced for quality of the development of DApps lifecycle are as follows:

- **Testing practices i.e., TDD and ATDD** to carry out testing and development activities simultaneously can mitigate the debugging challenges for quality-oriented development of DApps.
- **CRC Modeling** can provide an easy and simple way to design and brainstorm the requirements of the DApp system with stakeholders and design the in-detail UML diagrams i.e., class diagram, with improved quality and ease.
- **Prioritizing the user stories** can improve the requirements elicitation capabilities which leads to quality-oriented development through focused development, enhanced customer satisfaction, and efficient resource allocation.

- **Small and Incremental releases** provide quality through continuous improvement and adaptation, increased user satisfaction and engagement, manageable resources, reduced complexity, improved focus, and testing.

3.7.3 Agile Testing in the Proposed Framework for the Development of DApps

The proposed framework supports agile testing in the lifecycle of the development of DApps, the practices that are introduced as agile testing for the development of DApps lifecycle are as follows:

- **TDD** approach reduces the bugs, increases confidence in code, and improves communication between the developers leading to efficiency and quality in the development of DApps.
- **Pair Programming** improves the code quality, reduces errors, or bugs, increases knowledge sharing, supports improved communication between programmers, and increases productivity that leads to efficient and quality-oriented development of DApps.
- **ATDD** facilitates the development of the DApp system with improved communication, early feedback, increased confidence in the software, improved alignment with user needs, and reduced risk of scope creep which leads to efficient and quality-oriented development of DApps.

3.7.4 Efficient Development of DApps in the Proposed Framework

The proposed framework contributes to the efficiency in the lifecycle of the development of DApps, the practices that are introduced for efficiency of the development of DApps lifecycle are as follows:

- **Prioritizing the user stories** provides efficiency in the development of DApps through focused development efforts, early validation and feedback on the high-priority requirements, reduced risk, improved user experience, and efficient resource allocation.

- **SMART analysis** can contribute to the efficient development of DApps through clarity and focus on the goal of the project, measurable progress, realistic expectations about project goal, Relevance to DApps system, and time-bound (in the proposed framework it can be flexible) completion.
- **CRC modeling** can provide efficiency in the development of DApps through improved class design and modules, enhanced communication and collaboration, simplified development, and making it readable, and understandable.
- **Small and incremental iterations and releases** contribute to efficiency in the development of DApps through early feedback and course corrections, continuous improvement, adaptation, reduced scope creep and maintaining focus, enhanced user experience and feedback, validated learning with knowledge sharing, reduced maintenance burden and improved agility, early market validation, and adaptation.

3.8 Summary

In this chapter rationale and motivation for the proposed framework are described in detail in support of the logic to propose this framework. After rationale in support of the proposed framework, the actual framework is illustrated with the proper diagrammatic flow. At the end of the chapter, the key attributes of the proposed framework are also discussed.

Validation of Proposed Framework

4.1 Introduction

This chapter describes how the proposed framework is validated for efficient and quality-oriented development of DApps, there are two methods of validation introduced. One is theoretical and a questionnaire survey and the other is a case study to examine the proposed framework in a real time environment with the help of a software development team. A brief introduction of both types of validation methods, and how they are prepared, initiated, and completed, are described in this chapter.

4.2 Questionnaire Survey

A questionnaire survey is a method of collection of data to validate and analyze a theoretical concept, comparative study, or review of the public or community of users of a new product or a proposed framework. This research questionnaire survey is to validate and analyze a proposed SMART agile-based framework for efficient and quality-oriented development of DApps. The proposed framework claims that it provides simplicity, and agile testing with efficient and quality-oriented development of DApps. The questionnaire survey validates and assures these claims through diverse but simple multiple-choice questions that are based on the professional life of the respondents.

Google Forms is a platform that is used for conducting this survey and collecting relevant data from respondents to analyze their responses. It proved a valuable tool because of its auto-charts generation feature from the collection of data, however, several factors are considered

while using Google Forms for questionnaire survey for validation purposes, some of the key factors are as follows:

- **Target Audience:** The Audience for this questionnaire survey is professionals in the software industry, who have at least 1 year to 5+ years of experience in the software industry. The expected size of the audience was 80 individuals and the received responses were 83 which is above the estimated target.
- **Survey Design:** Questions are clear, logical, and flow, most of the questions are multiple-choice however there is always a part or text field for open-ended questions.
- **Survey Distribution:** The survey is shared with only professionals in the software industry who have good knowledge of blockchain technology and its contributions to the modern information technology world.
- **Data Analysis:** Data is collected through Google Forms and there is a feature to generate charts and analyze the collected data using formulas in Excel or Google Sheets linked with Google Forms.

The questionnaire survey has a sequence from start to end every question followed by the previous one. The survey consists of five sections, each section has relevant questions to get appropriate information from experienced professionals in the software industry, five sections of the survey are as follows:

1. **Purpose of Survey** – This section describes the introduction and objectives of the questionnaire survey to respondents, with a clear statement of privacy protection. A clear figure of the proposed framework is also attached in this section.
2. **Respondent Information and Demographics** – In this section, introductory information has been collected from respondents to get an idea about the background and demographics of the respondents.
3. **In practice Agile Development Methodologies and analysis of the project goal** – In this section questions are related to the current practice of agile methodologies in respondents' organization and to check the familiarity of respondents with agile development methodologies in their organization.

4. **Evaluation of proposed framework** – This section has four sub-sections that possess the questions related to the claims of the proposed framework such as simplicity, agile testing, and efficient and quality-oriented development of DApps.
5. **Conclusion and Feedback** – In this section any additional feedback is recommended to share, however, this section is not mandatory to fill.

Analysis of the collected data in the questionnaire survey is available in Chapter 5 and a complete form that is used for validation purposes is attached as Annex ‘A’.

4.3 Implementation Example - Development of DApp

The method to validate the proposed framework is a case study implementation of the proposed framework, to develop a DApp, following the proposed SMART analysis of project goals or objectives and agile-based efficient and quality-oriented development of DApps. The case study is a supply chain management system for grapes juice production, an example development of DApp. The complete software and supply chain lifecycle process is carried out in this case study by following the proposed agile and SMART objectives based on efficient and quality-oriented development of DApps. This case study under the proposed framework is practiced in a completely independent and unbiased environment, with the collaboration of a software house V3Solutions, where all practitioners are trained and well-equipped with relevant tools to perform the implementation tasks related to the SMART agile-based framework. It is an effort to develop a connection between academia and industry and promote the culture of Research and Development (R&D). The complete process of implementation of the case study is available in Chapter 5.

4.4 Summary

In this chapter two validation methods are discussed that are carried out to ensure the validity of the proposed framework for efficient and quality-oriented development of DApps. The factors that were in the focus questionnaire survey, all five sections of the survey, and the tool used for this survey generation are discussed in detail in this chapter. A case study also discussed briefly, which was conducted with industry collaboration is introduced in this chapter and a detailed analysis of both validation methods is available in chapter 5.

Case Study Implementation of Proposed Framework

5.1 Introduction

The proposed framework implemented in a real development environment as a case study is discussed in detail in this chapter. A practical description of every section, subsection, step, and phase of the development of DApps is discussed in this chapter. Formal and Informal modeling, testing practices, and other proposed approaches are discussed in this chapter.

5.2 Case Study for Implementation

5.2.1 Objective

The purpose of this case study is the development of an efficient and quality-oriented blockchain-enabled smart contracts Decentralized Application (DApp) by following the proposed SMART Agile framework for efficient and quality-oriented development of DApps. This case study implementation is an effort to develop a connection between academia and industry and promote the culture of Research and Development (R&D).

5.2.2 Scope and Criteria

The proposed framework for efficient and quality-oriented development of DApps is hybrid and has a wide scope in the blockchain-based software development industry. The criteria that are adopted to implement the framework are solely practical and result-oriented.

5.2.3 Level of Assurance

All practitioners are trained and well-equipped with relevant tools to perform the implementation tasks related to the SMART agile-based framework.

5.2.4 Description of the Project

A Supply chain management system is the process of managing different phases of the flow of items or services that can help businesses reduce costs with customers' or stakeholders' satisfaction. The use of modern technology like blockchain, in supply chain management provides transparency, security, and improved efficiency. This project is a DApp authenticity management system backed by Ethereum smart contracts platforms. A DApp supply chain solution to verify the authenticity of an Agri product from plantation to sales to customer. The development of DApps is complex as compared to traditional software development projects, so there is a need for a systematic, disciplined, and quantifiable approach that is primarily based on customer collaboration and change management in the development of the DApp project. This complexity of DApps increases when a project like supply chain management demands efficiency and quality-oriented development with security. The proposed framework provides a full package, from project goal definition to deployment. In this project, our project team follows this framework for the secure, transparent, efficient, and quality-oriented development of DApps.

5.3 Implementation and Analysis

This section discusses the implementation details of the proposed framework from project goal definition to deployment with results and analysis. Every section and phase in the development of DApps is discussed with practical illustrations.

5.4 Vision (Define Vision)

5.4.0.1 Define Goal of the Project

To provide authenticity and transparency in the supply chain management process from production level to sales and distribution, through Ethereum smart contracts enabled DApps.

5.4.0.2 Identify the Actors

From production to sale and distribution there are different roles and actors involved in a grapes juice supply chain management DApp system, prominent actors in this system are **Farmer, Food Inspector, Grapes Fruit Juice Producer, Distributor, and Customer.**

5.4.0.3 User Stories and UML Use Case

DApp system requirements are collected by way of user stories, which leads us to a generic UML use case diagram for a better understanding of every user story and its relationship to the actors of the system. Some user stories are as follows:

- Farmer plants grapes on a farm and harvests them.
- The Food Inspector audits the quality of grapes, and the farmer then processes the grapes.
- The Grapes juice producer blends the juice of grapes with other ingredients and produces juice.
- The Food Inspector verifies and validates the quality of juice and issues a certificate.
- The Grapes juice producer sends juice products to distributors and sells them after packaging.
- The distributors sell the different juice products i.e., grapes juice.
- Customers buy grapes juice from distributors. Based on the above-mentioned use stories our design team developed a UML use case diagram as in Figure 5.1.

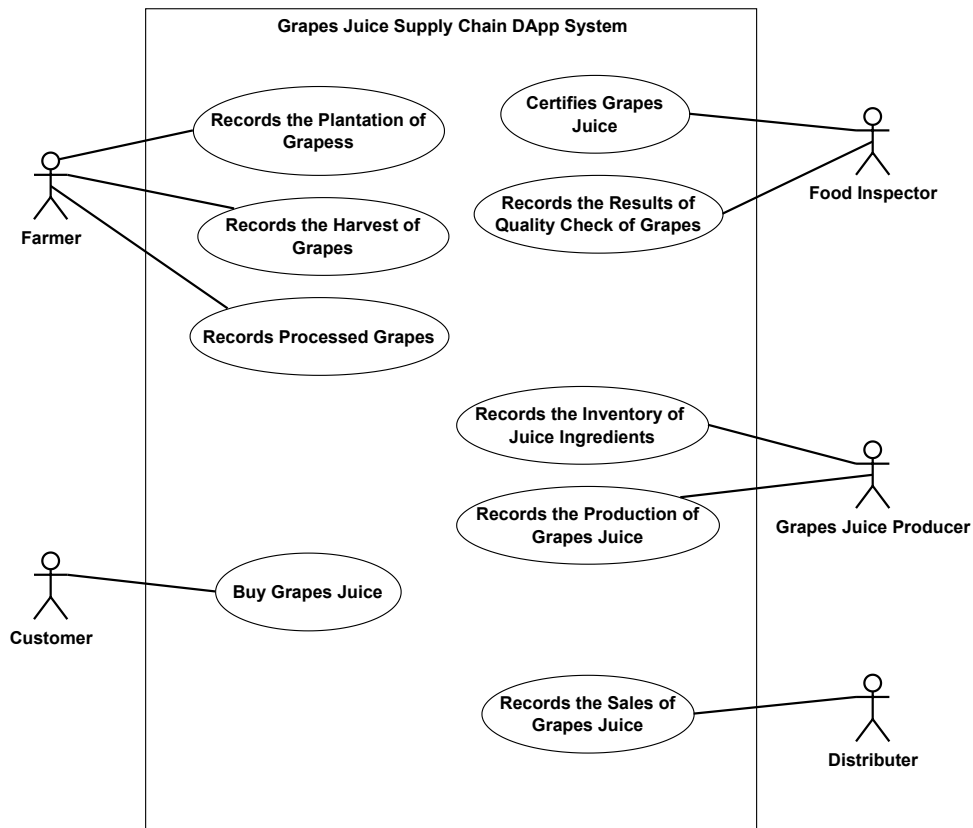


Figure 5.1: UML Use case diagram based on user stories.

5.5 SMART Analysis

SMART Analysis has been performed after every increment in the development phase to make the development process and goals more specified and achievable. Some instances of SMART statements that are used in the development process of grapes fruit juice supply chain management DApp system, to specify the goals are as follows:

- Design and develop the front end of the farmer module, using grapes and green color palate, in 2 to 3 weeks.
- Create a smart contract for farmer records and integrate it with the App system of DApp, in 2 to 3 weeks.
- Integrate the smart contracts of all roles in the DApp system, in 1 to 2 weeks.

5.6 Split the System

After SMART analysis of project goals, we split the system into two separate sub-systems for development purposes, one is for smart contracts development and the other is for front-end development of the DApp system.

5.7 Design and Implementation

5.7.1 Smart contracts Development

5.7.1.1 Release Planning

Planning for the next release of smart contracts in increments after 2 to 3 iterations has been carried out at this stage, user story-based requirements are prioritized, using priority values, with collaboration and discussion with stakeholders.

5.7.1.2 CRC Modeling

Based on prioritized user stories-based requirements, informal modeling is performed where CRC cards are designed to illustrate the user stories in the form of classes at a high level. A complete CRC card design, arranged concerning collaborators, is shown in Figure 5.2.

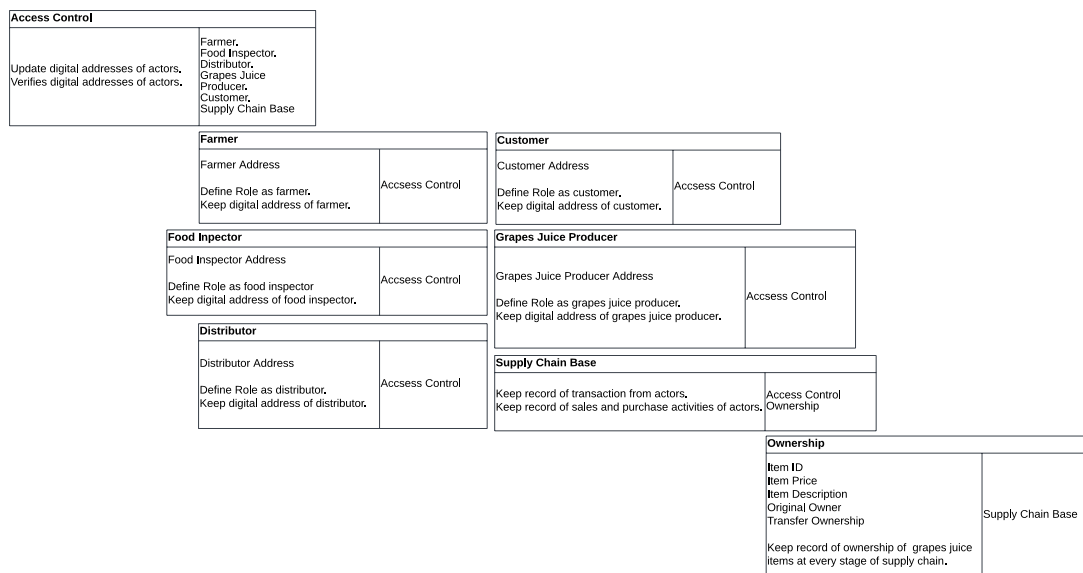


Figure 5.2: CRC Modeling, arranged with respect to collaborators

5.7.1.3 UML Modeling

UML diagrams i.e., class diagrams and sequence diagrams, are designed after CRC modeling, to help the programming team understand the relationship between different classes and their functions, messages, modifiers, and attributes. A class diagram is shown in Figure 5.3, which is a standard UML class diagram of the grapes juice supply chain management DApp system to illustrate the synchronous or asynchronous relation of all the available classes of the DApp system.

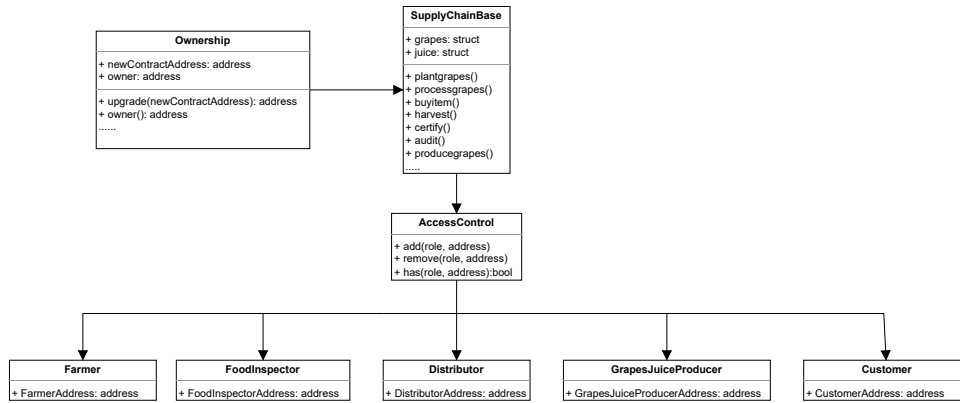


Figure 5.3: The Standard UML Class Diagram of Supply Chain Management DApp System of Grapes Juice

Every class shown in Figure 5.4, has smart contract files with the extension .sol (solidity-based), all these .sol files perform different functionalities. Classes of roles or actors i.e., farmer, food inspector, distributor, grapes juice producer, and customer have public-type account addresses and private roles. The supply chain base class controls all the records and states of grapes and grapes juice. Ownership class is assigned and reassigned by the owner of the product as shown in Figure 5.4.

The sequence diagram in Figure 5.5 illustrates the flow of the whole process from start to end of the process of grapes fruit supply chain management DApp system. Every factor involved in the supply chain process is associated with the supply chain base contract and the flow of this grapes fruit supply chain DApp system continues with the internal response from the supply chain base contract. The supply chain base class as shown in Figure 5.4 is in a relationship directly or indirectly with all the other classes the same aspect is shown in Figure 5.5 as well.

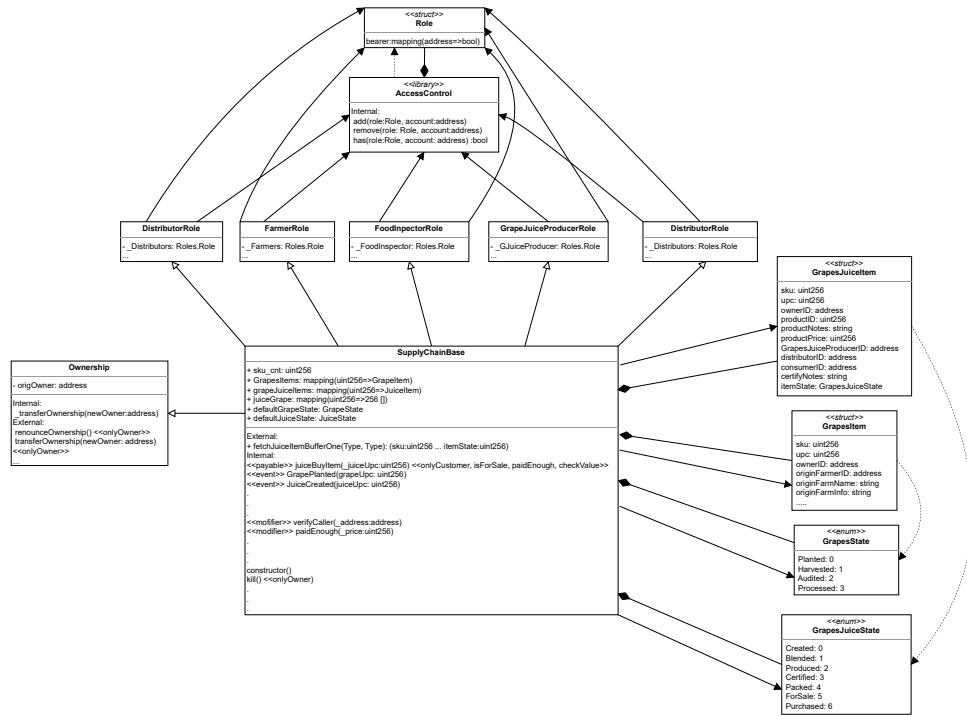


Figure 5.4: UML Class diagram, based on .sol files of smart contracts of Supply Chain Management DApp System of Grapes Juice

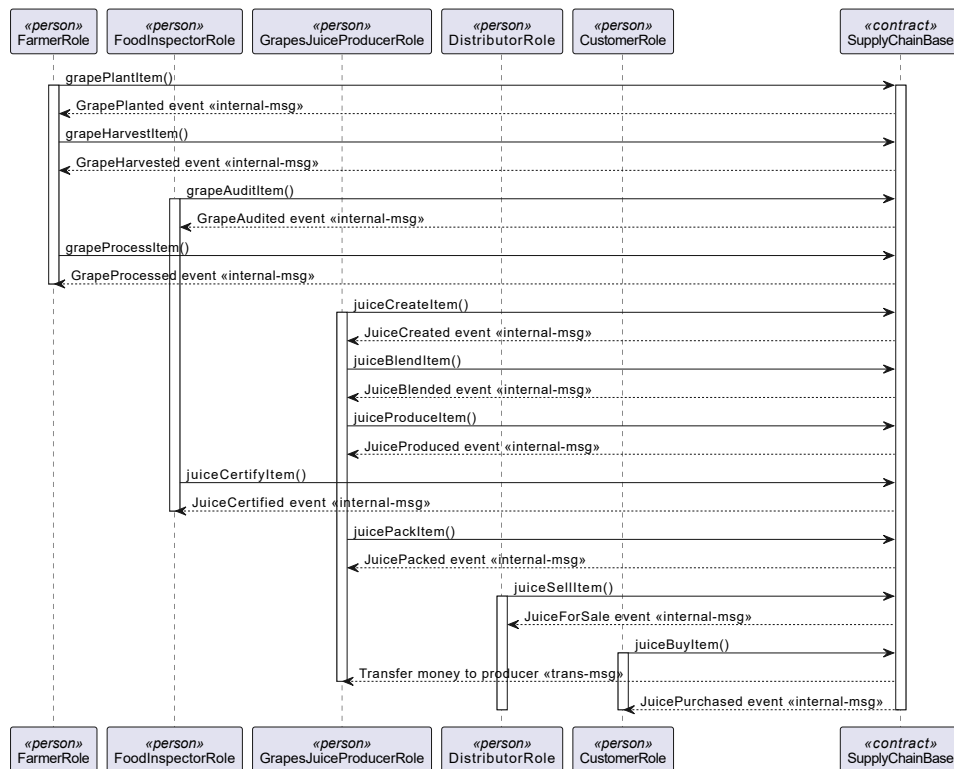


Figure 5.5: Sequence Diagram of the grapes fruit juice supply chain management DApp system

5.7.1.4 Write Series of Acceptance Test Cases

After the generation of test cases a complete implementation process was performed, this process was carried out using the pair programming method. Different JavaScript and solidity files are created .JS and .sol extensions.

```

it('can the Farmer harvest a Grape', async function () {
  this.timeout(20000);
  let upc = 1;
  let ownerID = acc_farm_0;
  let originFarmerID = acc_farm_0;
  let harvestNotes = "bordo wine";
  let itemState = 1;
  let harvest = await instance.grapeHarvestItem(upc, harvestNotes, { from:
    acc_farm_0 });
  let res1 = await instance.fetchGrapeItemBufferOne.call(upc);
  let res2 = await instance.fetchGrapeItemBufferTwo.call(upc);
  assert.equal(res1.upc, upc, 'Error: Invalid item UPC');
  assert.equal(res1.ownerID, ownerID, 'Error: Missing or Invalid ownerID');
  assert.equal(res1.originFarmerID, originFarmerID, 'Error: Missing or Invalid
    originFarmerID');
  assert.equal(res2.harvestNotes, harvestNotes, 'Error: Missing or Invalid
    harvestNotes');
  assert.equal(res2.itemState, itemState, 'Error: Invalid item State');
  truffleAssert.eventEmitted(harvest, 'GrapeHarvested');
});

it('can the Inspector audit a Grape', async function () {
  this.timeout(20000);
  let upc = 1;
  let ownerID = acc_farm_0;
  let originFarmerID = acc_farm_0;
  let auditNotes = "ISO9002 audit passed";
  let itemState = 2;
  let audited = await instance.grapeAuditItem(upc, auditNotes, { from: acc_insp_0 });
  let res1 = await instance.fetchGrapeItemBufferOne.call(upc);
  let res2 = await instance.fetchGrapeItemBufferTwo.call(upc);
  assert.equal(res1.upc, upc, 'Error: Invalid item UPC');
  assert.equal(res1.ownerID, ownerID, 'Error: Missing or Invalid ownerID');

```

```

    assert.equal(res1.originFarmerID, originFarmerID, 'Error: Missing or Invalid
      originFarmerID');
    assert.equal(res2.auditNotes, auditNotes, 'Error: Missing or Invalid
      auditNotes');
    assert.equal(res2.itemState, itemState, 'Error: Invalid item State');
    truffleAssert.eventEmitted(audited, 'GrapeAudited');
  });
it('can the Inspector certify a Juice', async function () {
  this.timeout(20000);
  let juiceUpc = 1;
  let certifyNotes = "ISO9002 Certified";
  let ownerID = acc_prod_0;
  let itemState = 3;
  let certified = await instance.juiceCertifyItem(juiceUpc, certifyNotes, { from:
    acc_insp_0 });
  let res1 = await instance.fetchJuiceItemBufferOne.call(juiceUpc);
  assert.equal(res1.upc, juiceUpc, 'Error: Invalid item UPC');
  assert.equal(res1.ownerID, ownerID, 'Error: Missing or Invalid ownerID');
  assert.equal(res1.certifyNotes, certifyNotes, 'Error: Missing or Invalid
    certifyNotes');
  assert.equal(res1.itemState, itemState, 'Error: Invalid item State');
  truffleAssert.eventEmitted(certified, 'JuiceCertified');
});

```

5.7.1.5 Implementation and Evaluation

After the generation of test cases a complete implementation process was performed, this process was carried out using the pair programming method. Different JavaScript and solidity files are created with .JS and .sol extensions.

```

pragma solidity >=0.6.00;
// Provides basic authorization control
// Contains required functions that establish owner and the transfer of ownership.
contract Ownable {
  address private origOwner;
  // Define an Event
  event TransferOwnership(address indexed oldOwner, address indexed newOwner);

```

```

    /// Assign the contract to an owner
    constructor() public {
        origOwner = msg.sender;
        emit TransferOwnership(address(0), origOwner); }
    /// Look up the address of the owner
    function owner() public view returns (address) {
        return origOwner; }
    /// Check if the calling address is the owner of the contract
    function isOwner() public view returns (bool) {
        return msg.sender == origOwner; }
    /// Define a function modifier 'onlyOwner'
    modifier onlyOwner() {
        require(isOwner());
        _; }
    /// Define a function to renounce ownership
    function renounceOwnership() external onlyOwner {
        emit TransferOwnership(origOwner, address(0));
        origOwner = address(0);}
    /// Define a public function to transfer ownership
    function transferOwnership(address newOwner) external onlyOwner {
        _transferOwnership(newOwner);}
    /// Define an internal function to transfer ownership
    function _transferOwnership(address newOwner) internal {
        require(newOwner != address(0));
        emit TransferOwnership(origOwner, newOwner);
        origOwner = newOwner; }}

pragma solidity >=0.6.00;
// Based on openzeppelin-solidity@2.5.0:
    openzeppelin-solidity\contracts\access\Roles.sol
/**
 * @title Roles
 * @dev Library for managing addresses assigned to a Role.
 */
library Roles {
    struct Role {
        mapping(address => bool) bearer;}
/**

```

```

    * @dev Give an account access to this role.
    */
function add(Role storage role, address account) internal {
    require(!has(role, account), "Roles: account already has role");
    role.bearer[account] = true;}
/**
 * @dev Remove an account's access to this role.
 */
function remove(Role storage role, address account) internal {
    require(has(role, account), "Roles: account does not have role");
    role.bearer[account] = false;}
/**
 * @dev Check if an account has this role.
 * @return bool
 */
function has(Role storage role, address account)
    internal
    view
    returns (bool)
{
    require(account != address(0), "Roles: account is the zero address");
    return role.bearer[account];
}}

```

5.7.1.6 ATDD

For Acceptance Test Driven Development (ATDD) a review meeting was arranged of the development team with project representatives who can verify and validate the developed code, under the prioritized user stories and the results of test cases performed earlier. A signal or reminder was sent with the acceptance test-driven development meeting to the planning team for a check on the time taken and estimated time of project iterations.

5.7.2 App System Development

Release Planning, CRC modeling, UML modeling, writing a series of Test cases, Implementation, and Evaluation with ATDD are the same steps in App system development that have been

followed and carried out in the Smart Contract Development sub-system. The steps that are different from the smart contract development sub-system phase are discussed as follows:

5.7.2.1 UI, Design Modules and Messages

For user Interface design front end JavaScript based technology i.e., Vue.JS with HTML, CSS, and JavaScript. Screenshots of the front end through which the end user will interact with the grapes juice supply chain DApp system are shown in Figure 5.6 and 5.7:

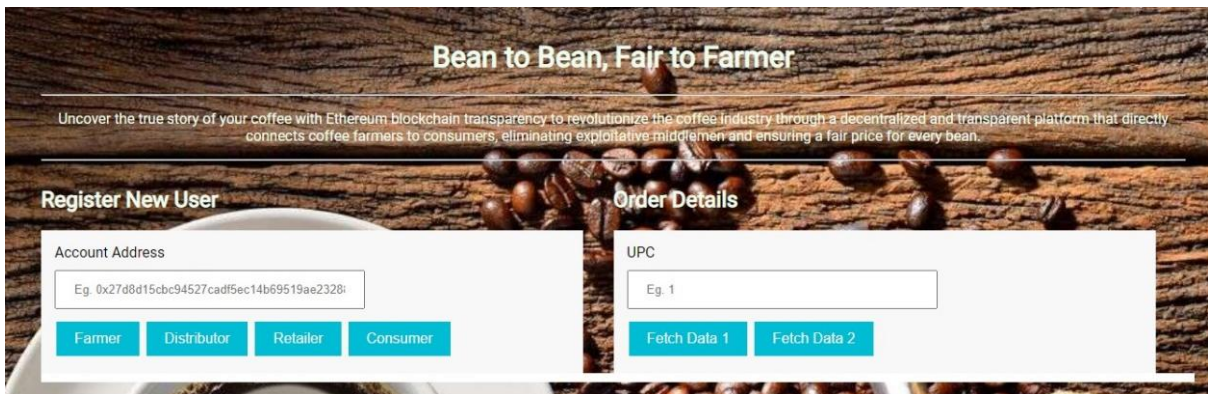


Figure 5.6: User Interface of Grapes Fruit Supply chain DApp System - Register New User with Order Details

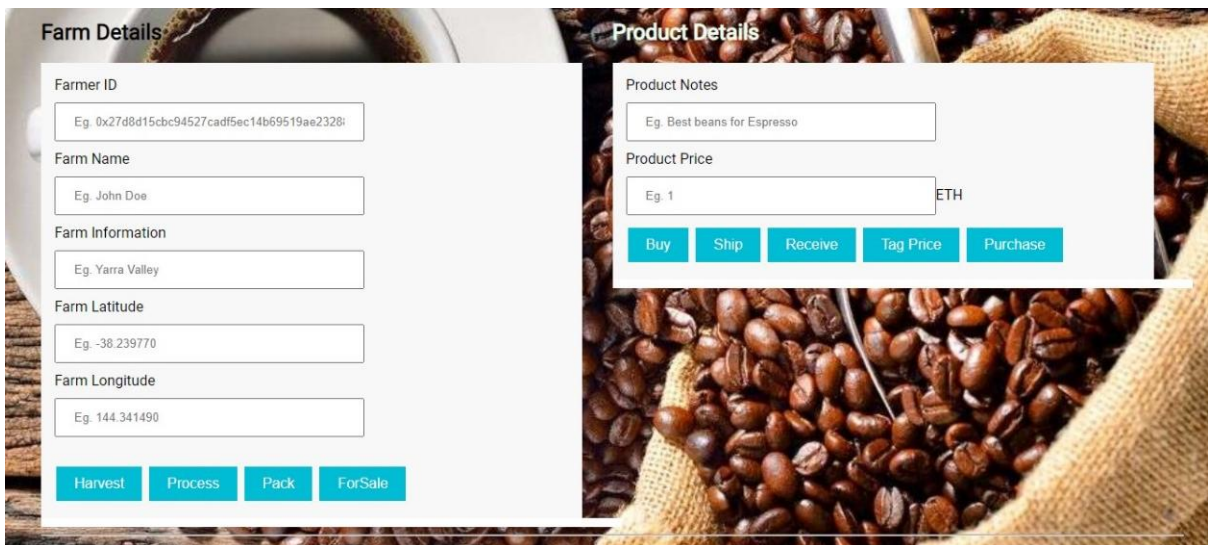


Figure 5.7: User Interface of Grapes Fruit Supply chain DApp System – Farm and Product Details

5.7.2.2 System Integration

System integration is performed by keeping in mind the complexity of smart contracts with front-end modules of the DApp system. The connection of the right module with the right smart contracts is considered a priority.

5.7.2.3 Security Assessment

At several stages in the development phases and sections of the DApp system, different security assessment practices are followed to maintain security aspects in the development process such as OWASP proactive controls.

5.7.2.4 Gas Optimization

For careful and efficient use of ETH, in smart contracts writing, different recommended patterns such as limited modifiers, limit external calls, and event logs, etc., are followed for efficient use of ETH as gas in transactions of the DApp system.

5.8 Maintenance

After system integration small release is initiated and the DApp system is updated with a new release after customer approval in an acceptance testing meeting, after 2 to 3 iterations another SMART analysis is performed to results and compare them with the estimated objectives.

5.9 Case Study Concluding Remarks

In this validation report a supply chain DApp system for grapes juice life cycle from production to sales and distribution is developed, using Ethereum-based smart contracts. In this whole process a proposed SMART agile-based framework is followed, every phase of development is carried out using the proposed framework and the result of using the proposed framework is satisfactory for efficient and quality-oriented development of DApps, some of the noticeable recommendations that are quite helpful in the development lifecycle of DApps for efficiency and quality are as follows:

- The recommended SMART analysis of project goals is specific which prevents the misunderstandings created by the short goals statements.
- Prioritizing the user stories is used in informal and formal UML modeling and the creation of test cases.
- Informal or CRC modeling before formal UML modeling is effective in eliminating the complexity of formal UML class, sequence, or other types of UML diagrams.
- Recommended Pair programming reduces the cost of debugging at the time of deployment and improves the quality of code with continuous reviews.
- Notify project velocity to the planning team after every iteration to support the time estimation for the next iterations.

5.10 Summary

Complete implementation details of the proposed framework are described in this chapter. Vision, Design and implementation, and Deployment sections and their relevant phases and development of sub-systems are discussed in this chapter on practical experience.

Results and Analysis

6.1 Introduction

The results and Analysis chapter of this thesis document is an integral part because it consists of two important and decisive elements of the thesis “Results” and “Analysis”. The results are all about the findings of the experimental implementation and validation process. Whereas analysis is the observations on the findings of the validation process represented through visual presentations like charts. This chapter possesses results and analysis parts based on the findings of the questionnaire survey validation process in which analysis is carried out using charts or graphs generated based on the collected data in the survey and the second one is a complete validation report generated by software house based on a case study. A comparison of the previous studies with the results and analysis of the proposed framework is also drawn in this section.

6.2 Analysis of Questionnaire Survey

A questionnaire survey is conducted to validate the claims of the proposed framework for efficient and quality-oriented development of DApps. In this survey, different types of questions are shared with professionals of the software industry, who have sound knowledge of blockchain and its related technologies such as smart contracts-based DApps and its applications in several business domains i.e., finance, gaming industry, and supply chain, front and backend development. Industry professionals shared responses are valuable insights to analyze how the proposed framework can be effective in providing simplicity, agile testing, and efficient and

quality-oriented development of DApps.

In this analysis chapter, insights from responses of software industry professionals are analyzed to draw valuable conclusions as follows:

6.2.1 Challenges in Agile SDLC of DApps

The main challenges identified by software industry professionals in response to the survey are Unspecified goals, Delays in feedback to iterations, Simplicity in design, requirements, and other phases of SDLC, Delay in response to bug detection, Inefficient consumption of resources, Lack of practices for quality-oriented development, and continuously changing and evolving requirements. These challenges ignite the need for software development standards to mitigate these identified challenges, graphically illustrated in Figure 6.1.

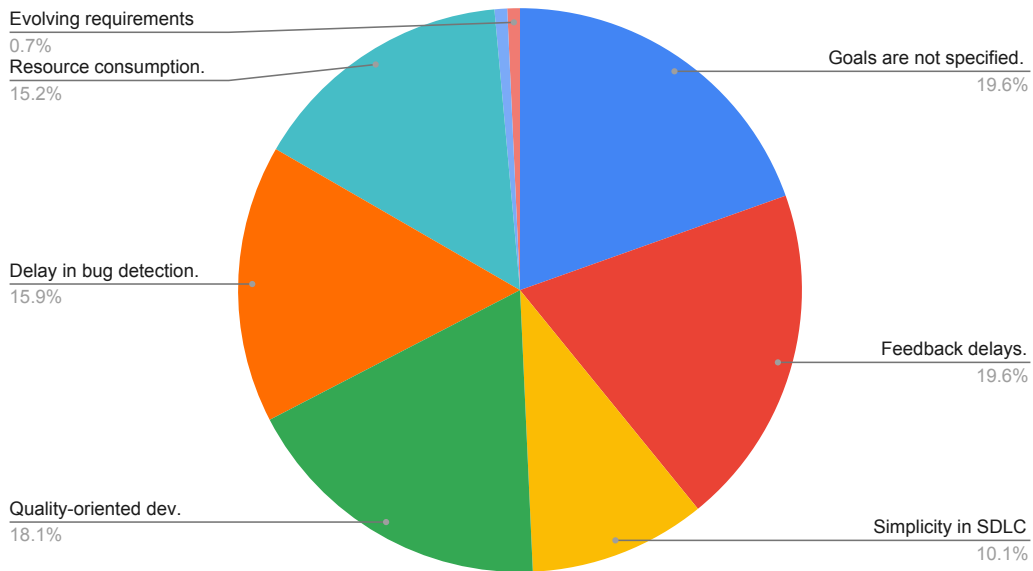


Figure 6.1: Identified Challenges in Agile SDLC of DApps

6.2.2 In practice Agile Methodologies for the Development of DApps

The most in-practice agile methodologies and practices for the development of DApps and other traditional DApps are Scrum, XP, Kanban, Feature Driven Development (FDD), Lean Software Development (LSD), and Crystal, as shown in Figure 6.2. It is clear in Figure 6.2 that most of the agile methodologies are not single but in groups of 2 or 3 and have a hybrid approach to practice agile methodologies in SDLC.

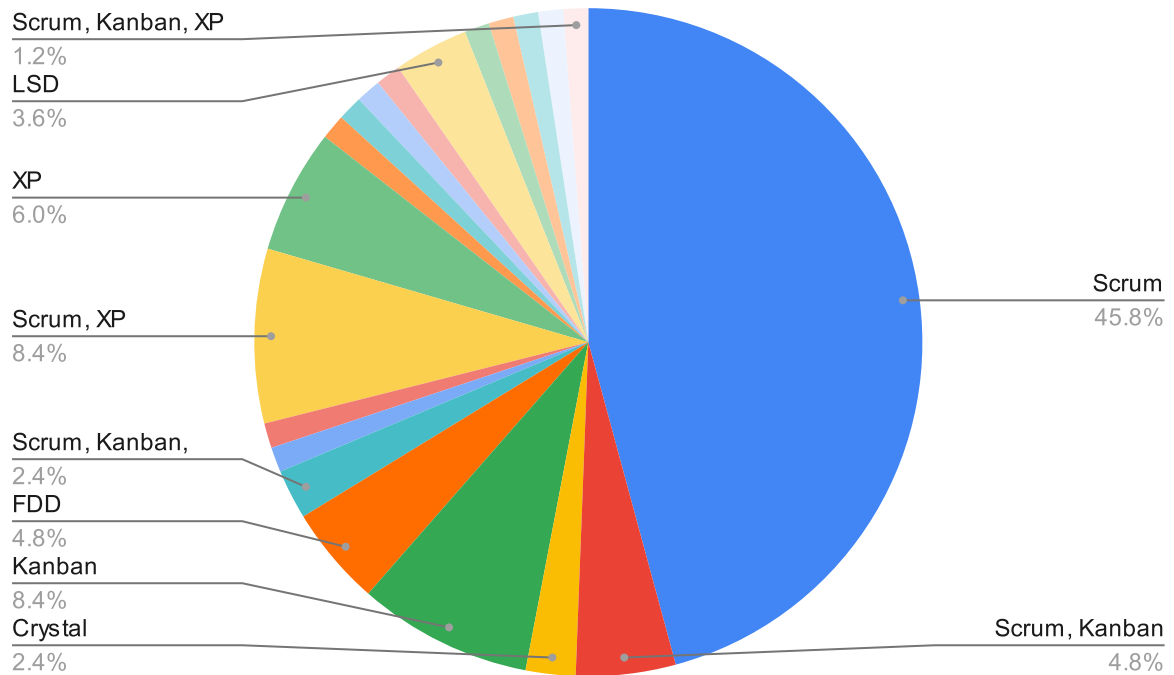


Figure 6.2: In practice Agile Methodologies for the Development of DApps and Software

6.2.3 Requirements Prioritization in the Development of DApps

Software industry professionals' response indicates that the practice of requirements prioritization in the development of DApps is essential to comprehend them. The average value of survey responses is 3.93* out of 5.0, with a Standard Deviation (SD) value of 0.84. These values support the practice of requirements prioritization in SDLC of DApps, the complete graph is shown in Figure 6.3.

Moreover, requirements prioritization is an effective approach for efficient agile-based development of DApps, this statement is also validated through the survey with an average value of 4.28* out of 5.0, with an SD value of 0.83, a complete graphical representation of these responses is in figure 6.4.

*Value 1 (strongly disagree/not effective/not essential/not crucial) -to- Value 5 (strongly agree/highly effective/highly essential/highly crucial).

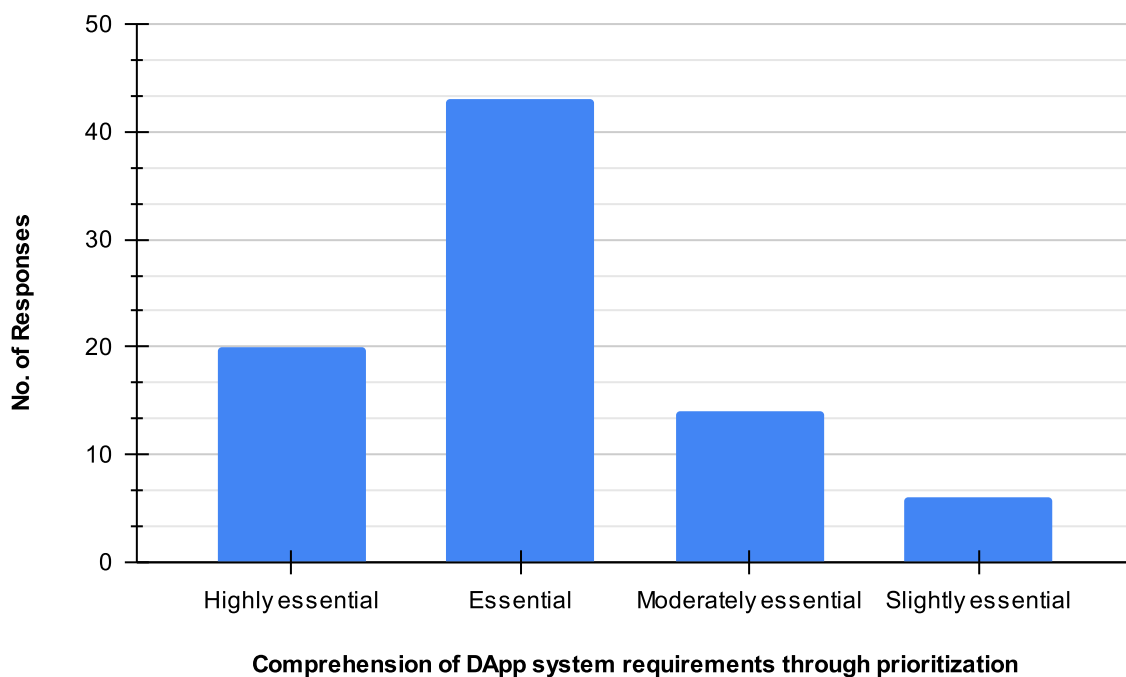


Figure 6.3: Results of how much essential prioritization of requirements in SDLC of DApps

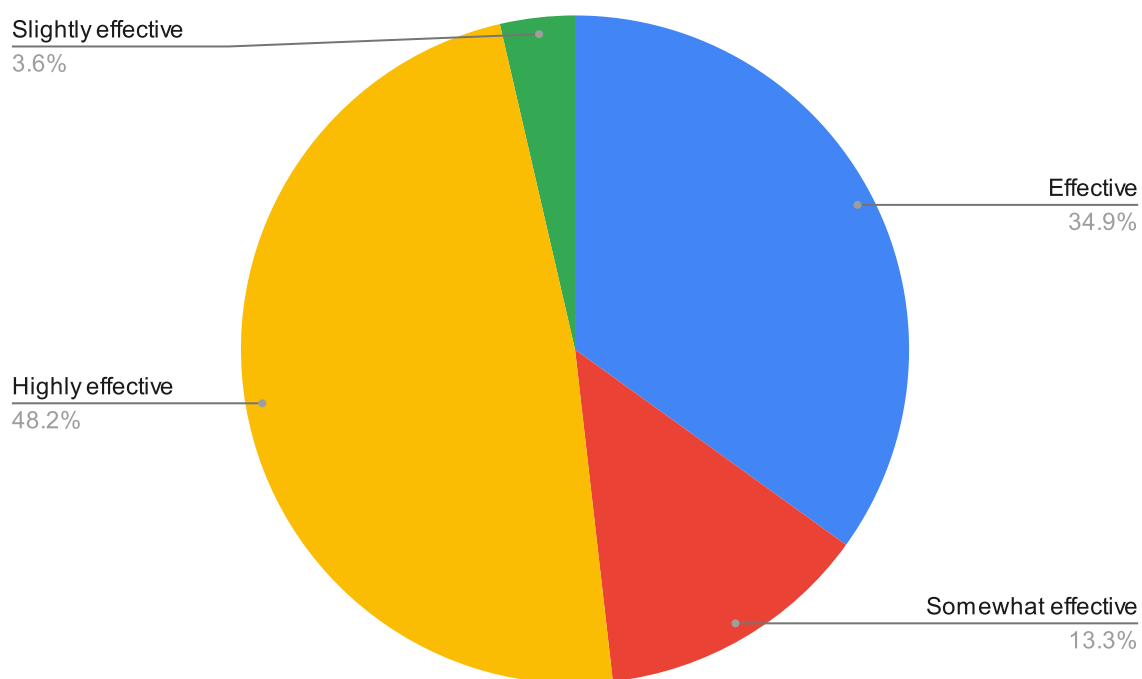


Figure 6.4: Graphical representation of survey results to show the effectiveness of prioritization of requirements for efficient agile development of DApps

6.2.4 Define Project goals using SMART Analysis

Using SMART analysis and objectives to define and specify the goals of the project is an effective approach in the agile development of DApps, identified through survey responses of software industry professionals. The average value of responses to this statement of the question is 4.01* out of 5.0, with an SD value of 0.79. These responses and values are motivation to define and specify project goals using the practice of SMART analysis, a complete analytical graph is shown in Figure 6.5.

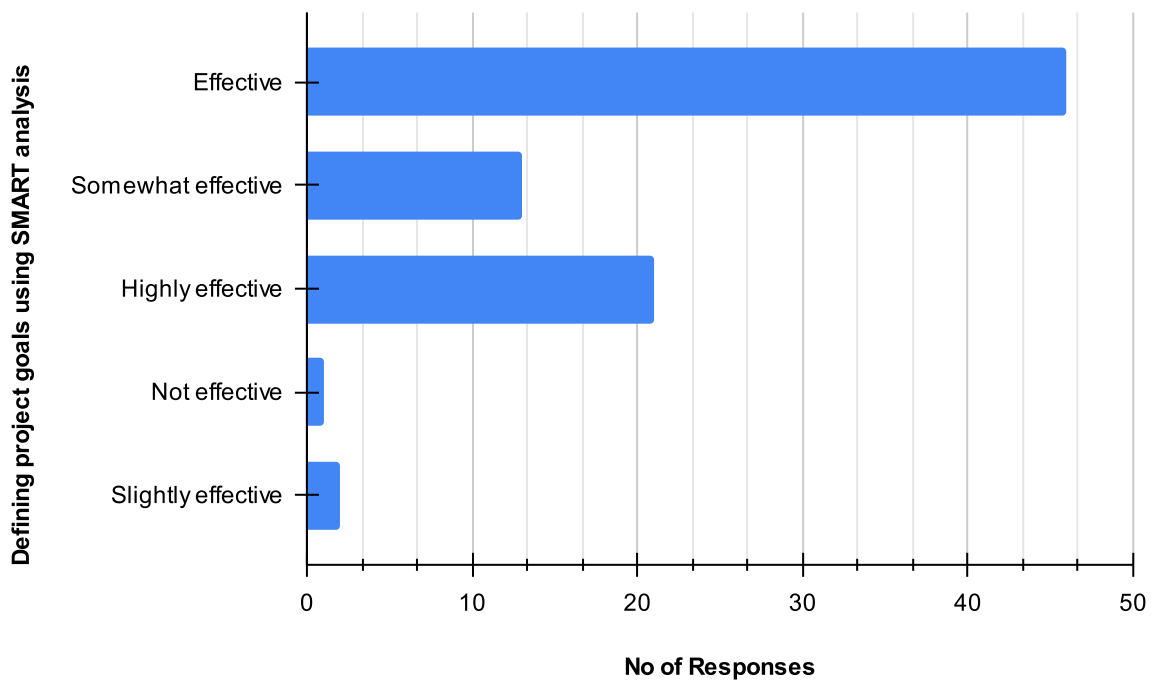


Figure 6.5: Results to validate the effectiveness of defining and specifying goals of the project using SMART analysis

6.2.5 Use of CRC Cards in the Design Phase of Agile DApp Development

The use of the CRC cards modeling approach for better comprehension of the use of stories-based requirements to design complex UML diagrams such as class diagrams in agile development of DApps, is identified as an effective approach with the support of results and analysis of survey responses. The average value of these responses is 3.92* out of 5.0, with an SD value of 0.81, a complete graphical analysis of survey responses is shown in Figure 6.6.

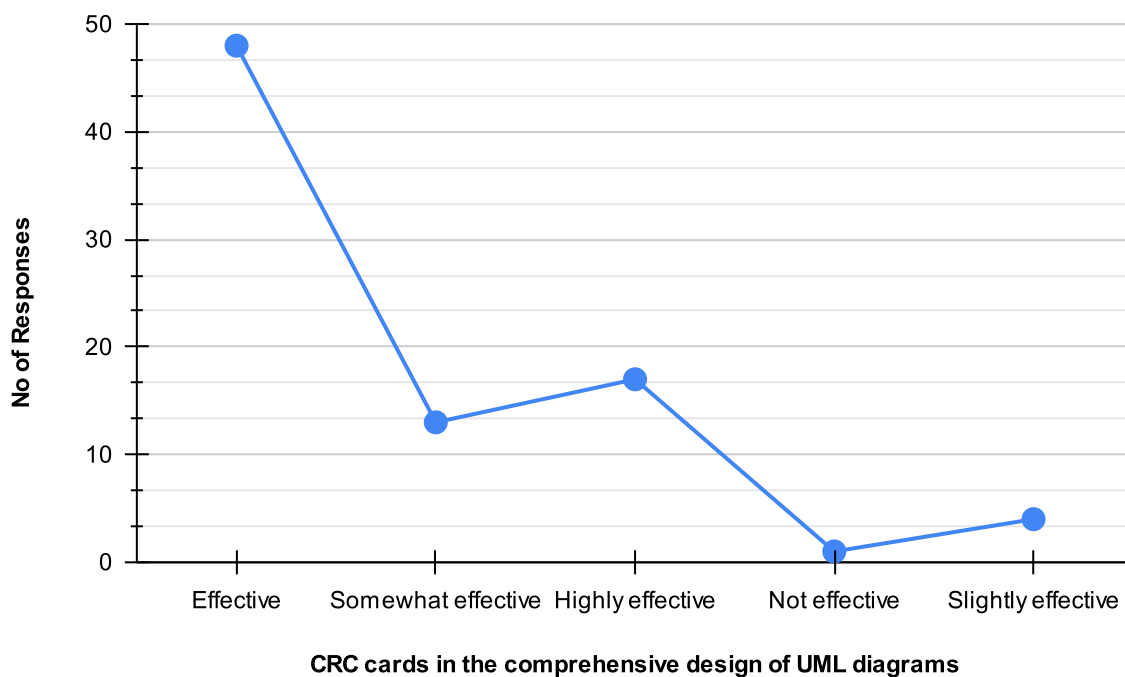


Figure 6.6: Analytical graph of survey responses to verify the effectiveness of CRC cards in the comprehension of UML diagrams in the agile development of DApps

6.2.6 Influence of Pair Programming in Code Quality DApp Systems

Responses of software industry professionals validate that code quality can be influenced by pair programming practice of XP methodology in the agile development of DApps. An analytical graph of responses is shown in Figure 6.7, to assure the validity of this question statement.

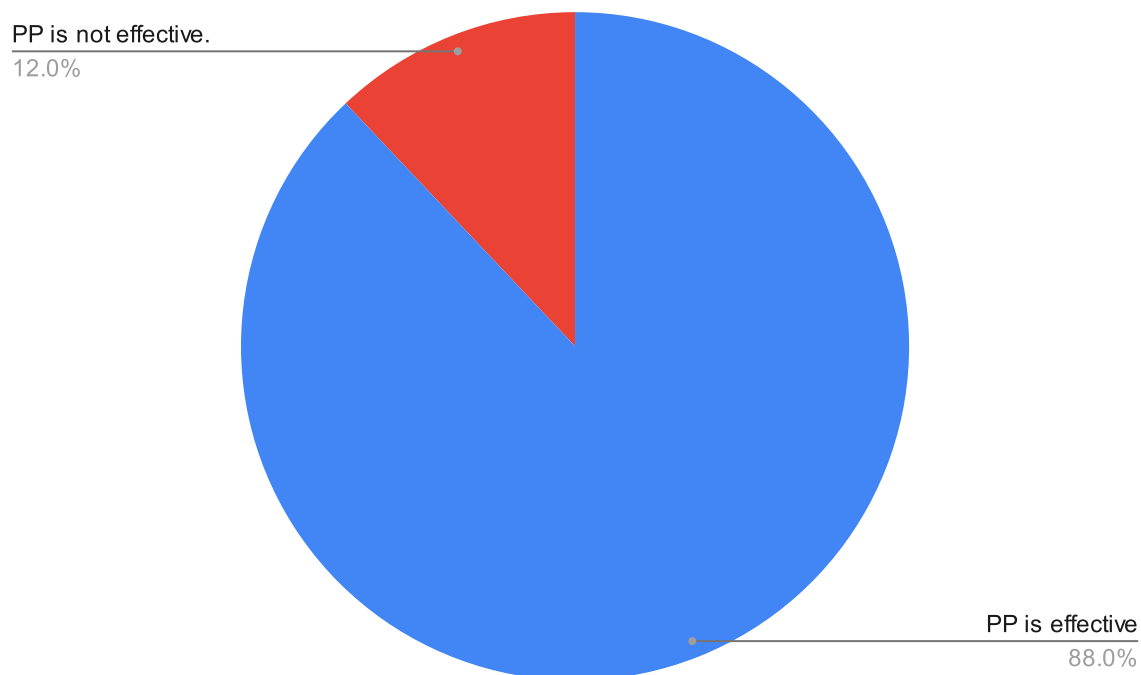


Figure 6.7: Analytical graph to validate the influence of pair programming in code quality of agile development of DApps.

6.2.7 ATDD to Validate and Acceptance Phase in Agile Development of DApps

The practice of ATDD to enhance the validation and acceptance criteria is a recognized approach in the agile development of DApps, validated through the results and analysis of responses to surveys shared with software industry professionals. The average values of responses are 4.08* out of 5.0, with an SD value of 0.72, a complete analysis of this question statement in graphical representation is shown in Figure 6.8.

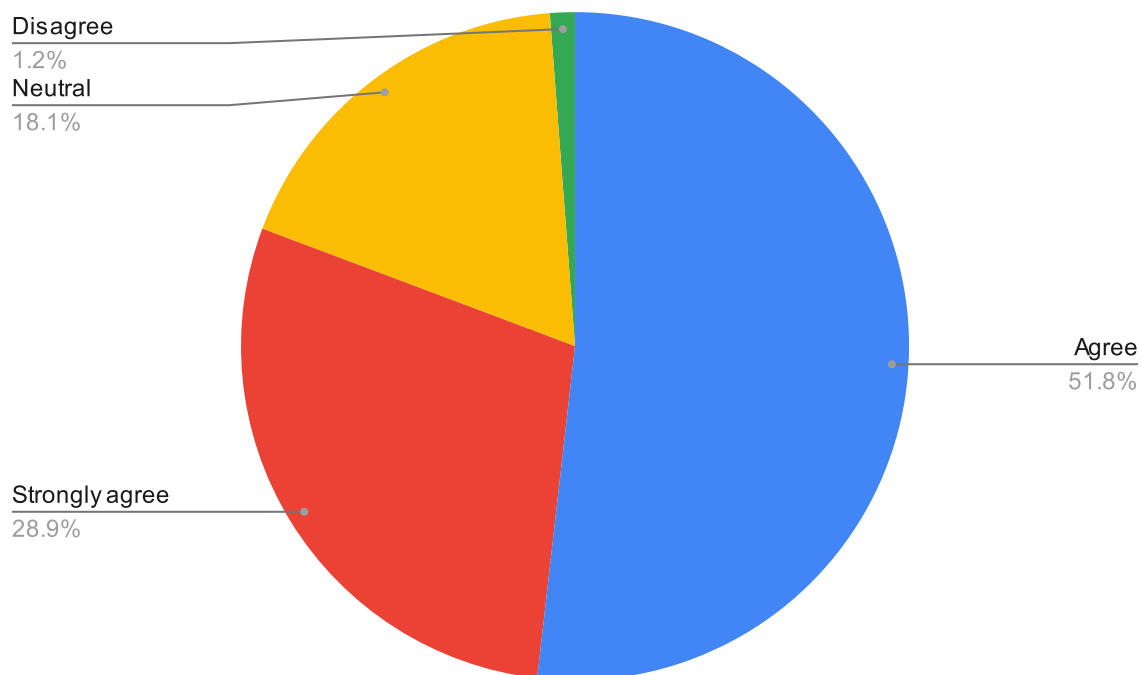


Figure 6.8: Analytical graph to verify that ATDD enhances the validation and acceptance in agile development of DApps.

6.2.8 Simple Design Leads to Efficient Design and Requirements Elicitation in Agile Development of DApps

Survey responses and insights from software industry professionals agree and encourage the use of a simple design that leads to efficient design and requirements elicitation. The average response rate or value on this statement is 4.02* out of 5.0, with an SD value of 0.87, a complete graphical representation of survey responses is shown in Figure 6.9.

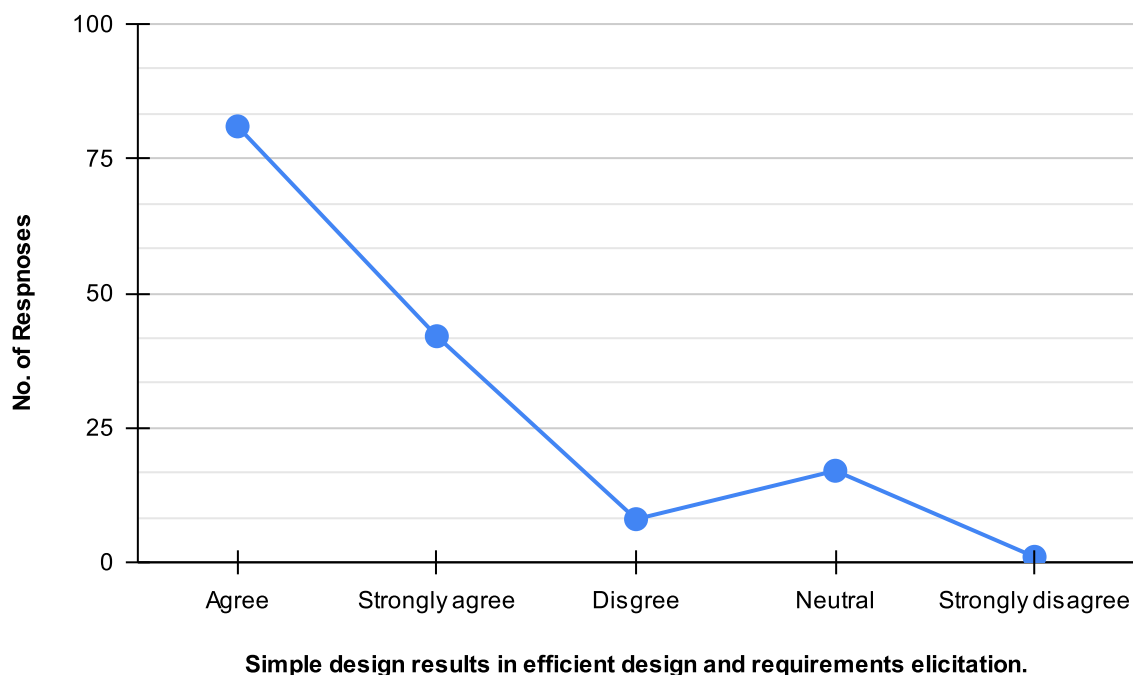


Figure 6.9: Graphical representation of responses validate that simple design leads to efficient design and requirements elicitation.

6.2.9 Agile Practices for Quality-oriented Development of DApps

Survey response results identify that Testing practices i.e., TDD and ATDD, CRC modeling, Prioritization of requirements, and small and incremental releases are crucial agile practices for quality-oriented development of DApps. A graphical representation of the survey responses is shown in Figure 6.10, to analyze the survey responses on the agile practices for quality-oriented development of DApps.

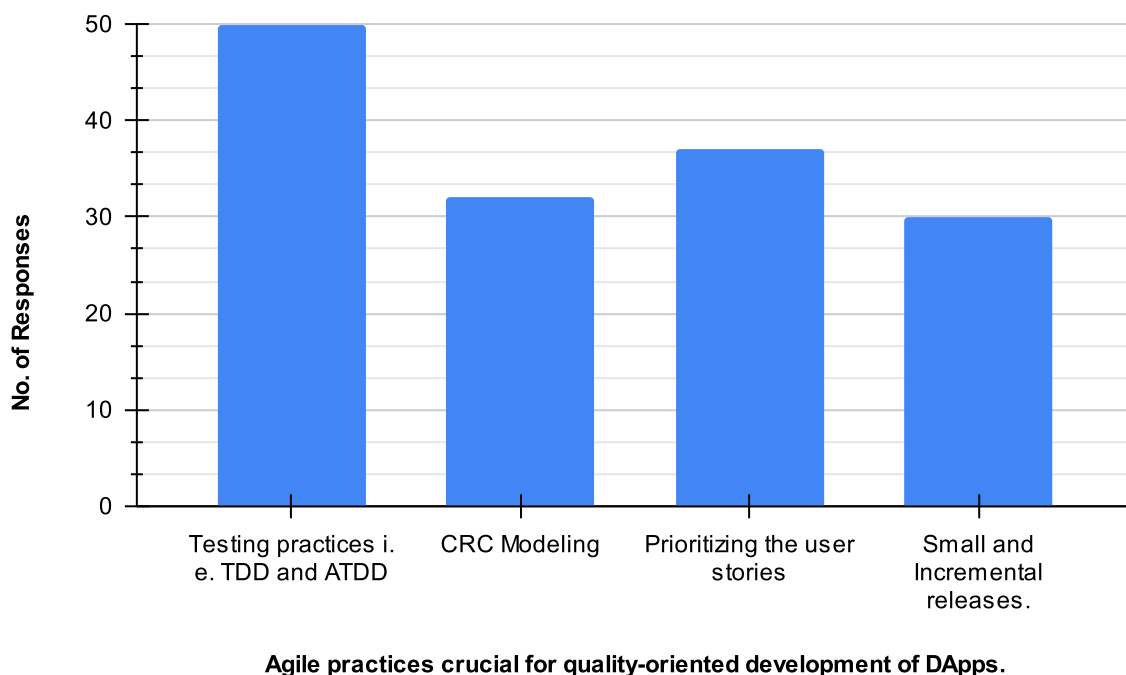


Figure 6.10: Identified and validated through survey crucial agile practices for quality-oriented development of DApps.

6.3 Central Tendencies of Survey Responses on Statements of Questions

Different questions are shared in questionnaire survey form to validate the proposed framework, each question received different responses after statistical analysis, and values are collected of respective question statements and shown in table 6.1.

| <i>Questionnaire Statements</i> | Central tendencies | | | | |
|--|---------------------------|--------------------------------|------------|------------|-------------|
| | <i>Average</i> | <i>Standard Deviation (SD)</i> | <i>Min</i> | <i>Max</i> | <i>Mode</i> |
| 1. Define project goals using SMART analysis are efficient. | 4.01 | 0.79 | 1 | 5 | 4 |
| 2. Prioritization of DApp system requirements is essential for comprehension. | 3.93 | 0.84 | 2 | 5 | 4 |
| 3. CRC cards in the comprehension of UML diagrams are efficient. | 3.92 | 0.81 | 1 | 5 | 4 |
| 4. Review meetings conducted at the end of each iteration in the Agile are crucial. | 3.94 | 1.11 | 1 | 5 | 4 |
| 5. Acceptance Test Driven Development (ATDD) enhances the validation and acceptance criteria. | 4.08 | 0.72 | 2 | 5 | 4 |
| 6. Prioritizing user story-based requirements provides efficient development of DApps. | 4.28 | 0.72 | 2 | 5 | 4 |
| 7. SMART objectives for efficient project goals specification are comprehensive. | 3.11 | 0.56 | 1 | 4 | 3 |
| 8. A simple design method can result in the efficient design and requirements elicitation phase of the development of DApps. | 4.02 | 0.87 | 1 | 5 | 4 |

Table 6.1: Central tendencies of survey responses on statements of questions

6.4 Comparison of Proposed Framework with Related Studies

Several research studies and proposed frameworks on BOSE and agile methodologies for the development of DApps are analyzed and a comparison is drawn with the proposed framework to

ensure the validity. Results of this comparative analysis are shown in Table 6.2, where proposed BOSE approaches and attributes of the proposed framework are compared with related studies already carried out in this area of research.

| <i>BOSE Framework</i> | BOSE Approaches and Attributes | | | | | | | | |
|--|---------------------------------------|--------------------------------|-------------------------|------------------------|-----------------------|-------------------------|-----------------------------|-------------|----------------------------|
| | <i>CRC Modeling</i> | <i>Prioritize User Stories</i> | <i>Pair Programming</i> | <i>Review Meetings</i> | <i>SMART Analysis</i> | <i>Project Velocity</i> | <i>Resource Consumption</i> | <i>ATDD</i> | <i>Security Assessment</i> |
| AgilePlus [22] | X | ✓ | X | ✓ | X | X | ✓ | X | ✓ |
| ABCDE [4] [48] | X | X | ✓ | ✓ | X | X | ✓ | X | ✓ |
| Test Oriented Enticement (TOE) [64] [48] | X | ✓ | X | X | X | X | X | X | X |
| XP and Spike Solution (SS) [65] [48] | X | X | ✓ | X | X | X | X | X | ✓ |
| The Proposed Framework | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

Table 6.2: Comparison of the proposed framework with related studies in this area of research

6.5 Summary

In this chapter, the proposed framework is analyzed with the help of results obtained from a questionnaire survey and case study conducted in collaboration with partners in the software industry. All results of the questionnaire survey are represented graphically and analyzed statistically, whereas a detailed report generated with the help of a partner software house is also included in this chapter. In the end, a comparison of the proposed framework is also drawn with related studies in this area of research.

Conclusion and Future Work

7.1 Conclusion

Quality and efficient use of resources is always crucial in the traditional software development life cycle, in blockchain-enabled smart contracts-based DApps systems both concerns have become top priorities. BOSE provides solutions to these challenges, in BOSE software engineering approaches are proposed and practiced for efficient and quality-oriented development of DApps. In this research, an agile-based hybrid framework, with SMART analysis of project goals, has been introduced. In this framework, appropriate and effective agile-based approaches are used for simplicity and agile testing that leads to efficient resource consumption with quality-oriented development of DApps. The proposed framework is also introduced to software industry professionals in two ways i.e., questionnaire survey and case study implementation, to validate the approaches that have been proposed in this research. The analysis of the results of the questionnaire survey identified that all proposed approaches are effective in the efficient and quality-oriented development of DApps. All responses are statistically analyzed and a good average of responses from software industry professionals identified that the proposed framework follows agile-based approaches for efficient resource consumption and management with quality. Besides this questionnaire survey, a case study for the development of a supply chain management DApps also validates the practical implementations of the proposed framework. A comparison of the proposed framework with practice frameworks from related literature verifies the exclusiveness of this research.

7.2 Future Work

A hybrid agile framework with SMART analysis of project goals will help to mitigate the challenges of efficient resource consumption and management with quality-oriented development of DApps. Development of AI-enabled tools for integration of smart contracts with App system in DApps and for SMART analysis and comparison of project goals to streamline the changing and evolving requirements with project goal statements can improve the efficiency of DApps development with quality. In this global environment, where project teams are located in different geographical locations risks of improper requirements elicitation, on-time feedback, and customer collaborations with delays in bug detection and response these factors can lead to inefficient resource consumption, especially in blockchain-enabled smart contracts-based DApps where resources are costly, are open areas of research. Furthermore, the practice of the proposed framework in this research in collaboration with industry will refine the framework.

References

- [1] Kimani, D., Adams, K., Attah-Boakye, R., Ullah, S., Frecknall-Hughes, J., & Kim, J. (2020). Blockchain, business and the fourth industrial revolution: Whence, whither, wherefore and how? *Technological Forecasting and Social Change*, 161, 120254.
- [2] Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System.
- [3] Wood, D.D. (2014). ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER.
- [4] Marchesi, L., Marchesi, M., & Tonelli, R. (2019). ABCDE - Agile Block Chain Dapp Engineering. ArXiv, abs/1912.09074.
- [5] Tikhomirov, S. (2017). Ethereum: State of Knowledge and Research Perspectives. *Foundations and Practice of Security*.
- [6] Fenu, G., Marchesi, L., Marchesi, M., & Tonelli, R. (2018). The ICO phenomenon and its relationships with ethereum smart contract environment. *2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE)*, 26-32.
- [7] Mahmoud, N., Aly, A.A., & Abdelkader, H. (2022). Enhancing Blockchain-based Ride-Sharing Services using IPFS. *Intell. Syst. Appl.*, 16, 200135.
- [8] Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., ... & Thomas, D. (2001). Manifesto for agile software development.
- [9] Santos, W. (2016, June). Towards a better understanding of simplicity in agile software development projects. In *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering* (pp. 1-4).
- [10] Cao, L., Mohan, K., Xu, P., & Ramesh, B. (2004, January). How extreme does extreme programming have to be? Adapting XP practices to large-scale projects. In *37th Annual Hawaii International Conference on System Sciences*, 2004. *Proceedings of the* (pp. 10-pp). IEEE.

REFERENCES

- [11] Cauevic, A., Punnekkat, S., & Sundmark, D. (2012, September). Quality of testing in test driven development. In 2012 Eighth International Conference on the Quality of Information and Communications Technology (pp. 266-271). IEEE.
- [12] Burris, J. W. (2017, April). Test-Driven Development for Parallel Applications. In 2017 Second International Conference on Information Systems Engineering (ICISE) (pp. 27-31). IEEE.
- [13] Guerra, E. (2012, October). Fundamental test driven development step patterns. In Proceedings of the 19th Conference on Pattern Languages of Programs (pp. 1-15).
- [14] Brataas, G., Hanssen, G. K., & Ræder, G. (2018). Towards agile scalability engineering. In Agile Processes in Software Engineering and Extreme Programming: 19th International Conference, XP 2018, Porto, Portugal, May 21–25, 2018, Proceedings 19 (pp. 248-255). Springer International Publishing.
- [15] Santos, W. B., Cunha, J. A. O., Moura, H., & Margaria, T. (2017, August). Towards a theory of simplicity in agile software development: A qualitative study. In 2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA) (pp. 40-43). IEEE.
- [16] Khan, R., Srivastava, A. K., & Pandey, D. (2016, November). Agile approach for Software Testing process. In 2016 International Conference System Modeling & Advancement in Research Trends (SMART) (pp. 3-6). IEEE.
- [17] Gușeală, L. G., Bratu, D. V., & Moraru, S. A. (2019, August). Continuous testing in the development of iot applications. In 2019 International Conference on Sensing and Instrumentation in IoT Era (ISSI) (pp. 1-6). IEEE.
- [18] Hamsini, R., & Smitha, G. R. Agile Development Methodology and Testing for Mobile Applications-A Survey. *International Journal of New Technology and Research*, 2(9), 263424.
- [19] Talby, D., Keren, A., Hazzan, O., & Dubinsky, Y. (2006). Agile software testing in a large-scale project. *IEEE software*, 23(4), 30-37.
- [20] Leffingwell, D. (2010). *Agile software requirements: lean requirements practices for teams, programs, and the enterprise*. Addison-Wesley Professional.
- [21] Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2017). Agile software development methods: Review and analysis. *arXiv preprint arXiv:1709.08439*.
- [22] Farooq, M. S., Kalim, Z., Qureshi, J. N., Rasheed, S., & Abid, A. (2022). A blockchain-based framework for distributed agile software development. *IEEE Access*, 10, 17977-17995.

REFERENCES

- [23] Nofer, M., Gomber, P., Hinz, O., & Schiereck, D. (2017). Blockchain. *Business & Information Systems Engineering*, 59, 183-187.
- [24] Natarajan, H., Krause, S., & Gradstein, H. (2017). Distributed ledger technology and blockchain.
- [25] Wang, W., Lumineau, F., & Schilke, O. (2022). *Blockchains: Strategic implications for contracting, trust, and organizational design*. Cambridge University Press.
- [26] Capocasale, V., & Perboli, G. (2022). Standardizing smart contracts. *IEEE Access*, 10, 91203-91212.
- [27] Zheng, Z., Xie, S., Dai, H. N., Chen, W., Chen, X., Weng, J., & Imran, M. (2020). An overview on smart contracts: Challenges, advances and platforms. *Future Generation Computer Systems*, 105, 475-491.
- [28] Antonopoulos, A. M., & Wood, G. *Mastering Ethereum: Building Smart Contracts and Dapps*.
- [29] Zheng, P., Jiang, Z., Wu, J., & Zheng, Z. (2023). Blockchain-based Decentralized Application: A Survey. *IEEE Open Journal of the Computer Society*.
- [30] Akhtar, A., Bakhtawar, B., & Akhtar, S. (2022). Extreme Programming Vs Scrum: A Comparison Of Agile Models. *International Journal of Technology, Innovation and Management (IJTIM)*, 2(2), 80-96.
- [31] Santos, W. B. (2018). Simplicity in agile software development.
- [32] Crispin, L., & Gregory, J. (2009). *Agile testing: A practical guide for testers and agile teams*. Pearson Education.
- [33] Puleio, M. (2006, July). How not to do agile testing. In *AGILE 2006 (AGILE'06)* (pp. 7-pp). IEEE.
- [34] Mohanty, H., Mohanty, J. R., & Balakrishnan, A. (Eds.). (2017). *Trends in software testing*. Springer Singapore.
- [35] Ibba, S. (2019). Agile methodologies and blockchain development.
- [36] Malik, M. U., Chaudhry, N. M., & Malik, K. S. (2013). Evaluation of efficient requirement engineering techniques in agile software development. *International Journal of Computer Applications*, 83(3).

REFERENCES

- [37] Butt, S. A., Misra, S., Anjum, M. W., & Hassan, S. A. (2021). Agile project development issues during COVID-19. In *Lean and Agile Software Development: 5th International Conference, LASD 2021, Virtual Event, January 23, 2021, Proceedings 5* (pp. 59-70). Springer International Publishing.
- [38] Lee, G., & Xia, W. (2010). Toward agile: an integrated analysis of quantitative and qualitative field data on software development agility. *MIS quarterly*, 34(1), 87-114.
- [39] Amler, H., Eckey, L., Faust, S., Kaiser, M., Sandner, P., & Schlosser, B. (2021, September). Defi-ning defi: Challenges & pathway. In *2021 3rd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS)* (pp. 181-184). IEEE.
- [40] Pierro, G. A., & Tonelli, R. (2022, March). Can solana be the solution to the blockchain scalability problem?. In *2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)* (pp. 1219-1226). IEEE.
- [41] Saingre, D., Ledoux, T., & Menaud, J. M. (2022). Measuring performances and footprint of blockchains with BCTMark: a case study on Ethereum smart contracts energy consumption. *Cluster Computing*, 25(4), 2819-2837.
- [42] Alharby, M., & Van Moorsel, A. (2017). Blockchain-based smart contracts: A systematic mapping study. arXiv preprint arXiv:1710.06372.
- [43] Scherer, M. (2017). Performance and scalability of blockchain networks and smart contracts.
- [44] Cheng, M., Chong, H. Y., & Xu, Y. (2023). Blockchain-smart contracts for sustainable project performance: bibliometric and content analyses. *Environment, Development and Sustainability*, 1-24.
- [45] Porru, S., Pinna, A., Marchesi, M., & Tonelli, R. (2017, May). Blockchain-oriented software engineering: challenges and new directions. In *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)* (pp. 169-171). IEEE.
- [46] Faruk, M. J. H., Subramanian, S., Shahriar, H., Valero, M., Li, X., & Tasnim, M. (2022, May). Software engineering process and methodology in blockchain-oriented software development: A systematic study. In *2022 IEEE/ACIS 20th International Conference on Software Engineering Research, Management and Applications (SERA)* (pp. 120-127). IEEE.
- [47] Reddivari, S., & Wilson, A. (2022, August). Blockchain-Oriented Requirements Engineering: New Directions. In *2022 IEEE 23rd International Conference on Information Reuse and Integration for Data Science (IRI)* (pp. 118-123). IEEE.

REFERENCES

- [48] Farooq, M. S., Ahmed, M., & Emran, M. (2022). A survey on blockchain acquainted software requirements engineering: model, opportunities, challenges, and future directions. *IEEE Access*, 10, 48193-48228.
- [49] Hoda, R., Salleh, N., & Grundy, J. (2018). The rise and evolution of agile software development. *IEEE software*, 35(5), 58-63.
- [50] Fahmideh, M., Grundy, J., Ahmad, A., Shen, J., Yan, J., Mougouei, D., ... & Abedin, B. (2022). Engineering Blockchain-based Software Systems: Foundations, Survey, and Future Directions. *ACM Computing Surveys*, 55(6), 1-44.
- [51] SK, P. N. (2022, March). Secure Metadata Curating the Agile Software Development Process and its Need for Blockchain Storage. In *2022 International Conference on Electronics and Renewable Systems (ICEARS)* (pp. 958-963). IEEE.
- [52] Janzen, D., & Saiedian, H. (2005). Test-driven development concepts, taxonomy, and future direction. *Computer*, 38(9), 43-50.
- [53] George, B., & Williams, L. (2004). A structured experiment of test-driven development. *Information and software Technology*, 46(5), 337-342.
- [54] Nanthaamornphong, A., & Carver, J. C. (2017). Test-Driven Development in scientific software: a survey. *Software Quality Journal*, 25, 343-372.
- [55] Spendier, T. (2023). Integration of Web Front-End Testing in an Acceptance Test Automation Framework for Distributed Systems within an Emergency Center Environment (Doctoral dissertation, Wien).
- [56] Afrianto, I., Heryandi, A., Finandhita, A., & Atin, S. (2021). User acceptance test for digital signature application in academic domain to support the covid-19 work from home program. *IJISTECH (International Journal of Information System and Technology)*, 5(3), 270-280.
- [57] Marchesi, M., & Succi, G. (2003). Extreme programming and agile processes in software engineering. *Proceedings of XP*.
- [58] Extreme Programming. (2023). Frequent small releases. Retrieved from <http://www.extremeprogramming.org/rules/releaseoften.html>
- [59] Wells, D. (1999). Project Velocity - Extreme Programming [SNIPPET]. *ExtremeProgramming.org*. Retrieved from <http://www.extremeprogramming.org/velocity.html>

REFERENCES

- [60] Churcher, N., & Cerecke, C. (1996, November). groupCRC: Exploring CSCW support for software engineering. In *Proceedings Sixth Australian Conference on Computer-Human Interaction* (pp. 62-68). IEEE.
- [61] Reenskaug, T., Wold, P., & Lehne, O. A. (1996). *Working with objects: the OOram software engineering method* (pp. I-XXI). Greenwich: Manning.
- [62] Ogbeiwi, O. (2017). Why written objectives need to be really SMART. *British Journal of Healthcare Management*, 23(7), 324-336.
- [63] Bjerke, M. B., & Renger, R. (2017). Being smart about writing SMART objectives. *Evaluation and program planning*, 61, 125-127.
- [64] Yilmaz, M., Tasel, S., Tuzun, E., Gulec, U., O'Connor, R. V., & Clarke, P. M. (2019). Applying blockchain to improve the integrity of the software development process. In *Systems, Software and Services Process Improvement: 26th European Conference, EuroSPI 2019, Edinburgh, UK, September 18–20, 2019, Proceedings 26* (pp. 260-271). Springer International Publishing.
- [65] Marchesi, M., Marchesi, L., & Tonelli, R. (2018, October). An agile software engineering method to design blockchain applications. In *Proceedings of the 14th Central and Eastern European Software Engineering Conference Russia* (pp. 1-8).
- [66] Mannion, M., & Keepence, B. (1995). SMART requirements. *ACM SIGSOFT Software Engineering Notes*, 20(2), 42-47.

Annex A



Military College of Signals, NUST
Department of Computer Software Engineering
MSSE Thesis | Proposed Framework Validation

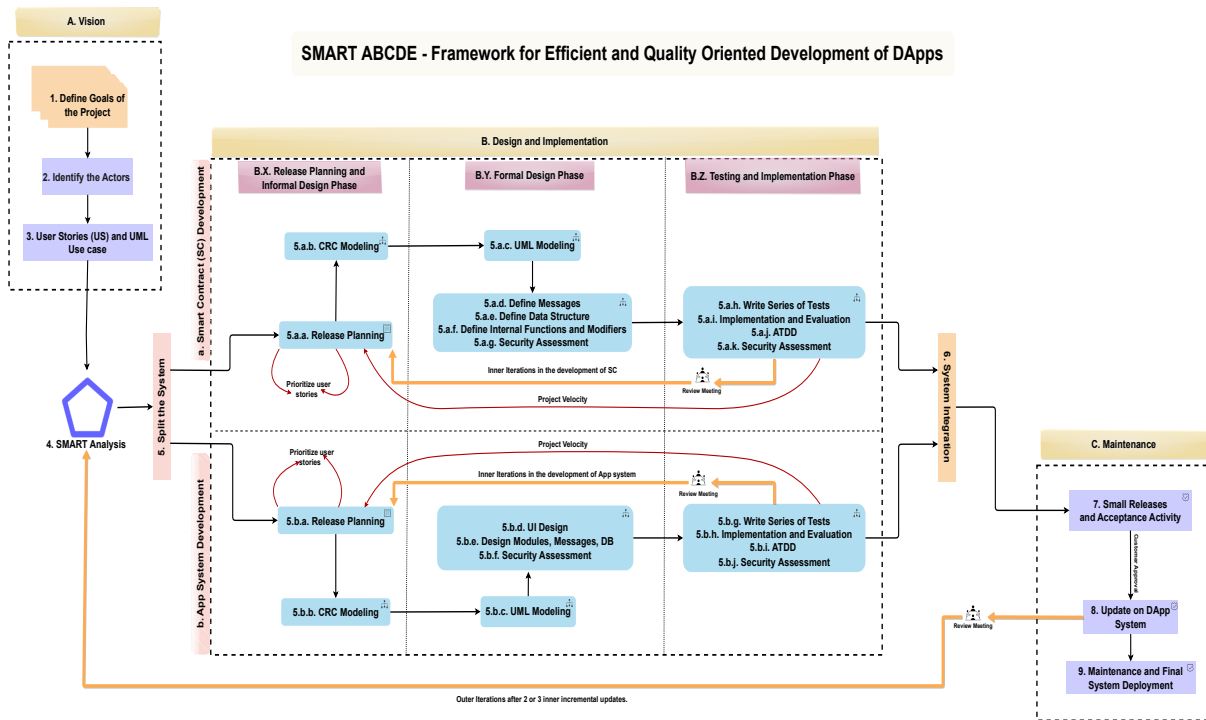


Questionnaire Survey

"Agile-based Approach for Efficient and Quality-oriented Development of DApps."

Section 1 - Purpose of Survey

Thank you for participating in this survey to validate my research on the topic **"Agile-Based Approaches for Efficient and Quality-oriented Development of Blockchain-enabled Smart Contracts based Decentralized Applications (DApps)"**. The primary objective of this survey is to gather insights and perspectives from industry professionals, developers, and stakeholders involved in traditional software as well as DApp development regarding the efficiency and quality aspects of utilizing agile methodologies. All responses will be handled with confidentiality and used solely for academic research purposes with anonymity.



Section 02 - Respondent Information and Demographic

1. What is your current role in the projects of the software development industry?

- Blockchain Developer
- Node JS developer
- Full Stack Developer (MERN, MEAN Stack, etc.)
- Project Manager
- System Analyst
- Software Quality Assurance (SQA) Engineer
- Other (please specify)

2. How many years of experience do you have in the software development industry, mainly in the development of DApps?

- Less than 1 year
- 1-3 years
- 3-5 years

ANNEX A

- More than 5 years
3. In which business domain does your software organization operate?
 4. On a scale of 1 to 5, how familiar are you with Blockchain technology and concepts?
 - 1 - Not at all familiar
 - 2 - Somewhat familiar
 - 3 - Moderately familiar
 - 4 - Familiar
 - 5 - Extremely familiar
 5. Rate your level of experience with agile methodologies in software development.
 - No experience
 - Beginner
 - Intermediate
 - Advanced
 - Expert

Section 03 - In practice Agile Development Methodologies and SMART analysis of the project goal

6. What agile methodology does your organization currently use in software applications development projects?
 - Scrum
 - Kanban
 - Extreme Programming (XP)
 - Lean Software Development (LSD)
 - Crystal
 - Feature Driven Development (FDD)
 - Other (please specify)

7. How satisfied are you with agile methodologies at your organization for the development of DApps?

- Not Satisfied
- Slightly Satisfied
- Neither satisfies nor dissatisfied
- Satisfied
- Highly Satisfied

8. What are the main challenges you face in the software development lifecycle even with the practice of agile methodologies?

- Project goals are not specified for development teams.
- Feedback delays on software releases.
- Inefficient resource allocation.
- Simplicity in UML design and communication.
- Delay in software bug detection and response.
- Quality-oriented development.
- Other (Please Specify)

Section 04 - Evaluation of the Agile-Based Proposed Methodology and SMART analysis of the goal for the efficient and quality-oriented development of DApps

Section 4.1 - Simplicity of the Proposed Methodology for the Development of DApps.

- Prioritizing the requirements

9. In your opinion, how essential is it to comprehend and prioritize DApp system requirements for the success of an Agile-driven approach?

- Extremely essential
- Very essential
- Moderately essential
- Slightly essential
- Not essential at all

- Analyze the project goal using SMART analysis

10. How effective, do you think of defining project goals using SMART analysis to simplify the agile development of DApps?

- Extremely effective
- Very effective
- Somewhat effective
- No very effective
- Not at all effective

- CRC Modeling

11. What do you think, how effective are CRC cards in the comprehension of UML diagrams in agile DApp development?

- Extremely effective
- Effective
- Somewhat effective
- Slightly effective
- Not effective at all

- Small and to-the-point incremental iterations and releases with review meeting

12. In your experience, how crucial are review meetings conducted at the end of each iteration in the agile development process of DApps?

- Extremely Crucial
- Crucial
- Somewhat crucial
- Not very crucial
- Not at all crucial

Section 4.2 - Agile Testing in the Proposed Methodology for the Development of DApps

- Test Driven Development

13. Do you believe that Test-Driven Development (TDD) affects the efficiency and quality of the development of DApps?
- Yes
 - No

- Pair Programming

14. Do you think that software testing using pair programming influences code quality and the ease of refactoring in the agile development of DApps?
- Yes
 - No

- Acceptance Test Driven Development

15. Do you agree that Acceptance Test Driven Development (ATDD) enhances the validation and acceptance criteria in the development of DApps?
- Strongly agree.
 - Agree.
 - Neutral.
 - Disagree.
 - Strongly disagree.

Section 4.3 - Efficient Development of DApps in the Proposed Methodology

- Prioritizing the user stories-based requirements

16. How effective is prioritizing user story-based requirements for efficient Agile development of DApps?
- Extremely effective
 - Very effective
 - Moderately effective

ANNEX A

- Slightly effective
- Not effective at all

- SMART analysis

17. How comprehensive are SMART objectives for efficient project goals specification in an agile framework?

- Extremely comprehensive
- Very comprehensive
- Moderately comprehensive
- Slightly comprehensive
- Not comprehensive at all

- Small and to-the-point incremental iterations and releases with a review meeting

18. Do you think Small and to-the-point incremental iterations and releases with review meetings can develop efficient software applications?

- Yes
- No

- CRC Modeling

19. Do you agree that a simple design method can result in the efficient design and requirements elicitation phase of the development of DApps?

- Strongly agree.
- Agree.
- Neutral.
- Disagree.
- Strongly disagree.

Section 4.4 - Quality Oriented Development of DApps using the Proposed Methodology

20. Which of the following agile practices do you find most crucial for the quality-oriented development of DApps?

- Testing practices i.e., TDD and ATDD
- CRC Modeling
- Prioritizing the user stories
- Small and Incremental releases.
- Other (please specify)

21. Do you believe that adopting the proposed agile practices could lead to efficient and quality-oriented development of DApps?

- Yes
- No

22. How well do you think the proposed agile framework aligns with the unique architecture (blockchain-based) of DApps?

- Poor
- Moderate
- Good
- Very Good
- Excellent

Section 5 - Conclusion and Feedback

23. Your additional feedback or comments would be appreciated regarding the proposed framework.