Agile Scrum Software Development: A Design of An Employment System

Samuel Oladapo Onamade

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Technology
in
Information Technology

Eastern Mediterranean University
August 2023
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Ali Hakan Ulusoy
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Technology in Information Technology.

Asst. Prof. Dr. Ece Çelik
Director, School of Computing and Technology

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Master of Technology in Information Technology.

Asst. Prof. Dr. Akile Oday
Supervisor

Examining Committee

1. Asst. Prof. Dr. Emre Özen

2. Assoc. Prof. Dr. Kamil Yurtkan

3. Asst. Prof. Dr. Akile Oday

ABSTRACT

Agile approaches have become a significant force, transforming several industries, including software development and Information Technology (IT). Agile is a set of values and principles that guide the delivery of a quality and satisfactory software project for the customer, using an adaptive, incremental, and iterative way of working by cross-functional and self-organized teams. This study delves into the complex world of Agile methodologies, clarifying the responsibilities, lifecycles, benefits, and drawbacks. The design and development of a new software solution as a design case called an Employment System (EMS), using the Agile Scrum software development paradigm. The Software Usability Measurement Inventory (SUMI) is used in this study to gauge users' opinions of the usability and quality of the EMS. The SUMI survey data are subjected to meticulous statistical analysis to produce verifiable insights.

The EMS outperforms the global average usability criteria, earning a score of 53.67 in the SUMI evaluation of the usability of the EMS produced within the Agile Scrum Software Development paradigm. These findings confirm the EMS's efficiency in meeting fundamental functional needs and highlight its high level of user satisfaction. This study has significant ramifications, especially for software development teams striving to improve system usability and user experiences.

Keywords: Agile, Scrum, Agile Scrum Software development, Employment System Design, Software usability measurement, SUMI (Software Usability Measurement Inventory), Agile project management.

ÖZ

Çevik yaklaşımlar, yazılım geliştirme ve bilgi teknolojisi (BT) dahil olmak üzere çeşitli endüstrileri dönüştüren önemli bir güç haline gelmiştir. Agile, çapraz fonksiyonel ve kendi kendini organize edilmiş ekipler tarafından uyarlanabilir, artımlı ve yinelemeli bir çalışma şekli kullanarak, müşteri için kaliteli ve tatmin edici bir yazılım projesinin sunulmasına rehberlik eden bir dizi değer ve ilkedir. Bu çalışma, sorumlulukları, yaşam döngülerini, faydaları ve dezavantajları netleştirerek çevik metodolojilerin karmaşık dünyasını incelemektedir. Çevik Scrum yazılım geliştirme paradigmasını kullanarak bir İstihdam Sistemi (EMS) adı verilen bir tasarım vakası olarak yeni bir yazılım çözümünün tasarımı ve geliştirilmesi. Yazılım Kullanılabilirlik Ölçüm Envanteri (SUMI), kullanıcıların EMS'nin kullanılabilirliği ve kalitesi hakkındaki görüşlerini ölçmek için bu çalışmada kullanılmaktadır. SUMI anket verileri, doğrulanabilir bilgiler üretmek için titiz istatistiksel analize tabi tutulur.

EMS, küresel ortalama kullanılabilirlik kriterlerinden daha iyi performans gösterir ve Agile Scrum yazılım geliştirme paradigmasında üretilen EMS'nin kullanılabilirliğinin SUMI değerlendirmesinde 53.67 puan kazanır. Bu bulgular, EMS'nin temel fonksiyonel ihtiyaçları karşılamadaki verimliliğini doğrulamakta ve yüksek kullanıcı memnuniyetini vurgulamaktadır. Bu çalışmada, özellikle sistem kullanılabilirliğini ve kullanıcı deneyimlerini geliştirmeye çalışan yazılım geliştirme ekipleri için önemli sonuçlara sahiptir.

Anahtar Kelimeler: Çevik, Scrum, Agile Scrum Yazılım Geliştirme, İstihdam Sistemi Tasarımı, Yazılım Kullanılabilirlik Ölçümü, SUMI (Yazılım Kullanılabilirlik Ölçüm Envanteri), Agile Proje Yönetimi.

DEDICATION

I dedicate this thesis to my late mother, Onamade Adebisi. I am forever grateful for her unwavering support and the time I was able to spend with her. I will never forget her wise words and guidance as I navigate life.

ACKNOWLEDGEMENT

I want to thank God Almighty, my Lord Jesus, for granting me the grace to complete my studies. I want to also thank Onamade Sunday, my father, for his love, support, and encouragement during my education. His unfailing faith in me and frequent support have provided me with strength and motivation. I would not have gotten this far without his prayers, love, and direction.

I am immensely grateful to my family and friends for their help during the process of my study and also for their unconditional support, understanding, patience, kindness, and encouragement towards me.

Lastly, my sincere appreciation goes to my thesis adviser, Asst. Prof. Dr. Akile Oday supported me throughout the process of my study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

1NF	First Normal Form
2NF	Second Normal Form
3NF	Third Normal Form
ASSDP	Agile Scrum Software Development Process
CI	Confidence interval
CSS	Cascading Style Sheets
DSDM	Dynamic System Development Model
EMS	Employment System
FDD	Feature-Driven Development
FK	Foreign Key
HTML	Hyper Text Markup Language
ICA	The Item Consensual Analysis
IQR	Interquartile Range
IT	Information Technology
JS	JavaScript
JSON	JavaScript Object Notation
MVC	Model View controller
PB	Product Backlog
PI	Program Increment
PK	Primary Key
PO	Product Owner
SAFe	Scaled Agile Framework
SB	Sprint Backlog

SDK	Software Development Kit
SME	Small and medium-sized enterprises
SoS	Scrum of Scrums
SSL	Secure Sockets Layer
SUMI	Software Usability Measurement Inventory
TDD	Test Driven Development
TEIM	The Evolved Integrated Model
XP	Extreme programming

Chapter 1

INTRODUCTION

1.1 Background Study

The phenomenon of agile software development has been the core of innovation in software development companies and other Information Technology (IT) industries. Agile is a philosophy in which software development projects are approached (Gheorghe, Gheorghe, & Iatan, 2020). These scholars also defined agile as a set of values and principles that guide the delivery of a quality and satisfactory software project for the customer, using an adaptive, incremental, and iterative way of working by cross-functional and self-organized teams.

Agile is a way of thinking that strongly emphasizes teamwork, communication, intrinsic motivation, enabling innovation, and creating value while creating a software product. The agile methodologies are an innovative approach compared to the traditional method (plan-driven processes), which places emphasis on meticulous preparation and documentation with rigid procedures and holds the belief that specific and foreseeably solutions to issues exist (Al-Saqqa, Sawalha, & AbdelNabi, 2020). The software industry has seen practitioners who use the plan-driven approach use a more flexible system because of the rapid changes occurring in the sector (Gheorghe, Gheorghe, & Iatan, 2020).

As the project is still in progress, agile software development allows for a high degree of responsiveness to changes based on the client's demands, thereby reducing the rate of project failures and effective team communication. (Al-Saqqa, Sawalha, & AbdelNabi, 2020).

Agile software development is a product of the Agile Manifesto, which states:

a. Individuals and interactions over processes and tools,
b. Working software over comprehensive documentation,
c. Customer collaboration over contract negotiation and
d. Responding to a changeover following a plan.

In summary, understanding the influence of innovation on agile software development is crucial for software development organizations and practitioners who want to stay competitive and successful.

This study will add to the current knowledge on agile software development and innovation and give insights that software development businesses can utilize to enhance their processes and outcomes.

1.2 Problem Statement

In today's dynamic and fast-changing business climate, enterprises must create high-quality software solutions that fulfill their customers' expectations. Agile software development has evolved as a popular strategy, providing better flexibility, collaboration, and speed than traditional approaches. Nevertheless, implementing agile development may be difficult, especially when managing personnel and guaranteeing productivity and engagement.

By effectively adhering to the scrum methodology to minimize these issues. Hence, adopting scrum methodology to develop an Employment System (EMS) for this study and assess user usability and satisfaction. We created the EMS with innovation and cooperation in mind and the tools and resources required for successful personnel management and performance monitoring.

It is critical to undertake usability testing with end-users like SUMI to guarantee the effectiveness and usability of the developed EMS. SUMI is a typical system end-user, and her opinion will be crucial in ensuring that the system is intuitive, user-friendly, and satisfies the demands of its intended users.

1.3 Research Questions

This study is carried out to provide answers to the following questions.

(a) How does the Agile Scrum Software Development Process (ASSDP) impact EMS learnability?
(b) How would ASSDP influence the efficiency of the EMS?
(c) How would ASSDP impact the EMS's effect on the user?
(d) How would ASSDP impact the user's control of the EMS?
(e) How might ASSDP affect the EMS's helpfulness for users?

1.4 Purpose of the Study

The software development business is dynamic, with technology and consumer expectations constantly changing (Ghezzi & Cavallo, 2020). With its emphasis on flexibility, collaboration, and customer satisfaction, agile software development has developed as a popular method. In the software business, on the other hand, innovation

is a significant driver of competitiveness and growth (Brand, Tiberius, Bican, & Brem 2021).

The primary purpose of this study is to capture how agile scrum methodology innovation contributes to increased usability, efficiency, and success in software development. This can be accomplished by developing a supportive theoretical framework based on extant studies on various agile software development methods. After that, the design of an EMS is based on the Agile Scrum Software Development Process (ASSDP). The following are the primary objectives established to achieve the primary goal of this paper:

1. We are investigating the influence of innovation on agile software development.
2. To understand ASSDP adoption and use in the system's design.
3. Highlights the EMS's perceived attributes (affect, control, learning, efficiency, and helpfulness).

1.5 Definition of Terms

The ASSDP is a process of managing software projects via the Scrum methodologies. EMS is the system developed using ASSDP to appraise, record, and monitor personnel/Employee information.

1.6 Outline

The first chapter of the research describes the background of the study, objectives, and limitations. The second chapter provides a previous literature review, addressing general issues with software projects and various software development approaches. The third chapter describes the hypothesis analysis and the constructed system using the scrum method, including its architecture, database, and security details. The fourth

chapter examines this study's research methodologies, terminology, and ideas. The fifth chapter was a detailed analysis of the SUMI report on the EMS that has been evaluated via the SUMI tool. The sixth and final chapter was a summary conclusion and recommendation for further study.

1.7 Limitations

Like any research, this research focuses on agile scrum software development. The design case of EMS is based on a Small and medium-sized enterprises (SME) organization. The focus is on the scrum master and scrum team; a sample small-scale company team examines the scrum life-cycle, as all scrum teams worldwide cannot be entertained for the study.

This study also has time constraints due to limited resources, such as funding and personnel, which may affect the depth and scope of the research. It is also an impediment to ascertaining the participant response data’s trustworthiness and accuracy.

Chapter 2

LITERATURE REVIEW

2.1 General Issues with The Software Development Project

According to Cruz & Alves (2020), the complexity of software process development cycles is impacted by several factors, including the number of lines of code, the frameworks used, embedded code, and untraced errors.

Lavanya (2020) stated that accurately forecasting time and resources for software development endeavors is another common problem. This scholar also noted that a realistic timescale is essential given the constraints of resources and time. A project component may not operate correctly or be functionally deficient if a developer does not devote enough attention. We risk wasting unnecessary work hours due to poor time and resource management.

Heriyanti and Ishak (2020) state that limited, well-structured guidelines or plans to develop a software project constitute one of the most common problems. Teams must get clear instructions to perform their duties effectively; otherwise, confusion and mayhem may result, causing delays and declining project quality (Iqbal et al., 2020).

Moyo (2020) stated that insufficient testing computers, a shortage of software engineers, or a lack of pertinent technologies for their projects are some issues that software development teams could run against.

According to Sutherland and Schwaber (2011), defining requirements is a time-consuming job for software teams. This involves figuring out what the project should achieve and how it should operate. The developers must understand what they must do in a software project; hence, the requirements must be crystal clear, concise, and detailed. Unfortunately, getting a clear idea of what is anticipated might be challenging, and it is even more difficult to translate that idea into specific instructions. Unclarified concepts and directives cause developer annoyance and confusion, ultimately delaying the project.

According to Iqbal et al. (2020), Miscommunication with stakeholders and customers is yet another significant problem for software engineers. This may happen for several reasons, such as a lack of communication channels, misunderstandings of requirements, or inadequate project documentation. It may also result from a breakdown in communication between members of software development teams; hence, developing effective communication techniques is essential. Whatever the cause, poor communication can lead to project delays and unsatisfactory results (Sutherland and Schwaber, 2011).

According to Thummadi and Lyytinen (2020), an approach-induced variation using agile and waterfall accounts for 40% of the activities and other factors, such as people, project stability, and other environmental factors.

2.2 Software Development Approaches

Agile methodologies have effectively managed these issues through innovation in the software development process in which the traditional (waterfall) approach could not address the escalating complexity and unpredictability, thereby becoming extremely popular in creating innovative software products.

2.2.1 Traditional approach (Waterfall model)

Heriyanti and Ishak (2020) state that the model is easy to understand and may be used to illustrate the conventional approach to software development. The waterfall model is a static paradigm that approaches system development linearly and sequentially, with one activity being finished before the next.

The waterfall, according to Al-Saqqa, Sawalha, and AbdelNabi (2020), is segmented into communication (project initiation and requirement gathering), planning (Estimating, scheduling, and tracking), modeling (Analysis and design), construction (coding and testing), and deployment (delivery, support, and feedback). The phases of a design include requirement analysis, system design, program design, coding, unit and integration testing, system testing, acceptability testing, and maintenance (Heriyanti & Ishak, 2020).

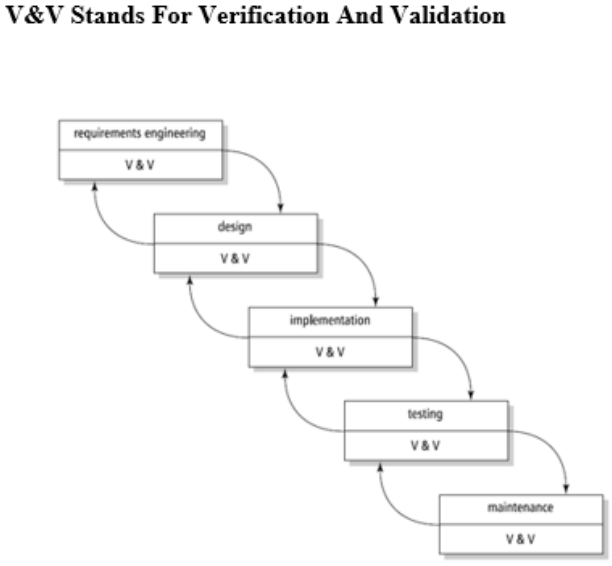


Figure 2.1: Waterfall model by (Van, Van & Van, 2008, p.45).

Al-Saqqa, Sawalha, and AbdelNabi (2020) Briefly stated the waterfall model entails the steps of requirement engineering, design, implementation, various types of testing, and maintenance, as shown in Figure 2.1. Verification tries to validate the legitimacy of moving on to the following step by determining if the system satisfies the requirements. Validation establishes whether the system meets the user's needs.

2.2.1.1 System Developed using Waterfall

Heriyanti and Ishak (2020) highlight some application problems that can be solved using the waterfall model: information control system, inventory information system, web-based hospital record management information system, and A Simulation Model for the Waterfall Software Development Life Cycle.

Budiarti, Fathin, and Sulistiyani (2022) developed a Website-Based Student Achievement Book using the waterfall method and performed usability testing using 21 questionary variables that focused on three aspects (efficiency, effectiveness, and satisfaction): respondents for the questionnaire, and the result for this assessment was an overall mean score of 84 and overall percentage of 90%.

2.2.1.2 Advantages of the Waterfall Model

Moyo (2020) stated the waterfall model has clear and structural stages that are easy to manage. According to Heriyanti and Ishak (2020), Documentation makes it easy for maintenance and new team members to adapt to the process. These scholars also suggest that the model is easy to understand and use.

2.2.1.3 Disadvantages of the waterfall model

According to Moyo (2020), this model is rigid and inflexible; hence, adapting to project needs or scope changes is difficult. This scholar also stated that there is little room for feedback and modifications. According to Moyo (2020), the Waterfall approach is considered high risk since any flaws or issues identified later in the process

might be costly to correct. These scholars also highlighted that because testing is often performed towards the end of the development cycle, any flaws or difficulties may not be discovered until late in the project.

2.2.2 Agile Methodology

The agile approach may be considered creative compared to the traditional method (plan-driven processes), which emphasizes thorough planning and documentation with rigorous procedures and holds that there are particular and foreseeable answers to problems. Agile approaches (frameworks or processes) employ a scientific technique that replaces a programmed algorithmic approach with a more heuristic one with respect for people and self-organization to cope with the unpredictability of complex situations (Lavanya, 2020). This is because it promotes self-learning and self-organization instead of a lot of planning, and an extensive collection has interwoven mandatory components.

The Agile Manifesto, which asserts that people and interactions come before procedures and technologies, working software comes before thorough documentation, customer participation comes before contract negotiations, and responding to change comes before following a plan, is said to be the source of agile software development, (Schwaber & Sutherland 2011).

Extreme Programming (XP), Dynamic System Development Model (DSDM), Test-Driven Development (TDD), Feature Driven Development (FDD), Kanban, Scaled Agile Framework (SAFe), and Scrum are just a few examples of agile approaches.

2.2.2.1 Extreme Programming (XP)

XP was one of the earliest agile methods that Kent Beck suggested 1999 to get around the limitations of the convolutional software development process in the face of quickly changing requirements and to create a methodology fit for object-oriented

projects involving multiple programmers in a single location. Shrivastava et al. (2021) and Moyo (2020) describe XP as one of the earliest agile methods. It integrates well-known methods used in software engineering. Al-Saqqa, Sawalha, and AbdelNabi (2020) stated that XP enhances client satisfaction and software reliability and decreases the cost of specification modifications by breaking down lengthy cycles into several shorter cycles.

Shrivastava et al. (2021) highlighted that XP enhances communication, feedback, simplicity, and high-quality work.

These scholars also highlighted the following responsibilities in XP:

a) Clients are responsible for documenting the features and requirements of the system as stories. Also, carrying out functional tests gives them the necessary precedence and determines the extent to which every prerequisite is met after the process.
b) Programmer: Responsible for creating clear, effective functioning code, typically with other programmers utilizing the pair programming technique.
c) Testers work with the client to create and execute the necessary functional tests.
d) Tracker evaluates the team's estimations and development throughout each iteration and offers feedback.
e) The coach serves as the process director and chief overseer.
f) A consultant is a knowledgeable outsider who helps the team solve problems.
g) The manager is responsible for leading the team, communicating, and removing barriers.

2.2.2.1.1 XP Life Cycle

The six steps of the XP life cycle, as described by Abrahamsson, Salo, Ronkainen, and Warsta (2017), are illustrated in Figure 2.2 and are as follows:

a. Phase of Exploration

The XP team, including the client, works in synergy to define and comprehend the project's requirements. This includes reviewing initial user stories, features, and project plans. Also, the XP team is presented with the resources that are accessible, including tools, technologies, and processes, so they are familiar with them throughout the project.

b. Phase of planning

This phase ensures the efficacy and success of any project. The developers propose and settle on a timetable and resource estimate, in addition to ranking the stories' duration for a few days.

c. The Phase Of "Iterations To Release"

It undergoes several iterations, each of which lasts one to three weeks. These iterations would include several activities such as design and development, testing, and integration

According to Al-Saqqa, Sawalha, and AbdelNabi (2020), pair programming and refactoring are the essential techniques used at this stage.

In **Pair programming**, engineers work together in teams to write code, with one working diligently on the script while the other observes, supports, and evaluates it. This has several benefits, such as spreading knowledge around the team, mainly when

these pairs are chosen randomly, and encouraging collective ownership and accountability for the resulting code. Additionally, it promotes teamwork and synergy.

Refactoring is a technique for enhancing software and making it simpler and easier to maintain by rewriting its code while maintaining its functionality (Al-Saqqa, Sawalha, & AbdelNabi (2020). Renaming code functions and variables and switching from procedural programming to object-oriented programming can all help to increase comprehension, reduce repetition, and increase flexibility.

d. Phase of Production

More testing and performance reviews are necessary before the system is available to the client. The system should continue to work after the first version is sent to the client while further iterations are developed.

e. Phase of Maintenance

The production phase is inextricably intertwined with the maintenance phase. Production and maintenance are not considered different operations in XP but rather two sides of the same coin. This is when bug fixes, feature additions, performance optimization, and security upgrades are implemented. Ultimately, the XP maintenance phase is continuous, including a continual effort to improve the program and fulfill changing needs. XP teams prioritize maintenance operations based on their value to the end user and their influence on the product's quality.

f. Final Stage (Death Phase)

This is the termination of activities due to the completion of user stories, features being met, and all system requirements being satisfied.

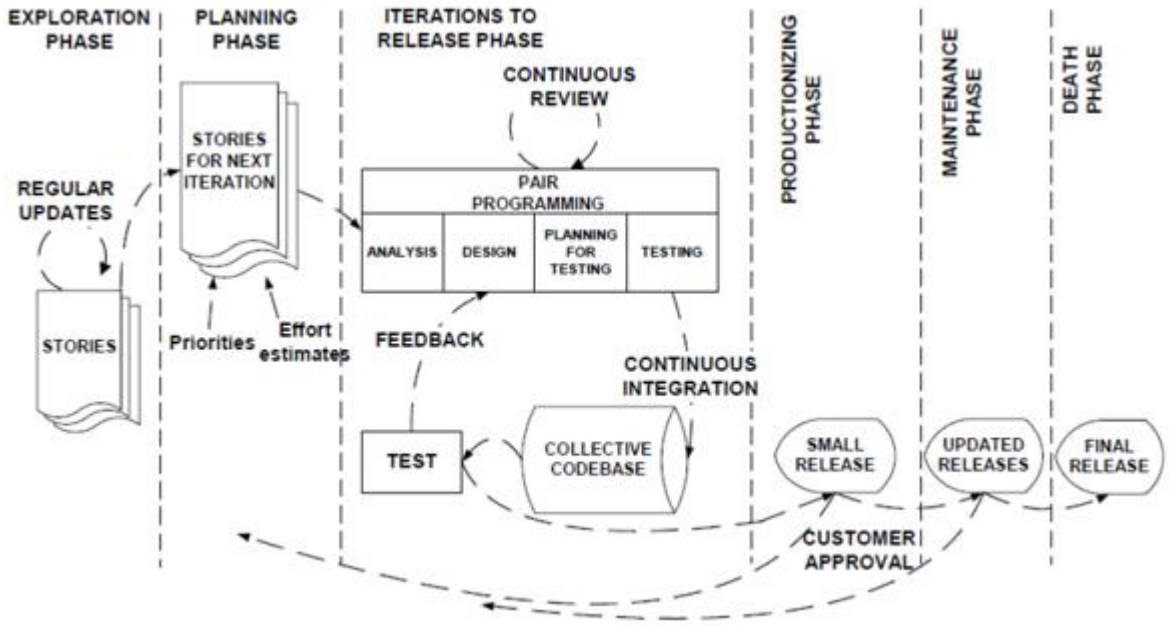


Figure 2.2: XP lifecycle (Al-Saqqa, Sawalha, & AbdelNabi, 2020)

Termination could also be due to scope creep or lack of resources for completing the project. 2.2.2.1.2

2.2.2.1.2 Advantages of XP

Lawal and Ogbu (2021) highlighted that Regular, minor system releases help incremental development, Hence Boost productivity fast feedback. Abrahamsson et al. (2017) stated Updating the code frequently to keep it simple. Before adding features to the system, a test is carried out to enhance quality (Marek, Wińska & Dąbrowski 2021).

2.2.2.1.3 Disadvantages of XP

Abrahamsson et al. (2017) stated that it cannot support remote teams because it focuses on community and co-location. Test-driven development demands that team members receive additional technical training. Al-Saqqa, Sawalha, and AbdelNabi (2020) highlighted that Customer participation is advantageous but time-consuming.

2.2.2.2 Dynamic System Development Model (DSDM)

DSDM is an agile project development methodology that prioritizes quality above speed when developing applications. It offers total support for the whole software development life cycle. Like other agile models, it employed an incremental and iterative method to produce high-quality software with ongoing user interaction.

The fundamental goal of the DSDM is to establish and specify the project's resources and time frame before modifying the amount of functionality that can be accomplished within these parameters. The list of system functionality is predefined first, and then time and resources are assigned and defined, which is the opposite of typical procedures.

2.*2.2*.2.1 The DSDM Lifecycle

According to Anwer, Aftab, Waheed, and Muhammad (2017), there are six main phases to the DSDM. Because DSDM uses an incremental approach, customer input raises the software's caliber. These are the DSDM phases:

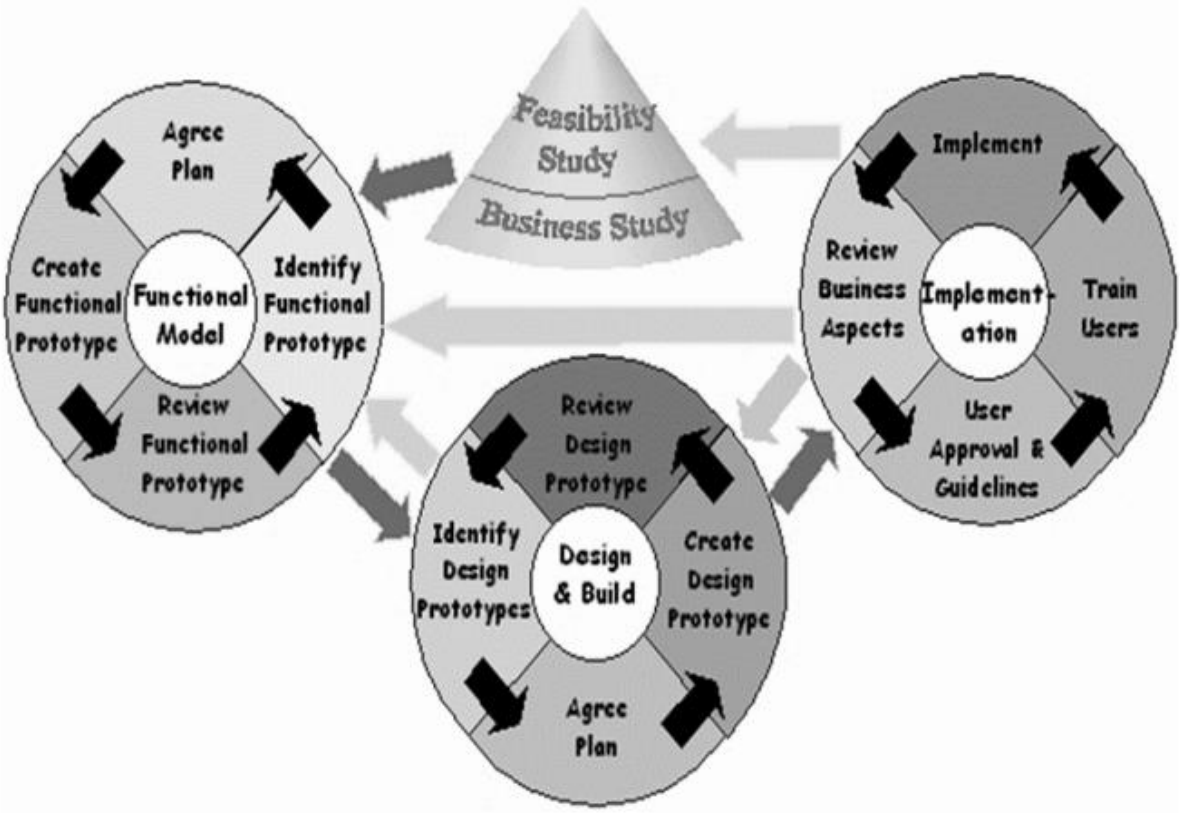


Figure 2.3: DSDM Lifecycle (Anwer et al. 2017)

a) The project that will be produced is chosen based on its suitability for DSDM. Estimating the project's financial needs is easy since its scope and goal are well stated. During this phase, crucial positions like a project manager and development team are also selected. The feasibility research phase is also scheduled for the pre-project stage.

b) The DSDM method's feasibility study phase will assess the project to see if the DSDM is suitable for use in development. The specification of technical requirements and risk assessments are also included in this step. An outline development plan and a feasibility study are the results of this phase.

c) During the business research phase, discussions are held between business experts from the client side and the development team to clarify the user's needs and generate a list of functions needed in the system.

d) Coding, analysis, and prototyping are carried out progressively and iteratively in the Functional model iteration. Prototypes are also examined to enhance the analytical model. This step's primary objective is to outline the software's development schedule, methods, and components. Functional prototypes, nonfunctional requirements, risk analysis, and prioritized functionalities are also covered.

e) This is an iterative phase where the defined needs from the prior phase are implemented and programmed before being released and put to user testing. User input is utilized iteratively to enhance the system. This process results in the creation of test software that meets at least a basic set of standards.

f) The phase of the implementation, when the program is released from development and given to users, also includes training sessions. Additionally, the user handbook is developed and updated. No more development procedures are necessary if the program satisfies every user's demand. If not, the whole thing is redone.

2.2.2.2.2 DSDM Responsibilities

Abrahamsson et al. (2017) highlighted the primary DSDM responsibilities are as follows:

a) The development team is a designer, programmer, tester, and Business analyst.

b) A technical coordinator oversees the project's business and technical operations. Additionally, it establishes the system's architecture and guarantees the technical excellence of the project.
c) A representative user from the client side will utilize the system once it is built and can communicate with the development team about all the users' demands.
d) An adviser user who is also a client can offer specific crucial perspectives that the ambassador user cannot, which are necessary for the project.
e) The client who is aware of the overall goals and demands of the company is the visionary.
f) An individual who has the power to make choices and handle money is known as an executive sponsor.

2.2.2.2.3 Advantages of DSDM

Abrahamsson et al. (2017) stated that it facilitates agile principles-based quick application development. These scholars also highlighted It can offer guidance for project management, risk management, and development methods.

2.2.2.2.4 Disadvantages of Using DSDM

Abrahamsson et al. (2017) stated that the several functions that make up the DSDM will make project administration difficult. These scholars also highlighted that DSDM does not consider project criticality. According to Anwer, Aftab, Waheed, and Muhammad (2017), when it comes to matters like team size and iteration duration, DSDM does not offer any precise recommendations.

2.2.2.3 Test Driven Development (TDD)

According to Al-Saqqa et al. (2020), it is based on the TDD methodology relation of small, iteratively automated testing programs, followed by writing programs that can pass those tests and postpone further code enhancement.

TDD is the exact antithesis of conventional software development techniques, which carry out testing after the code is written. TDD was initially suggested by Kent Beck in 2003. However, NASA began using it in 1950 while working on the Mercury project. Moyo (2020) and Al-Saqqa et al. (2020) stated that TDD will improve code quality and decrease the number of mistakes and problems. Furthermore, these researchers noted that TDD has two essential guidelines: the first is to test the program, and the second is to confirm the program's failure or success, depending on the outcome, an action to redefine the program if the test fails or to go to the next program if the test succeeds.

The written tests by the programmer will not be regarded as the sole necessary test for the project since they are based on a single testing program that the developer creates to fulfill a single minor need in the software. Therefore, further testing, such as stress tests, performance tests, and so on, must still be performed on the program.

2.2.2.3.1 TDD Life Cycle

According to Al-Saqqa et al. (2020), there are five critical sections in the TDD lifecycle, as shown in Figure 2.4. Each aspect of the necessary software will be subjected to this set of procedures. Adhering to the process enhances quality. They are as follows:

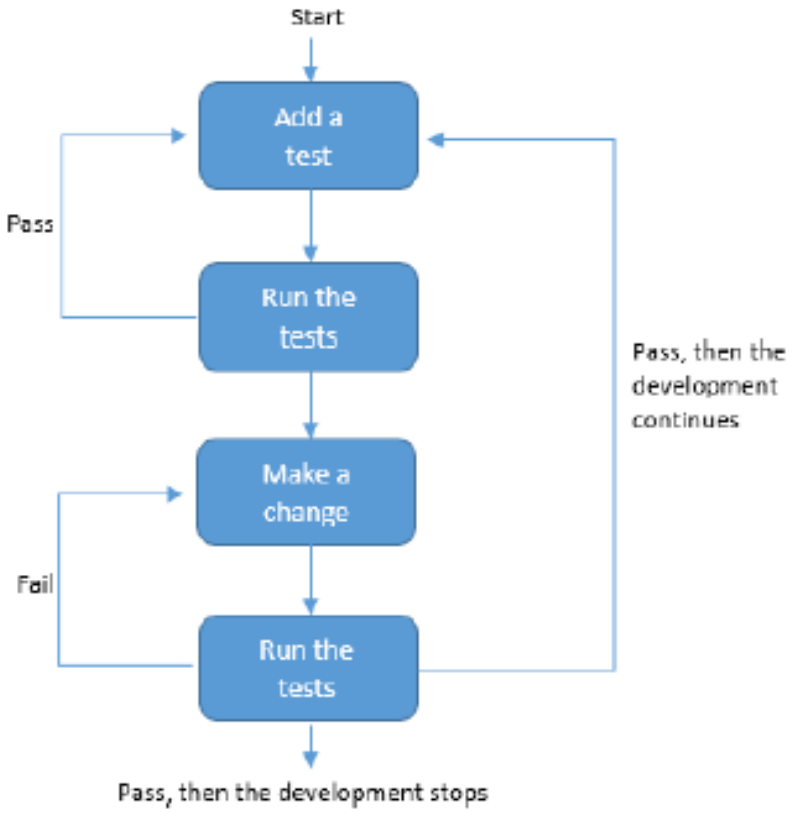


Figure 2.4: TDD lifecycle (Al-Saqqa et al. 2020)

a) Add a test; as previously said, the first step in creating software using the TDD methodology is to create an automated test based on the project's specifications.

b) The developer uses the client requirements list and the use cases for these needs to generate this automated test.

c) Run all test cases against the unavailable code in this stage, causing all test cases to fail to satisfy some criteria.

d) Write the code; the developer will examine the failed test cases and create the code to fix them. The resulting code may be inefficient for now, as the developer's top priority is passing the test.

e) The developer will rerun each test case on the created code to determine whether the issue has been fixed. The new software complies with the specifications if all test cases are passed, or the code must be altered several times until successful.

f) The TDD process ends with refactoring the code, which involves streamlining the code and eliminating duplicates.

2.2.2.3.2 Advantages of TDD

According to Al-Saqqa et al. (2020), errors are addressed early, and faults are promptly discovered, eliminating debugging needs since testing is done concurrently with software development. Since all code is tested while it is being developed, iterative testing enhances system quality.

Abrahamsson et al. (2017) stated that the development process will be more straightforward if the program is broken up into smaller pieces, as the developer will focus on a single issue to address in each iteration. According to Lawal and Ogbu (2021), Better design and easier integration with other features will follow from refactoring the code. Developing modest capabilities in each cycle will lower the total complexity of the product (Al-Saqqa et al., 2020).

2.2.2.3.3 Disadvantages of TDD

According to Al-Saqqa et al. (2020), Programmers will be confused since building testing cases is a skill that they must acquire in addition to their programming expertise. These scholars also highlighted that TDD techniques are challenging to use in projects when features and pieces synchronize.

Abrahamsson et al. (2017) stated that TDD produces subpar documentation since test cases are only utilized during development, whereas documentation is helpful during

maintenance. When failures occur repeatedly, TDD can be time-consuming. Management concepts do not govern software development using TDD; instead, it primarily concentrates on development activities (Lawal & Ogbu, 2021).

2.2.2.4 Feature-Driven Development (FDD)

FDD deals with small incremental iterations that lead to usable software. These small cumulative iteration functions are helpful for users as specified in system requirements and are called features.

Al-Saqqa et al. (2020) stated that the FDD approach was created by Jeff De Luca in 1997 when he realized that earlier development approaches weren't appropriate for creating a large project in a certain amount of time. These scholars also highlighted that Jeff De Luca and Peter Coad wrote about this approach in 1999 in their book Java Modeling in Color with UML by fusing the idea of features with the software development process.

According to Anwer et al. (2017), FDD is a very adaptable approach to software development that can consider last-minute changes in program requirements.

2.2.2.4.1 FDD Method Life Cycle

According to Anwer et al. (2017), Five consecutive steps are carried out gradually during the iterative life cycle of the FDD technique to produce the finished program. The following are the steps:

a) Develop the broad model: All team members and specialists start with the project scope definition at this phase. Detailed review and analysis of several models developed by experts and then selecting the most suitable model for the project based on requirements.

b) Developing a feature list: using the selected model and the requirements to build the system's overall feature list that satisfies the business or users' needs.

Before being validated, the customer and business experts will review this feature list.

c) Build a plan based on features: This step will develop a high-level strategy using the list of features that have already been authorized as a basis. The customer's priorities and the interplay of these factors will determine the sequencing of the plan. A timeline for the project's significant checkpoints and an in-depth calendar for each feature is both included in this plan. All necessary stakeholders are involved in this process. A developer is assigned the responsibility for the components.

d) Each iteration of the design process should run no more than two weeks—design by feature. The chief architect (programmer) and class owners (developers) also provide a design package for each class and the class and interaction diagrams.

e) Build by feature: This is the last part of the FDD process when the designs are put into code, tested, and evaluated. Like the design by feature stage, this method is iterative. Once all iterations have been finished, the created features are published in the main build, and a new set of features is started.

Figure 2.5: FDD lifecycle

Anwer et al. (2017) and Abrahamsson et al. (2017) highlight several team members can play the six critical roles, and one team member can play numerous positions in the FDD methodology. The following are the roles:

a) As the initiative's chief decision-maker, the project manager creates the ideal working environment.
b) The Chief Architect oversees the management design of the necessary software and grants formal approval of the plan.
c) The Development Manager oversees the complete development team and possesses the solid technical knowledge needed to address some issues that the Chief Programmer cannot handle. He is responsible for settling disputes amongst the team.
d) The Chief Programmer leads small teams through all of the development process's analysis, design, and implementation phases. Additionally, he decides which things need to be finished in this iteration and handles any implementation problems.
e) The class owner or developer.
f) A specialist in a particular industry, a domain expert is familiar with the industry's requirements as well as its workings. He adds his commercial expertise to the list of desired features that will be executed.
g) Feature Team is a transient team of developers that is iteratively working on a few feature sets.

Using Feature Driven Development techniques offers various advantages and disadvantages, according to Anwer et al. (2017).

2.2.2.4.2 Advantages of Implementing the FDD

According to Anwer et al. (2017), FDD is a highly adapted development model that strongly emphasizes the project's design and modeling elements. These scholars also state that FDD is highly flexible to incorporate last-minute client changes.

Anwer et al. (2017) and Abrahamsson et al. (2017) highlighted that each iteration's findings may be supplied in one to four weeks, enabling prompt client feedback.

2.2.2.4.3 Disadvantages of Implementing the FDD

Anwer et al. (2017) and Abrahamsson et al. (2017) highlighted that FDD offers no recommendations for obtaining requirements, conducting analyses, or managing risks.

According to Al-Saqqa et al. (2020), A highly competent design and modeling team is required for FDD. Lawal and Ogbu (2021) stated that FDD does not address project criticality issues effectively.

2.2.2.5 KANBAN

According to Alqudah and Razali (2018), Kanban is an agile software development methodology emphasizing "just-in-time" delivery. Kanban's main objective is to clearly define the work to be done and the allocated timeline for the work. It sets task priorities, develops processes, and calculates delivery times. Essential tasks in the project are highlighted by the Kanban technique to reduce the likelihood of failure and to enhance flexibility among other tasks in the project.

The Kanban emphasizes evolutionary development and ongoing process improvement. Kanban is composed of six core procedures:

a) Work in Process Restrictions (WIP).

b) Enhancing throughput.

c) Adding value while the project is being developed.

d) Making the development process more visible.

e) Embedding quality.

f) Using a predefined backlog.

Teams visualize their work on a Kanban board, which serves as a central information hub and where all tasks should be placed. As a result, people can exchange information much more swiftly and collaborate more effectively when working on different projects.

In a 2017 study by Lei, Ganjeizadeh, Jayachandran, and Ozcan, supported by Schwaber et al. (2011), the Kanban board is divided into columns representing the various phases of the procedure. Hence, this allows the project managers and teams to plan better and manage their workloads, keep track of multiple projects, and better understand the process.

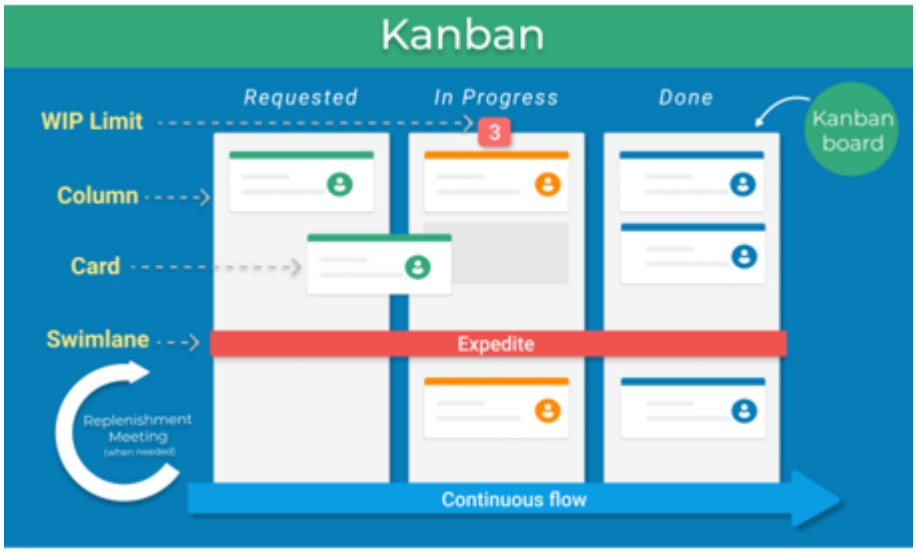


Figure 2.6: Kanban lifecycle (Schwaber & Sutherland 2011)

The Kanban technique focuses on finishing the appropriate project at the right time, given the expertise of the specialists. Developers may have a range of skill levels and productivity rates.

2.2.2.5.1 Advantages of Kanban

Alqudah and Razali (2018) suggested that Kanban encourages cooperation and collaboration by visualizing the workflow and allowing team members to see what tasks need to be performed, what is in process, and what has been finished. According to Schwaber et al. (2011), by reducing work in progress (WIP) and focusing on finishing activities one at a time, Kanban helps to decrease waste and boost productivity. Kanban helps teams generate high-quality work while reducing the possibility of mistakes and faults by focusing on finishing one job at a time (Lei, Ganjeizadeh, Jayachandran & Ozcan, 2017)

According to Moyo (2020), Kanban represents work items using visual indicators, such as task cards on a board, making it more straightforward for teams to see their workflow and discover bottlenecks or areas for improvement.

2.2.2.5.2 Disadvantage of Kanban

1. Kanban is not as organized as other Agile approaches, such as Scrum, which might make managing big and complicated projects more difficult. This might lead to a lack of consistency and make long-term goals harder to prepare.
2. Kanban focuses on individual tasks, which can lead to inadequate team collaboration and communication, especially in more significant projects involving numerous teams.
3. While visual signals can be helpful, they can sometimes be restrictive, especially in projects with more complicated tasks that require more extensive documentation.

4. Kanban lacks a standardized prioritizing method, making it challenging to prioritize activities and leading to confusion and misunderstandings.

2.2.2.6 Scaled Agile Framework (SAFe)

Dean Lifting paved the way for the well-scaled agile framework (safe), a significant development framework combining lean and agile principles with existing ones to provide a model for projects of this size (Kalenda, 2017).

Three (3) central bodies of knowledge are tapped into:

a) Agile software development.
b) Lean product creation.
c) Systematic thinking

According to Kalenda (2017), SAFe is both a framework and a collection of Agile development best practices for small and large enterprises. SAFe provides a high degree of flexibility and optional expansion. This scholar also stated that SAFe has been utilized by enterprises such as Intel, Hewlett-Packard, and Cisco.

2.2.2.6.1 Safe Architecture

According to Almeida and Espinheira (2021), SAFe architecture is divided into layers. There are three fundamental levels: portfolio, program, and team. There is an alternative Value Stream level for large organizations. Across all levels, the Foundation level is supported by additional aspects that help steer organizations. SAFe core values, a Lean-Agile mentality, and SAFe principles are examples of Foundation layer elements.

a. The Portfolio Levels

The enterprise's mission is guided by this SAFe level, which is the highest. It lays forth and controls the core strategic choices that benefit the company. Additionally, it manages the budget for the solution, provides services for projects, and coordinates program effort and architecture. While giant corporations may need many Portfolio levels, small and medium-sized organizations may just need one.

b. Program Levels

According to Kalenda (2017), The Program level executes the defined objective. It coordinates, supports, and upholds the numerous Agile teams that make up the solution.

Kalenda (2017) stated that numerous Agile Release Trains (ARTs) exist at the program level. These ARTs bring several teams together. Iterations are used to make ARTs. One such repetition is the Program Increment (PI).

c. Team Levels

The entry-level for SAFe is Team. It describes how an Agile team operates using Scrum, Kanban, and XP planning methodologies.

According to Kalenda (2017), ARTs teams are cross-functional and include all tasks necessary to take ideas from conception to deployment, including all stages of development. The description of the program and team level is illustrated in Figure 2.7 below. Architects and user experience designers support ARTs teams in ensuring the quality of their products.

2.2.2.6.2 Four ARTs Goals Are Identified in SAFe

a. Integrate Agile teams to complete a single task by using the program backlog.

b. Create a value for the system that can be analyzed twice a month.

c. Alignment of an agile team with their iterations.

d. PI should include team iterations.

2.2.2.6.3 Team Roles for SAFe Agile ARTs

Common Agile approaches, such as Scrum, XP, and Kanban, are adopted by SAFe Agile teams. Each side may choose any of these strategies. Among the techniques used to improve software and hardware quality are continuous integration, test-first, refactoring, pair programming, frequent system-level integration, and design verification.

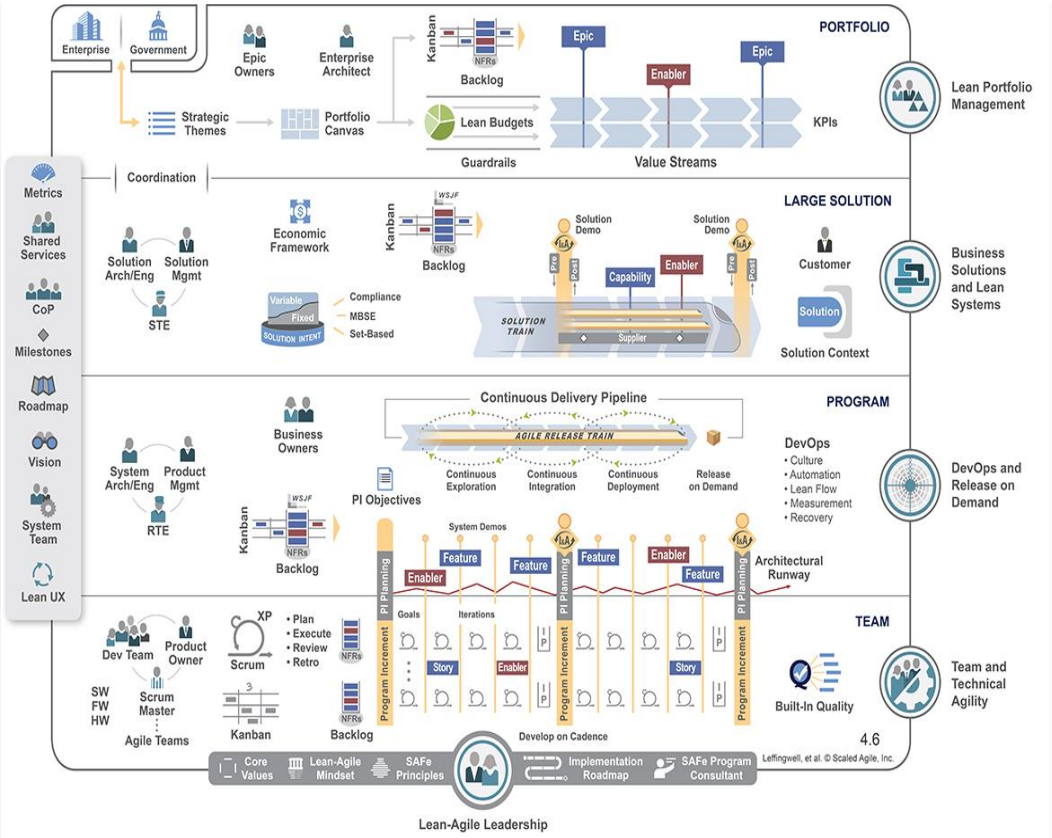


Figure 2.7: SAFe Architecture (SAFe 4.6 - IRIS Business Architect (biz-architect.com))

a. The RTE (Release Train Engineer) is the scaled-down Scrum Master. The Scrum Master is responsible for overseeing the whole Agile Release Train. He leads the implementation of programs at the program level, removes obstacles, controls risks, and promotes continual program development.
b. Program backlog management should develop a vision, collaborate with users, ascertain users' needs and requirements, and create efficient communication with stakeholders.
c. System Architect-Engineers are individuals and groups technically in charge of the system's overall architecture and engineering design. They must be fully conversant with the entire system. They outline the system's main components and supporting systems and how they work together.
d. The main stakeholders in the ART are business owners and customers. Owners of a business should consist of three to five people who are jointly responsible for the effectiveness.
e. Agile Teams receive assistance from System Teams in creating and maintaining development infrastructure, including continuous integration and solution testing. Teams using the SAFe System are not standalone units. They were included in the RTE. Additionally, they can attend events like System Demos.
f. DevOps establishes the deployment channel, enhancing process automation. They're part of the Agile teams, and DevOps is another crucial element of the ART.
g. Shared Services assists the team with specialized activities undelegated to ART, including business analysis and database administration. ART

compliance is essential for the shared service, even if they are not obliged to participate.

2.2.2.6.4 Program Increment (PI)

A version of the ART is the PI. What a sprint is to a Scrum Agile Team, it is to the ART. It is the period that passes between building and certifying a single system configuration increment; a single PI should last two to three months. Additionally, In SAFe, the most extensive planning block is the PI. The PI comprises tiny team iterations; this iteration equates to a sprint in a scrum. Iteration includes presentations, team planning, backlog adjustments, retrospectives, and daily meetings. Sessions in the PI are enlarged versions of team iteration meetings.

Meetings that are referred to as ART Sync meetings include Scrum of Scrums (SoS) and Product Owner (PO) Sync meetings. In addition, there are meetings for Release Management, System Demos, PI Planning, and review and change.

The most crucial meeting is PI Planning. It has a two-day agenda. The meeting gathers all necessary participants to plan the entire program iteration. A SoS should be conducted weekly at the least. Who should attend these workshops is not decided. No more than 30 minutes may be spent in each meeting. Barriers to completed tasks, synchronization, and other subjects should all be on the agenda.

Meetings for release management are held to control releases and notify management about them. The Release Management team is authorized to approve modifications to the release's scope, timing, and resources. Every second week of the PI, the system demo takes place, aiming to collect input from stakeholders while also assuring

effective integration of teams working on the same ART. PI Planning is a continuous process.

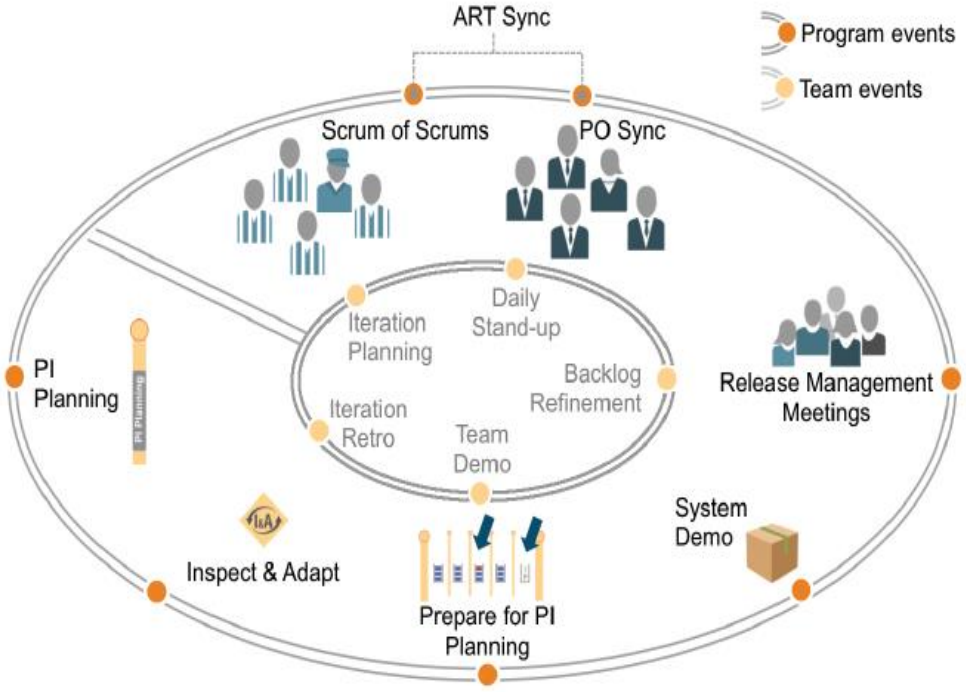


Figure 2.8: PI Kalenda, M. (2017)

The Scrum meeting serves as an enlarged retrospective. To speed up and enhance the correctness of the following Product Increment, all ART stakeholders should reflect, problem-solve, and carry out improvement actions.

2.2.2.7 Scrum

Lei et al. (2017), Scrum is a process that enables team members to handle complex adaptive issues while generating high-quality solutions in a timely and inventive manner. Scrum breaks down a problem into manageable pieces, leading to several iterations until the client's demands are met. Scrum is a lightweight framework that is basic and clear but might be challenging if not learned, according to Lei et al. (2017). According to Al-Saqqa et al. and Lei et al. (2017), Scrum encourages the development

of products through procedures, methods, and practice to give the most value possible while also enhancing communication between the scrum master, clients, and team members.

Each sprint lasts up to four (4) weeks, and the team launches a possibly deliverable software after each one. The tandem repeats these sprints until all of the product's features have been developed, which might take two or more sprints.

Lavanya (2020), Scrum uses the scientific technique of empiricism, which substitutes a programmed algorithmic approach with a more heuristic or self-learning one with respect for people and self-organization to handle uncertainty and solve complicated issues.

Working with facts, experience, and evidence is what empirical thinking entails. Scrum uses a practical method in which developments are based on observations of reality rather than made-up plans. To achieve business and organizational agility, Scrum also places a lot of focus on attitude and culture change. Scrum is an agile methodology that respects people and self-learning.

2.2.2.7.1 Pillars of Empiricism

a. Transparency

This entails delivering factual information and being honest with all parties engaged in the project, including the clients, the Scrum team, and the Scrum master.

Scrum bases essential decisions on how its three formal objects are viewed. Low transparency artifacts may influence decisions that lower value and raise the risk.

Sutherland and Schwaber (2011), Transparency makes it possible to inspect. Hence, inspection is misleading and ineffective without transparency (openness or integrity).

b. Inspection

It is essential to regularly and thoroughly review the Scrum artifacts and the progress made toward agreed-upon targets to spot any potential deviations or anomalies. An inspection enables flexibility. Hence, the key to scrum adaptability is quality and modifiable inspection

c. Adaptation

The scrum team uses the inspection results to improve processes and practices to maximize value delivery continuously. The key is learning from experience to improve productivity. The competence of the scrum team and stakeholders is critical for quality adaptation.

2.2.2.7.2 Roles in the Scrum

a. Product Owner/ Client

Sutherland and Schwaber (2011) stated that maximizing the worth of the product produced by the Scrum Team is the PO's responsibility. With Scrum Teams and people, this varies widely amongst firms.

The PO also manages the Product Backlog (PB) efficiently, which entails formulating and clearly articulating the Product Goal, producing and scheduling the PB, and ensuring that the PB is transparent, visible, and understandable.

The PO or Business Analyst may carry out the responsibilities above. The PO is in charge of making critical decisions that affect the project's success.

b. Scrum Team

A scrum team is the fundamental unit of the scrum. A Scrum Master, PO, developers, and testers make up the Scrum Team; in most Scrum teams, the developers and testers are typically the same persons. There are no sub-teams or hierarchies in a Scrum Team. It is a well-coordinated and cross-functional team of specialists who are each singularly focused on the Product Goal. They also choose who does what, when, and how since they are self-managing.

Sutherland and Schwaber (2011) stated that the Scrum Team, which typically has a maximum of 10 members, is agile and big enough to complete a lot of work in a sprint. We've found that smaller teams work more effectively and communicate more effectively. If Scrum Teams get too big, they should consider dividing them into several cohesive Scrum Teams, each concentrating on a single product. They ought to have the same Product Goal, Backlog, and Client as a consequence.

The Scrum Team's focus and consistency are improved by working in Sprints at a constant pace.

c. Scrum Master (SM)

SM leader is in charge of enforcing and overseeing the scrum rules and values throughout the project (Al-Saqqa et al., 2020).

The scrum master conducts meetings, interacts with clients and management outside of the team, and keeps the sprint free from outside interference. Lei et al. (2017) stated that the scrum master also keeps track of work against the backlog and ensures the team is operating at peak efficiency by removing obstacles and giving the tools required to maintain productivity.

Scrum Masters are real leaders that assist the Scrum Team and the entire business.

The Scrum Master helps the Scrum Team in many different ways, such as coaching team members in self-management and cross-functionality, helping the Scrum Team focus on producing high-value increments that match the Definition of Done, causing obstacles to the team's progress to be eliminated or minimized, and ensuring all events occurs according to schedule and are fruitful (Sutherland & Schwaber 2011).

The Scrum Master helps the PO in a variety of ways, including by helping to identify the best practices for creating Product Value. The Scrum Team's awareness of the need for clarity and conciseness is supported by PB management. Items in the PB that help develop empirical product planning for a complex environment and provide stakeholder participation as desired or necessary.

The Scrum Master assists the organization in several ways, such as leading, educating, and coaching it in adopting Scrum; organizing and providing advice on Scrum implementations within the organization; helping Employee members and stakeholders understand and put into practice an empirical approach to complex work; and removing obstacles between stakeholders and Scrum Teams.

2.2.2.7.3 Scrum Lifecycle and Scrum Ceremonies/Events

The Sprint is the main event in the scrum lifecycle. Each Scrum event offers a formal opportunity to assess and amend Scrum items. These events are specifically designed to foster essential transparency. These events are organized to reduce non-scrum meetings to ensure high productivity.

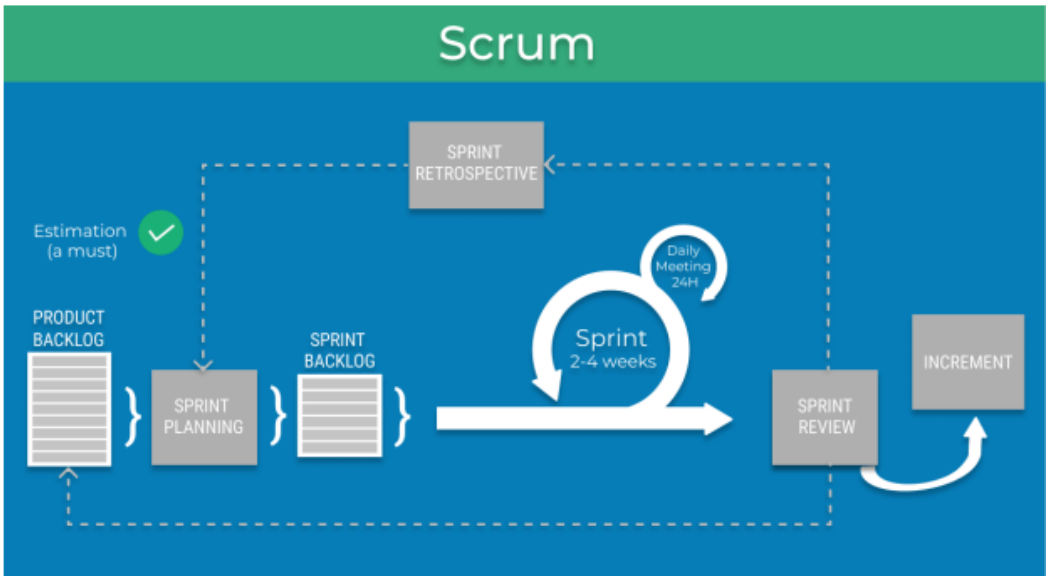


Figure 2.9: Scrum lifecycle

a. Sprint

Sprint is a critical activity in the scrum in which goals, ideas, and plans are converted to value (Sutherland and Schwaber 2011). Sprint usually lasts one month or less, and as one Sprint ends, a new one starts. All of the tasks necessary to accomplish the product goal, including sprint planning, sprint reviews, daily scrums, and sprint retrospectives, are included in interval training.

The PB is adjusted as necessary, no modifications are made throughout the Sprint that would jeopardize the Sprint Goal, and when sufficiently much progress has been made, the scope adjustment is renegotiated with the PO. Sprints increase predictability by mandating a minimum once-per-month evaluation and adjusting progress toward a Product Goal.

b. Sprint Planning

Sutherland and Schwaber (2011) stated that Sprint Planning is the first activity in the sprint, which details the tasks that will be accomplished during the Sprint. The PO makes sure everyone is ready to discuss the most critical PB items and how they relate to the Product Goal, thereby enhancing the sprint planning process.

According to Lavanya (2020), The Scrum Team is responsible for the planning process, and they may consult other stakeholders to aid in the Sprint Planning process. Sprint planning is aligning the Sprint goal to the delivery strategy. The Sprint goal is achieving the requirement for the PB items selected for the Sprint, and the delivery strategy is all included in the Sprint Backlog. Sprint Planning is restricted to eight hours for a one-month Sprint, while shorter ones require less time.

According to Sutherland and Schwaber (2011), the scrum team should consider the following queries when planning a sprint.

a. What is possible, given the next sprint increment?
b. How much work is necessary to create the increment?
c. When is a piece of work deemed "Done"?

c. Daily Scrum

Sutherland and Schwaber (2011) stated that reviewing the Sprint Goal's progress and, if necessary, making changes to the Sprint Backlog (SB) to change the subsequent planned work are the objectives of the Daily Scrum. The Scrum Team's developers meet daily for 15 minutes for a scrum meeting. It takes place simultaneously and places every working day of the Sprint to reduce complexity.

According to Lavanya (2020), Daily Scrum meetings facilitate faster decision-making, improve communication, and eliminate the need for following sessions.

d. Sprint Review

The goal of the Sprint Review is to evaluate the results of the Sprint and make decisions about future modifications. According to Lavanya (2020), the Scrum Team discusses project progress and presents project results to essential stakeholders. The Scrum Team and stakeholders assess what was accomplished throughout the Sprint and how their environment has evolved throughout the event; based on this information, participants collaborate on what to do.

The Scrum Team should refrain from making the Sprint Review exclusively on a presentation because it is a working session. The Sprint Review is the second to last event of the Sprint, and it has a four-hour time restriction for a one-month Sprint. For shorter Sprints, the event is often faster.

e. Sprint Retrospective

According to Lavanya (2020), this is a critical event in the scrum that occurs at the end of every sprint. The Scrum Team reflects on the previously completed sprint and develops plans for enhancing efficacy and quality. The Sprint is concluded with the Sprint Retrospective and timebox to a limit of three (3) hours for a four-week sprint.

2.2.2.7.4 Scrum Artifacts

In 2011, Sutherland and Schwaber published Scrum's artifacts representing value or effort. They are designed to increase the transparency of important information. These scholars also stated that the scrum artifacts are used to standardize evaluation and

change management. Every artifact commits to providing data that increases transparency and focus, against which advancement can be gauged:

Sutherland and Schwaber (2011) stated that the PB is used to access progress on the Product Goal, the SB is used for the Sprint Goal, and the Increment is used to access the Definition of Done. These commitments for the Scrum Team and its stakeholders promote empiricism and Scrum principles.

1) Product Backlog (PB)

A PB is a prioritized list of what needs to be done to improve the product. It is the Scrum Team's primary source of work. It provides user stories for the whole creation; the client controls the list's content and hierarchy.

a) PB Refinements

Adding information to items on the PB is known as PB refinements. This is a continuous process in which the PO and development team work together on the specifics of the items on the PB (Sutherland and Schwaber, 2011).

2) Sprint Backlog (SB)

The development team is forecasting what functionality will be included in the upcoming increment and how much work will be needed to deliver the feature in a full iteration. This makes all the work the development team determined necessary to complete the sprint aim and facilitate ongoing progress. It includes a minimal prioritized process enhancement mentioned in the most recent meeting and is detailed and sufficient to explain adaptation in the daily scrum's progress.

Sutherland and Schwaber (2011), The development team changes the SB only during the sprint. The work the development team plans to complete during the sprint is evident in the spring backlog.

3) Increment

This is the sum of all PB items finished in a sprint and all previous sprints. Each sprint must end with the creation of a new increment. It is done as accomplished in line with the requirements set out by the scrum team.

According to the Definition of Done, Sutherland and Schwaber (2011) highlighted that the Increment is at its "done" state when it satisfies the quality metrics necessary for the product. A PB item receives an Increment when it meets the Definition of Done.

Lavanya (2020) explained that the Definition of Done encourages transparency by providing everyone with a shared understanding of the work completed as part of the Increment. A PB item cannot be published or even presented at the Sprint Review if it does not adhere to the Definition of Done.

Lavanya (2020) states that all team members must understand the items classified as done. This scholar also highlights that the key to a successful scrum process and activities is following the principle of empiricism

2.3 Summary

The summary of this chapter comprises the general issues with the Software Development Project, the traditional approach (waterfall model), and various agile software development models. Models such as XP, FDD, TDD, KABAN, and SCRUM highlight their life cycles, roles, and advantages and disadvantages. An in-

depth overview of the usability and perceived quality of EMS is provided in subsequent chapters. In addition, the research hypotheses were established and extensively discussed.

Chapter 3

THE DEVELOPED SYSTEM AND RESEARCH HYPOTHESIS

The development of an EMS using the scrum methodology for a design case study for this system and the hypotheses proposed and critical variables are discussed in this chapter.

Due to Scrsum's innovative approach in which adaptability and change are possible, software projects are embarked on in an organized and adaptable manner. According to Sutherland and Schwaber (2011), the pillar of Empiricism enhances the innovation and efficiency of the development process.

The first part of this chapter is detailed information on the implementation of scrum methodology and software overview, which includes hardware and software requirements, software design and architecture, database design, and software functionality. The second part would be the proposed hypothesis and discussion on critical terms.

3.1 Implementation of Scrum Methodology on Design Case Study

Before implementing this process, the first step is to get a clear understanding of project data, thereby determining the scope and limitations of the proposed case software project development and identifying the stakeholders and the project timeline.

The design case was developing an EMS for an SME business in Nigeria, and due to time and cost constraints, a comprehensive employee system was not designed; hence, minimal features based on PO-defined requirements in the PB in the form of user stories.

3.1.1 Scrum Phase for EMS

The initiation phase is a phase where the objectives and visions of the EMS are defined and aligned with the business and organizational needs. This phase is the appointment and selection of Scrum team members, which consists of the PO, SM, and Development Team (Bica & da Silva 2020). The PO defines, prioritizes, and maintains the PB items. At the same time, the SM is responsible for ensuring that the Scrum team produces a valuable product, and The Development Team is responsible for developing the project. The following objectives for the EMS were highlighted in this phase: an apt leave, task and salary management process, and effective employee login tracking. Due to cost constraints, the PO and the SM responsibility were handled by one person.

The planning stage is the most important in designing any software development project, as it increases the chances of success (Tam, da Costa Moura, Oliveira, & Varajão 2020). Planning the project to the available timescale is the goal of effective planning (Stober, Suškevičs, Eiter, Müller, Martinát & Buchecker, 2021). The PO creates and prioritizes the PB items after stakeholder collaboration in this phase. The PB for EMS contains lists of all the features, functionalities, tasks, and requirements that need to be implemented in the system in the form of user stories. PB includes all necessary data required for the completion of the EMS.

Also, in this phase, the sprint planning for the upcoming sprint, the SM and scrum team conduct a meeting to select data from the item in the PB for the SB item; the chosen items are used to know what would be in progress during the sprint. Sprint planning identifies tasks that would be engaged during the sprint.

The implementation phase is the phase in which the sprint is active; the Development team works on the selected items on the SB during the sprint, which are functionality and features in the form of user stories, as shown in Figure 3.1 below, and also during the sprint in the development of the EMS Scrum team have a daily scrum meeting, a stand-up meeting for 15 to 30 minutes to ascertain daily progress and obstacles. Monitoring and managing software projects are complex and challenging without using project management software such as Asana, Trello, MS Project, JIRA, and so on. Hence, the use of JIRA software, an agile project management tool developed by Atlassian, was an effective tool used for recording the PB and SB and monitoring and managing the project as a whole. As shown in Figure 3.1 below, items in the board are in the SB segmented into three statuses: "To Do, In Progress and Done." Items on the "To Do" are items in the SB that are on the query to be worked on; Items in the In Progress are currently being worked on; And finally, items in the "Done" are considered to be completed.

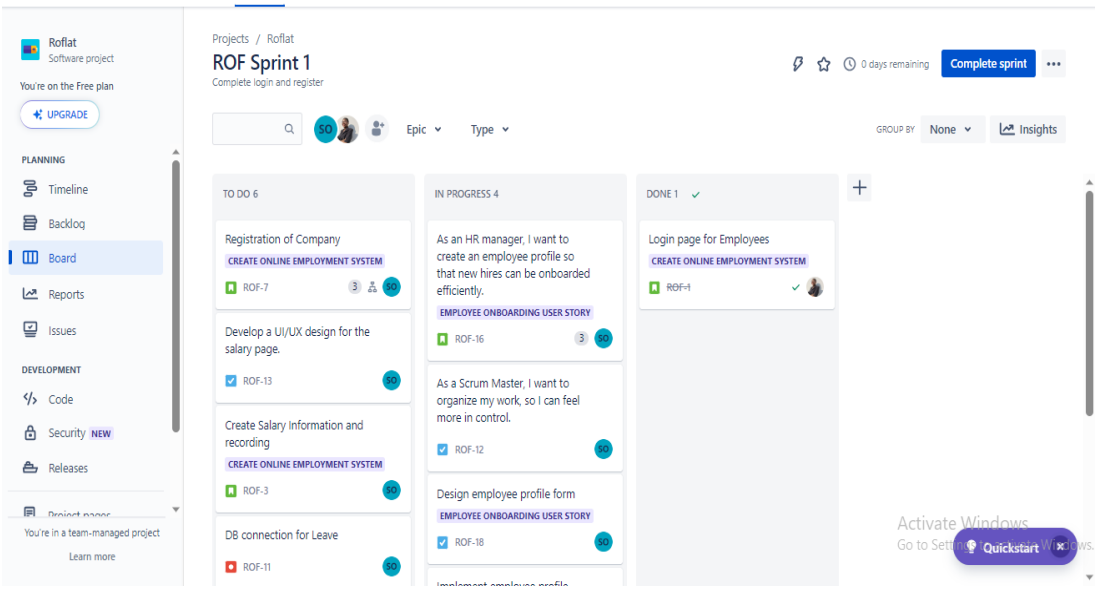


Figure 3.1: Jira software project's management tool

Jira Software helps scrum teams for efficient and effective collaboration between team members and also grants the team a well-structured way to track and monitor the sprint and the whole project. Jira software allows users to store backlog items (PB and SB items) in the backlog page when the backlog link is the sidebar clicked in Figure 3.1 above.

The review phase is the phase in which the scrum team reviews the output of the currently completed sprint aligned with the PB to ascertain if there is a need for PB refinement (i.e., an adjustment to the PB). The PO and other stakeholders confirm the PB refinement at the end of each sprint.

The release phase is the phase in which the release is planned to identify shippable features for the PB called increments, and the PO and other stakeholders ascertain product quality to the set standard; after that, there is a release of these increments.

This phase also involves a sprint retrospective for the team to reflect on ideas for improvement for future sprints.

3.1.2 Software Data Model

The system is called ROFLAT. It is used to manage employee information, i.e., registration, login, task allocation, and leave application and approval.

The model requires an email address and password to log in as an employee or admin (manager). Upon registration, the user automatically sends a confirmation email to the user, and only the admin can register an employee. For verification of online employees' timestamps and transaction data are implemented.

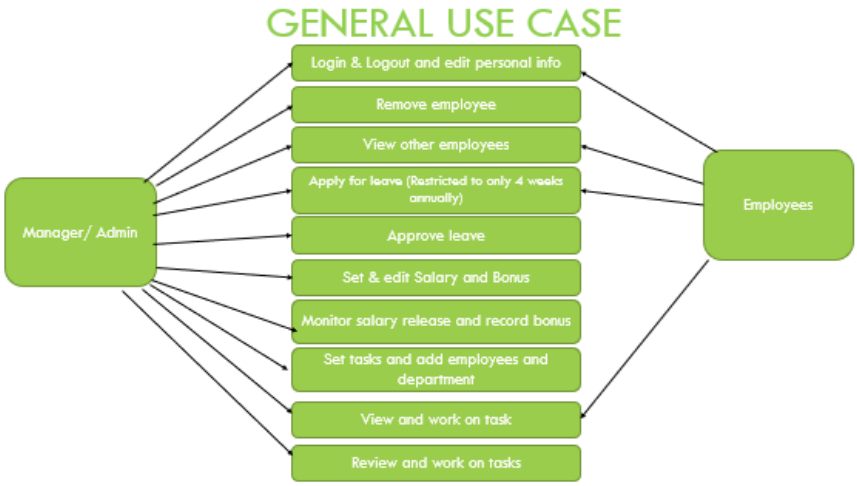


Figure 3.2: General Use Case

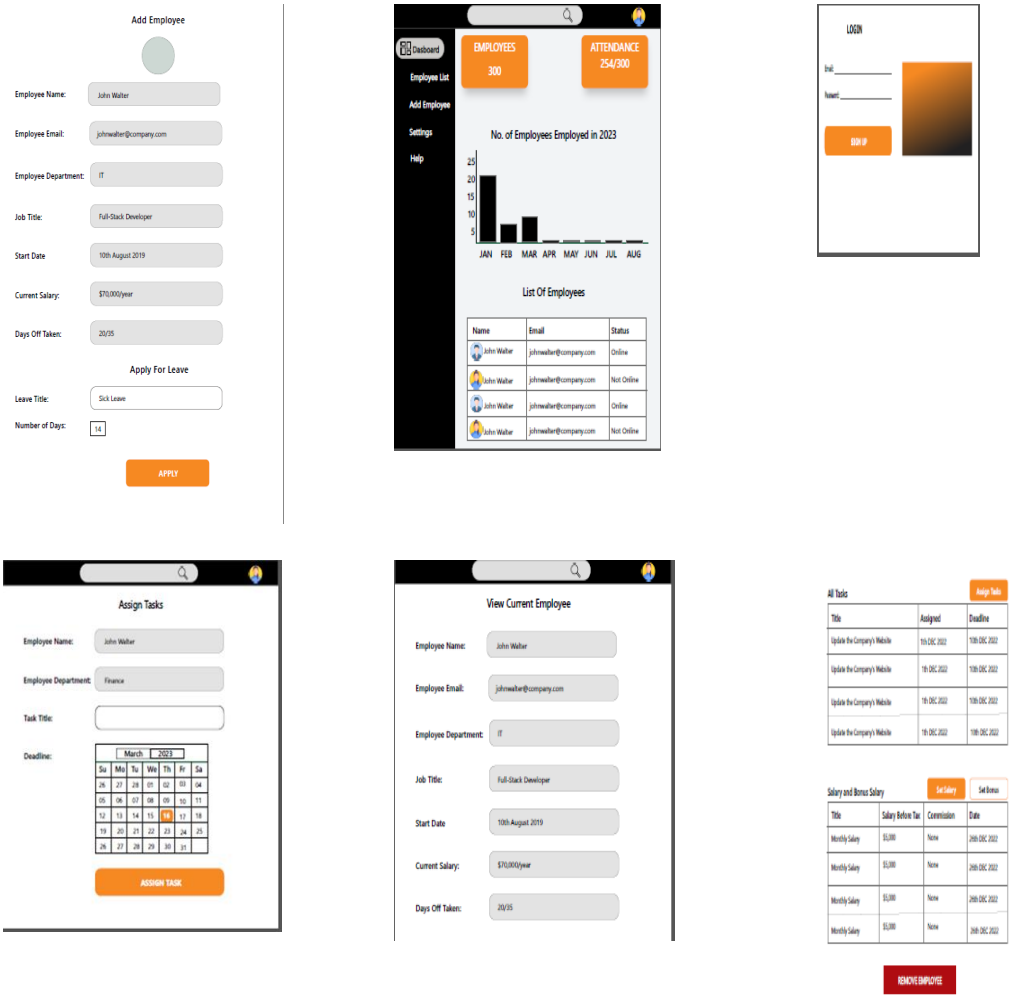


Figure 3.3: Roflat Software Interface

The application is divided into two views for easy development and maintenance.

1. The admin or manager view is a professional view in which the system is controlled and managed. In which managers register employees, set tasks, approve leave, etc. This view is shown in Figure 3.2 below.
2. The employee view has a friendly interface that makes it easy to use and understand.

3.1.3 Hardware and Software Requirements

The Visual Studio code (VS code) text editor was used throughout the software application's development. A detailed explanation is shown in Tables 3.1 – 3.2 below

Table 3.1: General summary of software requirements

Environments Supported	Version
VS Code	Version 4.6.0 and higher
JavaScript (JS)	Version 5.6.0 and higher
React JS	Version 5.6.0 and higher
Node JS	Version 5.6.0 and higher
JIRA Software	Version 4.0 and higher

Table 3.2: General summary of hardware requirements

Environments Supported	Version
Central Processing Unit	Intel i5 or higher
Random Access Memory	8 GB or Higher
Hard Disk Drive (HDD)	128 GB or higher

3.1.4 Programming Language

The application was developed using React JavaScript (React JS), a JS library that builds applications using user interfaces based on components. Bootstraps (BS) were used for the design of the application. BS is a framework design using Cascading Style Sheets (CSS), Hyper Text Markup Language (HTML), and JS.

3.1.5 System Architecture

The program was released in version one, which was tested and utilized by various users. The system is built on a client/server architecture to support future updates. To run this technology at a large scale for SMEs, detailed architectural design is required to guarantee that the EMS complies with the agile scrum process and adapts to change elegantly, fostering innovation. That data is not exposed to unauthorized access.

3.1.6 Software Design Architecture

The software application employs a Model View Controller (MVC) design architecture. This design pattern divides the program into three logical components: model, view, and controller (MVC). The project used this design to segregate the business logic from the presentation layer. The view component is used for user interaction and is controlled by the controller. The controller changes the Model, which alerts the view. This approach enables simple modifications or updates to the model, which then alerts the view. Utilizing this method allows for simple system adjustments or updates. The system's current design architecture is depicted in the diagram below.

3.1.7 Database Management System (DBMS)

The software development application uses Google’s cloud Firestore, a real-time NoSQL database that stores data in JSON format and allows real-time updates to be made to the data. Cloud Firestore aided in structuring the data into documents and collections and querying the data in real time. Firebase provides a powerful and flexible database management system that helps build high-quality, scalable applications quickly and easily.

Google also provides several tools and features that help manage the database, including the Firebase Console, which allows developers to view and manage their data in real time, and the Firebase Software Development Kit (SDK), which provides

a set of APIs for interacting with the database from client-side code. The database structure is shown below in Figure 3.3.

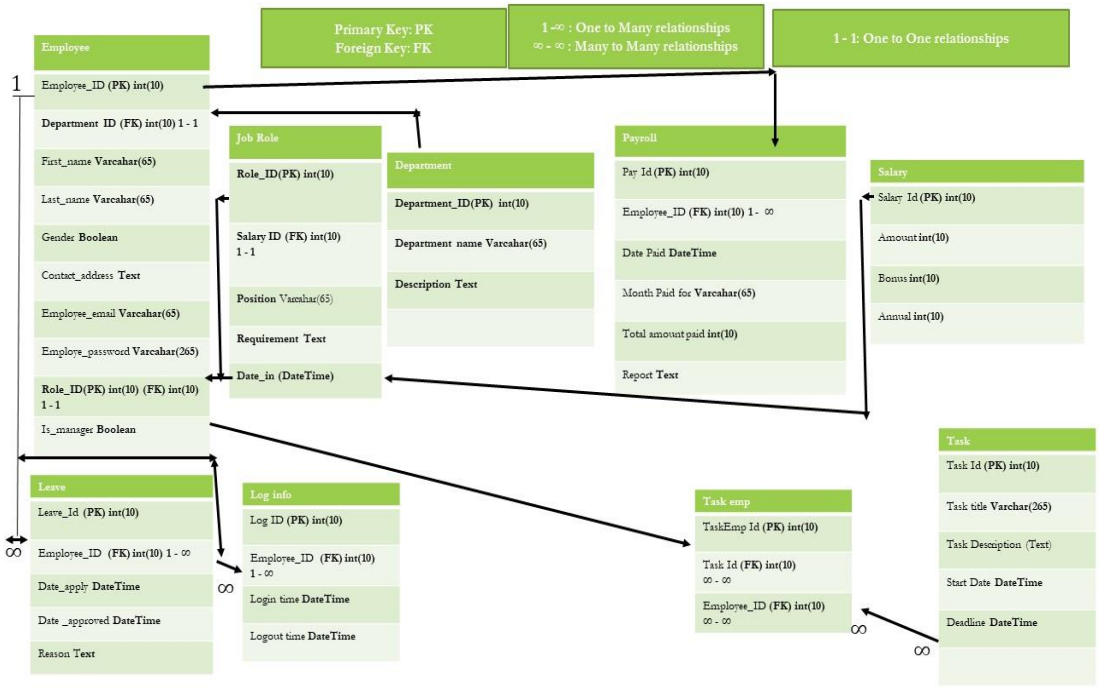


Figure 3.4: Database Management System (DBMS)

According to Amin, Romney, Dey, and Sinha (2019), it is crucial to normalize the database to trim down redundancy and also make data maintenance less complex. These scholars also noted that over-normalization could lead to query delay and performance inefficiency due to complex joins; hence, selecting the appropriate normalization level based on EMS requirement specification is paramount. The normalization of the EMS is restricted to the third normalization level, the Third Normal Form (3NF). Hence, each table has a primary key (PK) to identify them uniquely. Each table data is indivisible, and no repeated array or group in a cell (i.e., full name is broken down to first, last, and middle name) adhering to the First Normal Form (1NF). The Second Normal Form (2NF) states that all non-key attributes must

depend on the primary key, hence the need for a Foreign Key (FK) to establish these relationships, which could be one-to-one, one-to-many, or many-to-many. Lastly, the 3NF as salary and bonus are related as the bonus depends on salary. Hence, the need for a salary table as a non-key attribute can rely on another (Rolik, Amons, Ulianytska & Kolesnik, 2021)

3.1.8 Software Maintenance and Security

Google implements various security measures that protect users' data, including authentication and authorization mechanisms, encryption, and monitoring tools, which were integrated into the system. Moreover, it offers several security features, like Secure Sockets Layer (SSL) support, Firebase Security Rules, and two-factor authentication, to further safeguard the system.

We are adhering to Google's best practices using the most recent Firebase SSDK version and monitoring your application for security vulnerabilities.

3.1.9 System Functional Requirements

An EMS is a valuable tool that helps businesses to track employee statistics, performance, and other pertinent information. A concise outline of the system's functional requirements:

Table 3.3: System Functional Requirements

System	Description
Employee Information Management	The system shall allow the manager to register employee information, such as personal information, job titles, job descriptions, etc. Hence granting employees login access.

Leave Management	The system shall allow Employee leave application and approval and, such as monitoring vacation time, sick leave, and other sorts of time off.
Task Management	The system shall allow assigning tasks, measuring progress, and offering feedback
Attendance Management	The system shall allow employee attendance management, including tracking absences and tardiness.
Payroll Management	The system shall allow employee payroll management, including tracking salaries, deductions, and benefits.
Data Analytics	The system shall allow for data analytics, including generating reports and insights on employee performance, attendance, payroll, and other relevant metrics.
Security and Access Control	The system shall ensure that employee information is secure, with access only granted to authorized personnel.
Employee Communication	The system shall allow employee communication, including sending company-wide announcements, individual messages, and reminders.

3.1.10 System Non-Functional Requirements

Based on general acceptable Non-functional requirements for application development, the developed system adheres to this requirement as stated below.

Table 3.4: System Non-Functional Requirements

Performance	Roflat is quick and responsive, with little to no lag time when executing activities
Scalability	Roflat can manage enormous volumes of data and users while being performant.
Reliability	Roflat is dependable, with high availability and little downtime.
Security	Roflat is secure, with sufficient safeguards to prevent unauthorized access to employee data.
Usability	Roflat is simple to use, with a simple user interface and straightforward instructions
Accessibility	Roflat shall be used by Only employees of the registered company
Compatibility	Roflat shall work with a wide range of devices and operating systems.
Maintainability	The Roflat system shall be simple to maintain, with well-organized code and comprehensive documentation.
Interoperability	The Roflat system shall be able to communicate and exchange data with other systems and applications.
Privacy	The system shall respect employee privacy appropriately to ensure confidentiality and data protection.

3.1.11 Ethics and Social Impact of EMS

The EMS prioritizes data security and privacy, ensuring that employees and admin information are stored and encrypted. Hence, access is only restricted to authorized users, and users also have access levels, which are limited based on the admin(manager) level or user level. Consent and confirmation mail is sent to the user to validate data via e-mail.

The EMS enhances employee productivity and appropriate segregation of duty via task management, and it also enhances attendance tracking and leave management for managers. According to Adekoya, Adisa, and Aiyenitaju (2020), there has been a continuous rise in remote work post covid. Hence, the EMS enhances flexibility via remote work.

3.2 The Element of Usability and Perceived Quality of the EMS

To access the usability and perceived quality of the EMS, use the following elements: usability, efficiency, affect, helpfulness, control, and learnability. Perceived quality is a user's subjective perception or assessment of the EMS's usability based on their expectations and experience with the EMS. The Scrum team prioritizes continuously evaluating and improving the EMS's usability and perceived quality to meet users' demands.

3.2.1 Usability

This is the ability of a software product to be viewed, understood, and used in a way that is appealing to the user when used under certain conditions. Good usability has three essential characteristics:

a. The first is efficiency, which is the degree to which user goals or objectives are met by the resources employed (i.e., the EMS).
b. Second, effectiveness, or how successfully software assists users in achieving their objectives.
c. Third, satisfaction is the degree to which users conveniently utilize the EMS.

The management system is an effective way to increase productivity and track employee management, task management, and leave management.

3.2.2 Efficiency

The exactitude and completeness with which users achieve their goals and the quantity of effort required to attain those goals are characterized as efficiency. Users perceptions of EMS efficiency are inextricably linked to efforts to increase usability.

3.2.3 Affect

The statistic assesses how satisfied the user is with the program in general. User experience and software interest are both markers of strong positive impact. The improvement in software usability is a direct outcome of the software aligning to the user-specified program backlog. The clarity of its intended function, as well as the simplicity with which it can be comprehended via the navigation bar, both contribute to a greater adoption rate.

3.2.4 Helpfulness

This refers to the users' perception that the EMS delivers constructive guidance, feedback, and information via a user-friendly interface. The level of functional and technical help the EMS provides enhances usability.

3.2.5 Controllability

This refers to the level of perceived control of users over the EMS. Usability must assess users' perception of the level of control when utilizing the EMS.

3.2.6 Learnability

The learnability is a measure of how adaptable and speed in which users learn from the developed EMS. Learnability is a factor that affects the level of usability of the EMS

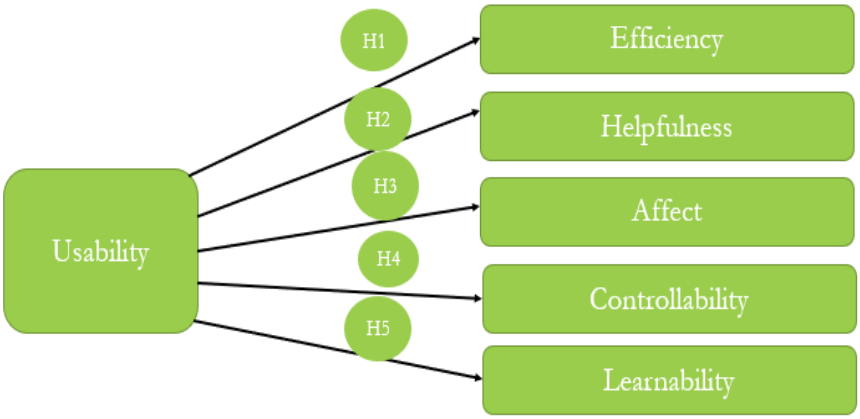


Figure 3.5: EMS Usability and Its Relationship with Other Variables

3.3 SUMI for Measuring Usability

Usability is essential to product design and development since it impacts how well consumers may achieve their goals using a product. Khairunisa, Tyas, Purwanto, and Aisyah (2020) employed SUMI to assess a student information academic system in their study. The responders were students enrolled in the multimedia engineering program. The SUMI evaluation findings demonstrated that the system was usable and fulfilled the SUMI assessment requirements. The study's conclusions were consistent with those of Darmawan et al. (2021), who employed SUMI as an assessment technique to assess the usability of the mobile-based Smart Regency Information System. The approach used in this study is based on the Knowledge Management System's Life Cycle (KMSLC).

Similarly, Khairunisa et al. (2020) use SUMI to assess the usability of PS, The Evolved Integrated Model (TEIM) of Software for Engineering and human-computer Interaction. Zaini et al. (2021) utilized SUMI to evaluate a Child Immunization Schedule Application. The SUMI study results revealed that users were enthusiastic about the program, and it was judged usable.

According to the authors, the study's findings will give valuable insights into the aspects that influence the effectiveness of knowledge management for innovative regency services. However, to our knowledge, SUMI has been employed as an assessment method for usability from the consumers' perspective for EMS.

3.4 Hypothesis Development

In light of the above discussions, six hypotheses are developed:

H_1: The EMS efficiency resolves employee management issues.

H_2: Users will find the EMS helpful in carrying out operations.

H_3: The EMS’s potential to elicit an emotional reaction influences the user's emotional state during engagement.

H_4: A user's sense of control over the program influences their perception of the EMS's reactivity and stability in reacting to their data inputs and instructions.

H_5: The learnability of EMS determines how readily individuals use it.

H_6: The EMS, designed with agile scrum software development, will result in a better overall user experience.

Chapter 4

RESEARCH METHODOLOGY

Describe, analyze, apply, statistical, descriptive, and investigative research can all be utilized successfully to achieve research objectives (Mohajan, 2020). This research will look at the perceived quality and usefulness of the system. This chapter will address the data gathering strategies, data gathering, research population, validity, data analysis strategy, and ethical approach in this study section.

4.1 Research Design

Research design may be quantitative or qualitative, determined based on the investigation being undertaken. According to Chigbu (2019), Quantitative research is gathering and analyzing numerical data to answer research questions or test hypotheses. It explores and interprets data using statistical methods and mathematical models. Quantitative and qualitative research share similarities and differences.

4.2 Population Sampling

The goal is to select a reasonable number of employees from an IT SME in Nigeria's population because a survey of the entire population is not feasible owing to time and resource constraints. For this study, purposive sampling was used. According to Etikan, Musa, and Alkassim (2016), purposive sampling, also known as selective or judgmental sampling, is the purposeful selection of individuals with specific traits or experiences relevant to the study's aims. This strategy would be utilized when the researcher needs particular expertise to carry out research

According to Kirakowski (1986), A minimum sample size of 10-12 participants is necessary for suitable outcomes in the SUMI assessment. However, a sample of 34 respondents is required to complete the survey in this study. A self-selection sampling strategy was employed in the investigation. Participants are employees of an IT SME in Nigeria who were contacted via mail.

4.3 Data Gathering Strategies

Gathering data from all reputable sources is the initial stage in addressing the study issue, testing the hypotheses, and analyzing the findings. There are two sorts of data collection methodologies: secondary and primary. This study's survey data is preliminary data acquired using an online survey. This guarantees that the data is easily accessible for future usage.

SUMI is a highly effective tool for evaluating software usage on a global rating. SUMI is a 50-question survey that includes the following: efficiency, Controllability, affect, learnability, and helpfulness (Kirakowski, 1986).

The questionnaire was developed in 1986 by Kirakowski, and it has subsequently been used in other usability studies. The survey also includes a global usability scale, which gauges the user's overall satisfaction with the tool. To express their ratings about the EMS, participants select one of three alternatives ("agree," "undecided," or "disagree"). Afzal (2023) and Tekinerdogan et al. (2016) noted that ISO/IEC 9126 is established to assure the quality of all sensitive software-intensive goods in which its failure may result in the loss of life or resources. SUMI is a suitable assessment instrument for evaluating quality and user experience (Richardson, Campbell-Yeo, &

Smit 2021). Hence, SUMI was used to appraise the EMS usability and perceived quality.

4.4 Administration of Questionnaires and Ethical Considerations

After the approval of the questionnaire by the ethics board, participants were fully informed about the purpose and nature of the survey alongside the link to the application developed, and their consent was obtained. Confidentiality and anonymity were preserved throughout the procedure, and all data obtained was securely archived. The participants' rights were respected, and no harm or coercion was inflicted. This study strictly adhered to the ethical guidelines of EMU's Ethical code of conduct, ensuring that the survey results accurately reflect the views and experiences of the respondents. Clearly expressing the use of respondents' data and their understanding that their data will be rendered anonymous and may be utilized for future research

4.5 Validity

The validity of SUMI entails a thorough assessment of the instrument's components, statistical analysis to discover the underlying constructs and aspects of software usability, and comparing SUMI scores to other measures of software usability. Darmawan et al. (2021) confirm that the instrument correctly assesses software usability and may be used to build effective methods for enhancing software usability by establishing validity using SUMI. Validity investigations have been conducted on SUMI. The Human Factors Research Group's usage of SUMI to execute a variety of laboratory-based research projects as well as consulting studies for corporate clients.

4.6 Data Analysis Strategy

SUMISCO is a sub-tool in SUMI in which the standardized database in which respondent input is stored and analyzed. The standardization database is used as a baseline for evaluating software usability. A score of 45-65 is considered appropriate,

while less than 45 is considered inadequate. SUMI is a worldwide recognized standard for analyzing the usability of software. It is a 50-question survey that evaluates five (5) unique characteristics: helpfulness, Controllability, efficiency, affect, and learnability. The Item Consensual Analysis (ICA) tool in SUMI allows the computation of the actual percentage of replies from the evaluation sample and the statistically predicted proportion based on the standardization data. This evaluation enables distinguishing software dimensions that are considerably better or worse than the worldwide software standard. The SUMI website is available at http://sumi.uxp.ie/.

Chapter 5

DATA ANALYSIS

5.1 SUMI Setup

The use of the SUMI questionnaire for the survey to evaluate participants' perceived quality and the usability of the EMS. The administration of the questionnaire was carried out in two stages. Firstly, consent from participants was obtained; afterward, participants completed the questionnaire online.

SUMI is an internationally standardized usability measurement tool consisting of a 50-item questionnaire to evaluate the participants' perceived quality and usability of the developed system. Participants were offered three choices for their responses: "agree," "undecided," or "disagree." After completing the questionnaire, the SUMISCO software, a subsystem of the SUMI evaluation suite, was used to calculate the scores and compare them to a standardized database. The standardized database set a mean score of 50 and a standard deviation 10. EMS with SUMI scores of 40-55 is deemed standard usability compared to most commercially successful products in the standardized database (scores above and below this range are included). A higher score implies more usefulness, whereas a lower number emphasizes improvement.

5.2 SUMI Usability Components

The use of the SUMI questionnaire to measure the system's usability and users' perceived quality of the system. The SUMI assessment is a valid method for assessing users' perceived quality of a system, with an integrated analysis and reporting tool

supported by a vast reference database (Van der Linde, Wessels, & Kirakowski, 2013; Jeddi, Nabovati, Bigham, & Khajouei, 2020). According to Darmawan et al. (2021), the user experience may be separated into five categories; hence, the final report is divided into a global scale and five SUMI-defined sub-scales.

a. **Efficiency:** This is the user's perception of the program's ability to accomplish the task or tasks in a timely, accurate, and cost-effective manner or, conversely, that the software impedes the users' performance.
b. **Affect:** This is the user's perception of the system's ability to influence the emotional feelings and experience of the user**.**
c. **Helpfulness: T**his is the user's perception of how the system helps users solve practical problems.
d. **Controllability:** The user's perception of how the system responds to commands and inputs appropriately and consistently.
e. **Learnability:** This is the user's perception of the degree to which the system was easy to operate and learn from

5.3 Analysis Result For EMS

This section summarizes the result of the EMS usability evaluation using SUMI. Respondents, employees in an IT SME in Nigeria, were asked to test the developed system. The online survey was completed by 34 participants, exceeding the SUMI minimum of 12 participants. Of the 34 participants, 30 had trustworthy responses that could be examined, while the remaining 4 had invalid answers.

Table 5.1 displays data for the Global and individual SUMI scale scores, organized in ascending order by Global score. Individual subscales typically have maximum scores

of approximately 70 out of a range of 100, which indicates a high score for the subscale, and lowest scores of around 37 out of a range of 100, which means a low or poor score for the subscale. A score above 50 indicates that the subscale score is above average, and a score below 50 suggests that the participation score for the subscale is below average.

Tables 5.1 Participant Global and individual SUMI scale scores (Overall score of 100)

Participant	**Global**	**Efficiency**	**Affect**	**Helpfulness**	**Controllability**	**Learnability**
1	70	60	69	67	72	69
2	70	54	51	70	62	46
3	68	54	56	65	53	57
4	64	58	58	58	65	49
5	63	53	64	62	56	42
6	62	55	52	57	69	50
7	61	57	62	57	59	55
8	61	44	58	58	56	43
9	59	45	56	61	59	41
10	58	62	45	55	60	52
11	58	43	58	56	59	49
12	57	37	55	60	55	35
13	56	54	56	55	58	50
14	55	37	55	60	52	35
15	55	52	61	50	57	49
16	55	35	42	59	52	42

17	53	58	54	55	43	52
18	52	39	53	55	52	38
19	51	43	58	49	55	58
20	50	39	52	53	50	42
21	49	64	44	48	56	48
22	47	43	41	52	42	63
23	47	46	43	45	42	49
24	46	43	41	49	43	46
25	45	52	47	37	49	55
26	43	41	38	41	47	64
27	40	55	46	38	40	49
28	39	42	55	38	42	52
29	39	47	37	39	42	50
30	37	53	37	39	34	59

When SUMI scores are converted with a z-score, the average mean is 50 with a standard deviation 10. This means that ratings above 50 imply above-average consumer satisfaction. According to Table 5.2, the EMS received a global score of 53.67; this falls above the range expected and suggests that the level of user contentment using the developed system is above average, thus confirming H6, implying that the EMS designed with agile scrum software development can result in a better overall user experience

The table also displays the descriptive statistics, including mean, median, standard deviation, interquartile range (IQR), minimum and maximum scores for each individual, and the overall scale. A statistical analysis of the data reveals that the Helpfulness sub-scale obtained the highest score of 52.93, implying that the EMS aid user in effectively carrying out task and operation, thus validating H2. In contrast, the Efficiency and learnability sub-scales are slightly below 50, which indicates users perceive the EMS to be less efficient and a bit complex to understand, thus nullifying H_1 and H_5. All other sub-scales obtained mean scores that exceeded 50, which indicates overall user satisfaction across the various aspects of the evaluation and validating H_3 and H_4. The median, as calculated from the data arranged in numerical order, is represented by the median boxplot in Figure 5.3.

Table 5.2: SUMI Result Breakdown

	Mean	**St Dev**	**Median**	**IQR**	**Minimum**	**Maximum**
Efficiency (H_1)	48.83	8.16	49.5	12.0	35	64
Helpfulness (H_2)	52.93	9.07	55.0	11.0	37	70
Affect (H_3)	51.47	8.46	53.5	14.0	37	69
Controllability (H_4)	52.70	9.00	54.0	16.0	34	72
Learnability (H_5)	49.63	8.17	49.0	12.0	35	69
Global (H_6)	53.67	9.24	55.0	14.0	37	70

Figure 5.1 is a bar graph illustrating the mean scores obtained on the SUMI evaluation's various subscales. Each bar's height indicates the average score for a

certain subscale. The bar becomes green when the average score surpasses 50 (the reference point). If the average score goes below 50, the bar becomes red. As shown in Figure 1 below, efficiency displays a red bar because it is below the reference point of 50. The graph also shows the range of scores within which we have a high degree of confidence (95% Confidence interval (CI)) that the genuine average score would be found if the same survey were repeated an unlimited number of times with the same sample of participants. The vertical "staples" on the graph symbolize the 95% CI. The presence or absence of the 50 mark inside or outside of this range affects our findings about the relevance of the SUMI score for that specific component.

If the 50 mark is outside of the CI, the SUMI score for that aspect is considerably different from the reference norm. If, on the other hand, the 50 mark is within the CI, greater caution is required in making conclusions since it signals that the difference from the reference standard may not be statistically significant.

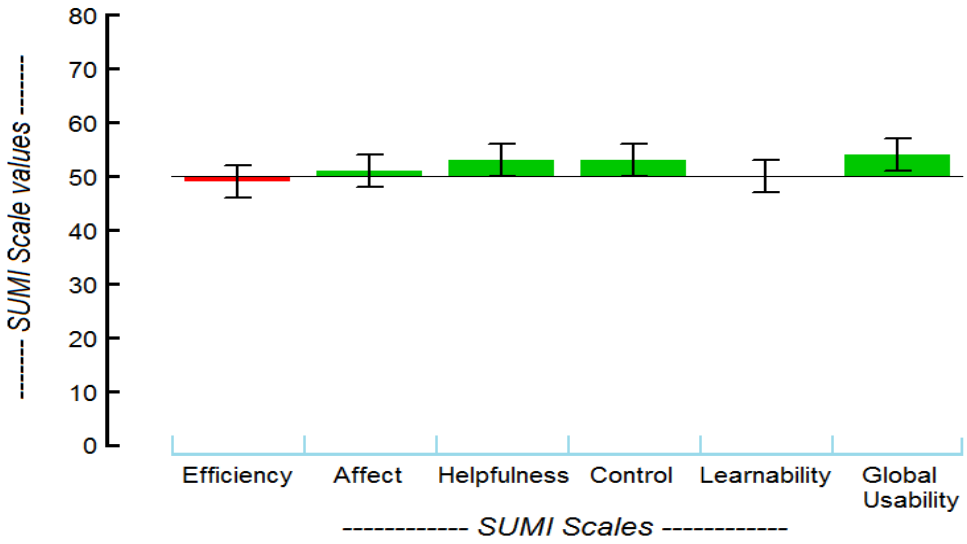


Figure 5.1: SUMI Scale Profiles: Means with 95% CI's

The Mean and standard deviations for various subscales are visually shown in Figure 5.2. The mean (the average value of a data group) and standard deviation (the dispersion of these values around the mean) are shown. The graph depicts the mean and deviation data for the SUMI and Global Usability scales in the sample under consideration. The circle mark indicates the mean, and the bars extending from it represent one standard deviation in each direction from the mean.

The mean scores for the bulk of the sub-scales are above average and fall within a suitable threshold (40–55). This shows that the EMS effectively meets its primary functional objectives and validates H6. Although there is room for improvement in the EMS's learnability and efficiency to maximize user satisfaction.

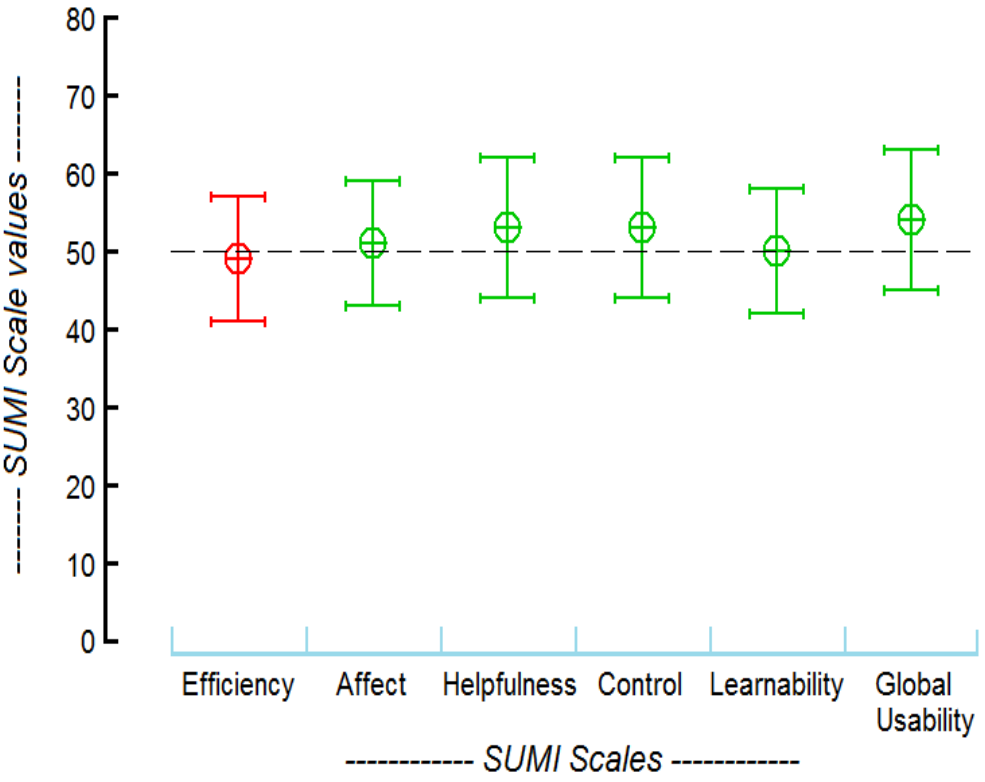


Figure 5.2: SUMI Scale Profiles: Means with Standard Deviation

The median is the center value in the datasets. The numerical number divides the bottom half of the sample from the upper half, with 50% of the observations falling below and 50% falling above it. The quartiles, particularly the first (25th percentile) and third (75th percentile), give extra information on the sample's spread around the median. These values are shown on a box plot (see Figure 5.3), with the median indicated by a horizontal line through the box and the quartiles defined by the box's borders.

The interquartile range (IQR) in Figure 5.3 is the distance between the first and third quartiles and the whiskers on each end of the box. This displays that 95% of all the data in the sample is projected to decrease.

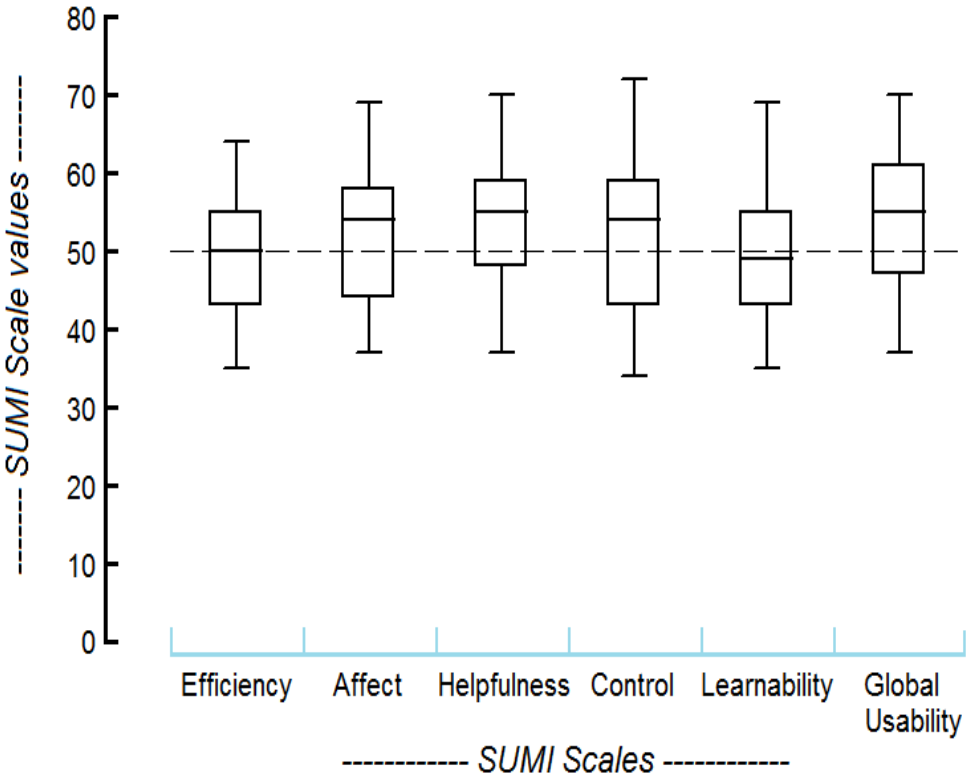


Figure 5.3: SUMI Scale Profiles: Median Boxplot

Chapter 6

CONCLUSION AND RECOMMENDATION

6.1 Conclusion

According to Lawal and Ogbu (2021), for more than 50 years, software has been a component of modern civilization. Today, various software development approaches are in use. Some businesses have their specific technique for producing software; however, the majority refer to agile software development (ASD) and the traditional approach. ASD is a process that has revolutionized the software industry, and it has spread to manufacturing and other business sectors in recent years. Unlike the traditional approach, it emphasizes iterative and incremental progress while stressing cooperation, adaptability, and customer satisfaction. The conventional practice focuses on comprehensive planning, detailed documentation, and expansive design (Lawal & Ogbu 2021).

This study's fundamental goal was to investigate how agile methodology innovation improves usability, efficiency, and success in the developed software. Agile Scrum methodology was adopted for the development of the EMS after a review study of various agile software development approaches. The critical objectives stated to attain the primary goal of this study are as follows:

1. We are researching the influence of innovation on agile software development.
2. To comprehend the adoption and application of the agile scrum software development approach in system design.

3. Identifies the system's perceived attributes (learning, efficiency, affect, Controllability, and helpfulness).

The first section of the research presented a summary of the study's goals, objectives, and contributions. The literature review addressed existing studies on the general challenges of software development projects and the historical transition from traditional to agile software development methodologies, including their lifecycles, responsibilities, benefits, and drawbacks.

The study looked at a group of employees from a Nigerian SME who were familiar with agile scrum and an EMS. The findings validated four of the six assumptions for the study, demonstrating that adopting agile scrum software development (ASSD) methodology to create the EMS improves user satisfaction and experience in an appropriate context. The agile scrum technique incorporates quality into the development of a system (Moyo, 2020).

The findings of this study validated H_2, H_3, H_4, and H_6 by establishing a favorable and substantial association between the EMS built utilizing the innovative, agile scrum software development (ASSD) technique and the system's perceived quality and usability. The findings showed that the EMS meets global usability criteria by improving Controllability, affect, and helpfulness. However, the results also rejected H_1 and H_5.

H_1 sought to investigate the relationship between the EMS's usability and its effectiveness in resolving management concerns. The results did not support the hypothesis, with users reporting that the EMS needs improvement in this area, even

though it provided valuable feedback and successfully addressed management issues. SUMI's statistical analysis reveals that the Efficiency sub-scale obtained the lowest score of 48.83, which is slightly lower than the average SUMI score of 50, implying that, while improvements are needed, the EMS was somewhat efficient. This is consistent with the findings of (Hinderks, Mayo, Thomaschewski, & Escalona 2022), who found that efficiency enhancements stimulate software usability.

H_2 investigated the users' perceptions of the EMS's help. The results of a statistical examination of the data show that the Helpfulness sub-scale received the highest score of 52.93, indicating the EMS's relative helpfulness in carrying out assigned activities.

H_3 investigates how the EMS affects the users from the users' point of view. The result of the EMS was a score of 51.47, above the usual benchmark of 50, which signifies that the affect sub-scale is high.

H_4 investigated how a user's sense of control over the software influences their impression of the EMS's responsiveness and consistency in reacting to their inputs and orders. The controllability score of the EMS was 52.70, suggesting an above-average amount of perceived user control over the model. Alghamdi et al. (2022). found that providing explicit instructions, controlling operations, and using commands to conduct activities improved software usability.

H_5 investigates how the EMS's learnability affects how readily individuals use it. This hypothesis is not supported by SUMI's result of 49.63, which is lower than the average score of 50. This shows that the system is difficult to use, indicating a need for simplification and that users may still be at the beginning of the system's learning

curve. According to Tortorella et al. (2022), the ease of use and learning rate rise as users become more familiar with the system. As a result, when the learning rate improves, so will efficiency and learnability.

Finally, the study used SUMI to assess the usability of the EMS produced utilizing the agile scrum software development approach. With a global usability score of 53.67, the research findings suggest that user satisfaction with the tool is above average. As a result, H_6 is validated by displaying overall user satisfaction with usability and implying that the EMS effectively achieves its primary functional criteria.

The global usability mean score of the EMS developed using ASSP is 53.67, low compared to the Website-Based Student Achievement Book created using the waterfall method by Budiarti, Fathin, and Sulistiyani (2022), with an overall mean score of 84. Still, there are differences in the number of respondents, system developed, and questionnaire variables. Hence, although further improvement needs to be done to the EMS and survey process due to cost and time limitation

6.2 Recommendations for Future Research

The data was acquired from employees of an SME in Nigeria through the administration of an online survey employing a purposive sample approach. Although several constraints were faced, such as the restricted scope of data collection in Nigeria, the study gives valuable insights into academic and practical issues, including theoretical implications and recommendations for future research. Based on the results of the study, the following are offered as recommendations for future endeavors:

- The scope of data gathering in future research should include a broader area in Nigeria's IT sector to acquire complete knowledge of EMS utilization across various enterprises.
- Further research should be done comparing Scrum and Waterfall by Designing a similar system.
- Further research should look towards methodologies like Scrumban to develop the EMS. Scrumban is a hybrid of Scrum and Kanban.

REFERENCE

Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2017). Agile software development methods: Review and analysis. *arXiv preprint arXiv:1709.08439*.

Adekoya, O. D., Adisa, T. A., & Aiyenitaju, O. (2022). Going forward: remote working in the post-COVID-19 era. *Employee Relations: The International Journal*, *44*(6), 1410-1427.

Afzal, M. W. (2023). *Sustainable Software Requirements* (Master's thesis, Itä-Suomen yliopisto).

Alghamdi, A. M., Riasat, H., Iqbal, M. W., Ashraf, M. U., Alshahrani, A., & Alshamrani, A. (2022). Intelligence and Usability Empowerment of Smartphone Adaptive Features. *Applied Sciences*, *12*(23), 12245.

Almeida, F., & Espinheira, E. (2021). Large-scale agile frameworks: a comparative review. *Journal of Applied Sciences, Management and Engineering Technology*, *2*(1), 16-29.

Alqudah, M., & Razali, R. (2018). An empirical study of Scrumban formation based on the selection of scrum and Kanban practices. *Int. J. Adv. Sci. Eng. Inf. Technol*, *8*(6), 2315-2322.

Al-Saqqa, S., Sawalha, S., & AbdelNabi, H. (2020). Agile Software Development: Methodologies and Trends. *International Journal of Interactive Mobile Technologies*, *14*(11).

Amin, M., Romney, G. W., Dey, P., & Sinha, B. (2019). Teaching relational database normalization in an innovative way. *Journal of Computing Sciences in Colleges*, *35*(2), 48-56.

Anwer, F., Aftab, S., Waheed, U., & Muhammad, S. S. (2017). Agile software development models tdd, fdd, dsdm, and crystal methods: A survey. *International journal of multidisciplinary sciences and engineering*, *8*(2), 1-10.

Bica, D. A. B., & da Silva, C. A. G. (2020). Learning process of agile scrum methodology with lego blocks in interactive academic games: Viewpoint of students. *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje*, *15*(2), 95-104.

Brand, M., Tiberius, V., Bican, P. M., & Brem, A. (2021). Agility as an innovation driver: towards an agile front end of innovation framework. *Review of Managerial Science*, *15*(1), 157-187.

Budiarti, R. P. N., FATHIN, A. N., & Sulistiyani, E. (2022). Website-Based Student Achievement Book Using the Waterfall Method. *IJRSM: International Journal of Scientific Research and Management*, *10*(3), 797-808.

Chigbu, U. E. (2019). Visually hypothesising in scientific paper writing: Confirming and refuting qualitative research hypotheses using diagrams. *Publications*, *7*(1), 22.

Cruz, A., & Alves, A. C. (2020). Traditional, agile and lean project management-A systematic literature review. *The Journal of Modern Project Management*, *8*(2).

Darmawan, A. K., Setyawan, M. B., Cobantoro, A. F., Masykur, F., Anwari, A., & Yulianto, T. (2021, August). Knowledge Management System Analysis of Smart Regency Mobile-Apps Service with Software Usability Measurement Inventory (SUMI) Approach. In *2021 International Conference on ICT for Smart Society (ICISS)* (pp. 1-6). IEEE.

Etikan, I., Musa, S. A., & Alkassim, R. S. (2016). Comparison of convenience sampling and purposive sampling. *American journal of theoretical and applied statistics*, *5*(1), 1-4.

Gheorghe, A. M., Gheorghe, I. D., & Iatan, I. L. (2020). Agile Software Development. *Informatica Economica*, *24*(2).

Ghezzi, A., & Cavallo, A. (2020). Agile business model innovation in digital entrepreneurship: Lean startup approaches. *Journal of business research*, *110*, 519-537.

Heriyanti, F., & Ishak, A. (2020, May). Design of logistics information system in the finished product warehouse with the waterfall method: review literature. In *IOP Conference Series: Materials Science and Engineering* (Vol. 801, No. 1, p. 012100). IOP Publishing.

Hinderks, A., Mayo, F. J. D., Thomaschewski, J., & Escalona, M. J. (2022). Approaches to manage the user experience process in Agile software development: A systematic literature review. *Information and Software Technology*, *150*, 106957

Iqbal, J., Ahmad, R. B., Khan, M., Alyahya, S., Nizam Nasir, M. H., Akhunzada, A., & Shoaib, M. (2020). Requirements engineering issues causing software development outsourcing failure. *PloS one*, *15*(4), e0229785.

Jeddi, F. R., Nabovati, E., Bigham, R., & Khajouei, R. (2020). Usability evaluation of a comprehensive national health information system: relationship of quality components to users' characteristics. *International journal of medical informatics*, *133*, 104026.

Kalenda, M. (2017). Scaling agile software development in large organizations. *Master's thesis in Faculty of Informatics, Masaryk University. Available from [accessed 3 Novemebr 2020]:*

Khairunisa, Y., Tyas, S. S., Purwanto, A., & Aisyah, S. (2020). Software Usability Measurement Inventory for Student Information Academic System at

Politeknik Negeri Media Kreatif. *IJISTECH (International Journal of Information System and Technology)*, *4*(1), 559-565.

Kirakowski, J., 1996. The software usability measurement inventory: background and usage. *Usability evaluation in industry*, pp.169-178.

Kola, B. (2014). Thinking Lean in Agile Software Development Projects. A qualitative study to Identify ways to improve productivity and increase business value.

Lavanya, R. (2020). Green Scrum Model: Implementation of Scrum in Green and Sustainable Software Engineering. *International Research Journal of Engineering and Technology (IRJET)*, *7*(11).

Lawal, A., & Ogbu, R. C. (2021). A comparative analysis of agile and waterfall software development methodologies. *Bakolori Journal of General Studies*, *11*(2), 1-2.

Lei, H., Ganjeizadeh, F., Jayachandran, P. K., & Ozcan, P. (2017). A statistical analysis of the effects of Scrum and Kanban on software development projects. *Robotics and Computer-Integrated Manufacturing*, *43*, 59-67.

Marek, K., Wińska, E., & Dąbrowski, W. (2021, January). The state of agile software development teams during the Covid-19 pandemic. In *International Conference on Lean and Agile Software Development* (pp. 24-39). Cham: Springer International Publishing.

Moyo, S. (2020). *A software development methodology for solo software developers: leveraging the product quality of independent developers* (Doctoral dissertation).

Richardson, B., Campbell-Yeo, M., & Smit, M. (2021). Mobile application user experience checklist: a tool to assess attention to core UX principles. *International Journal of Human–Computer Interaction*, *37*(13), 1283-1290.

Rodríguez, P., Mäntylä, M., Oivo, M., Lwakatare, L. E., Seppänen, P., & Kuvaja, P. (2019). Advances in using agile and lean processes for software development. In *Advances in Computers* (Vol. 113, pp. 135-224). Elsevier.

Rolik, O., Amons, O., Ulianytska, K., & Kolesnik, V. (2021). Modernization of the Second Normal Form and Boyce-Codd Normal Form for Relational Theory. In *Advances in Computer Science for Engineering and Education III 3* (pp. 296-305). Springer International Publishing.

Schwaber, K., & Sutherland, J. (2011). The scrum guide. *Scrum Alliance*, *21*(19), 1.

Shrivastava, A., Jaggi, I., Katoch, N., Gupta, D., & Gupta, S. (2021, July). A systematic review on extreme programming. In *Journal of Physics: Conference Series* (Vol. 1969, No. 1, p. 012046). IOP Publishing.

Stober, D., Suškevičs, M., Eiter, S., Müller, S., Martinát, S., & Buchecker, M. (2021). What is the quality of participatory renewable energy planning in Europe? A

comparative analysis of innovative practices in 25 projects. *Energy Research & Social Science*, *71*, 101804.

Tam, C., da Costa Moura, E. J., Oliveira, T., & Varajão, J. (2020). The factors influencing the success of on-going agile software development projects. *International Journal of Project Management*, *38*(3), 165-176.

Tekinerdogan, B., Ali, N., Grundy, J., Mistrik, I., & Soley, R. (2016). Quality concerns in large-scale and complex software-intensive systems. In *Software Quality Assurance* (pp. 1-17). Morgan Kaufmann.

Thummadi, B. V., & Lyytinen, K. (2020). How much method-in-use matters? A case study of agile and waterfall software projects and their design routine variation. *Journal of the Association for Information Systems*, *21*(4), 7.

Tortorella, G. L., Fogliatto, F. S., Anzanello, M. J., Vassolo, R., Antony, J., Otto, K., & Kagioglou, M. (2022). Learning curve applications in Industry 4.0: a scoping review. *Production Planning & Control*, 1-13.

Van der Linde, P. L., Wessels, C. H., & Kirakowski, T. (2013). A comparative study of three ICT network programs using usability testing. *Interim: Interdisciplinary Journal*, *12*(3), 174-187.

Van Vliet, H., Van Vliet, H., & Van Vliet, J. C. (2008). *Software engineering: principles and practice* (Vol. 13). Hoboken, NJ: John Wiley & Sons.

Zaini, H., Ishak, N. H., Johari, N. F. M., Rashid, N. A. M., & Hamzah, H. (2021, November). Evaluation of a Child Immunization Schedule Application using the Software Usability Measurement Inventory (SUMI) Model. In *2021 IEEE 11th International Conference on System Engineering and Technology (ICSET)* (pp. 281-285). IEEE.

APPENDICES

Appendix A: Survey Utilized for the Study

Software Usability Measurement Inventory

SUMI

NB The information you provide is kept completely confidential and no information is stored on computer media that could identify you as a person.

This questionnaire has 50 statements. Please answer them all. After each statement there are three boxes.

- Check the first box if you generally AGREE with the statement.
- Check the middle box if you are UNDECIDED, or if the statement has no relevance to your software or to your situation.
- Check the right box if you generally DISAGREE with the statement.

In checking the left or right box you are not necessarily indicating strong agreement or disagreement but just your general feeling most of the time.

There are also some general questions at the end.

Password:

What, in general, do you use this software for?

Statements 1 - 10 of 50.	Agree	Undecided	Disagree
This software responds too slowly to inputs.	○	○	○
I would recommend this software to my colleagues.	○	○	○
The instructions and prompts are helpful.	○	○	○
This software has at some time stopped unexpectedly.	○	○	○
Learning to operate this software initially is full of problems.	○	○	○
I sometimes don't know what to do next with this software.	○	○	○
I enjoy the time I spend using this software.	○	○	○
I find that the help information given by this software is not very useful.	○	○	○
If this software stops it is not easy to restart it.	○	○	○
It takes too long to learn the software functions.	○	○	○

Statements 11 - 20 of 50.	Agree	Undecided	Disagree
I sometimes wonder if I am using the right function.	○	○	○
Working with this software is satisfying.	○	○	○
The way that system information is presented is clear and understandable.	○	○	○
I feel safer if I use only a few familiar functions.	○	○	○
The software documentation is very informative.	○	○	○
This software seems to disrupt the way I normally like to arrange my work.	○	○	○
Working with this software is mentally stimulating.	○	○	○
There is never enough information on the screen when it's needed.	○	○	○
I feel in command of this software when I am using it.	○	○	○
I prefer to stick to the functions that I know best.	○	○	○

Statements 21 - 30 of 50.	Agree	Undecided	Disagree
I think this software is inconsistent.	○	○	○
I would not like to use this software every day.	○	○	○
I can understand and act on the information provided by this software.	○	○	○
This software is awkward when I want to do something which is not standard.	○	○	○
There is too much to read before you can use the software.	○	○	○
Tasks can be performed in a straight forward manner using this software.	○	○	○
Using this software is frustrating.	○	○	○
The software has helped me overcome any problems I have had in using it.	○	○	○
The speed of this software is fast enough.	○	○	○
I keep having to go back to look at the guides.	○	○	○

Statements 31 - 40 of 50.	Agree	Undecided	Disagree
It is obvious that user needs have been fully taken into consideration.	○	○	○
There have been times in using this software when I have felt quite tense.	○	○	○
The organisation of the menus seems quite logical.	○	○	○
The software allows the user to be economic of keystrokes.	○	○	○
Learning how to use new functions is difficult.	○	○	○
There are too many steps required to get something to work.	○	○	○
I think this software has sometimes given me a headache.	○	○	○
Error messages are not adequate.	○	○	○
It is easy to make the software do exactly what you want.	○	○	○
I will never learn to use all that is offered in this software.	○	○	○

Statements 41 - 50 of 50.	Agree	Undecided	Disagree
The software hasn't always done what I was expecting.	○	○	○
The software presents itself in a very attractive way.	○	○	○
Either the amount or quality of the help information varies across the system.	○	○	○
It is relatively easy to move from one part of a task to another.	○	○	○
It is easy to forget how to do things with this software.	○	○	○
This software occasionally behaves in a way which can't be understood.	○	○	○
This software is really very awkward.	○	○	○
It is easy to see at a glance what the options are at each stage.	○	○	○
Getting data files in and out of the system is not easy.	○	○	○
I have to look for assistance most times when I use this software.	○	○	○

How important for you is the kind of software you have just been rating?

- ○ Extremely important
- ○ Important
- ○ Not very important
- ○ Not important at all

How would you rate your software skills and knowledge?

- ○ Very experienced and technical
- ○ I'm experienced but not technical
- ○ I can cope with most software
- ○ I find most software difficult to use

What do you think is the best aspect of this software, and why?

What do you think needs most improvement, and why?

When you've answered all the questions:
please click the 'Send' button.

Send

4.2 SUMI EN 4.0

Appendix B: Request for the usage of SUMI

4/6/2023

To whom it may concern,

Samuel Oladapo Onamade is a student in Information Technology Department at the School of Computing and Technology, Eastern Mediterranean University and I guarantee that his usage of SUMI is solely for research purposes, and forms part of his research leading to a MSc in Information Technology. I guarantee that the student is not working in a consultancy relationship with any commercial interest as far as their use of SUMI is concerned, and that I will help the student to take reasonable steps to protect the intellectual property rights and copyright of SUMI.

Akile Oday *Asst. Prof. Dr.* in Information Technology Department (I.T)

Appendix C: Authorization to Utilize SUMI

SUMI record report

From: sumi_admin@uxp.ie

To: samuelonamade@yahoo.com

Date: Thursday, April 13, 2023 at 01:24 PM GMT+3

Please find information about your SUMI setup.

Do not reply to this email.

Password: fz674
Client: Samuel Onamade
E-mail: samuelonamade@yahoo.com
Originating page: sumi.uxp.ie/en/
Creation date: 2023-04-13:11:20:27
Software evaluated: Roflat

Ask respondents to go to the originating page, and put in the password as given at the top, so that their responses will be validated. Passwords are NOT case sensitive.

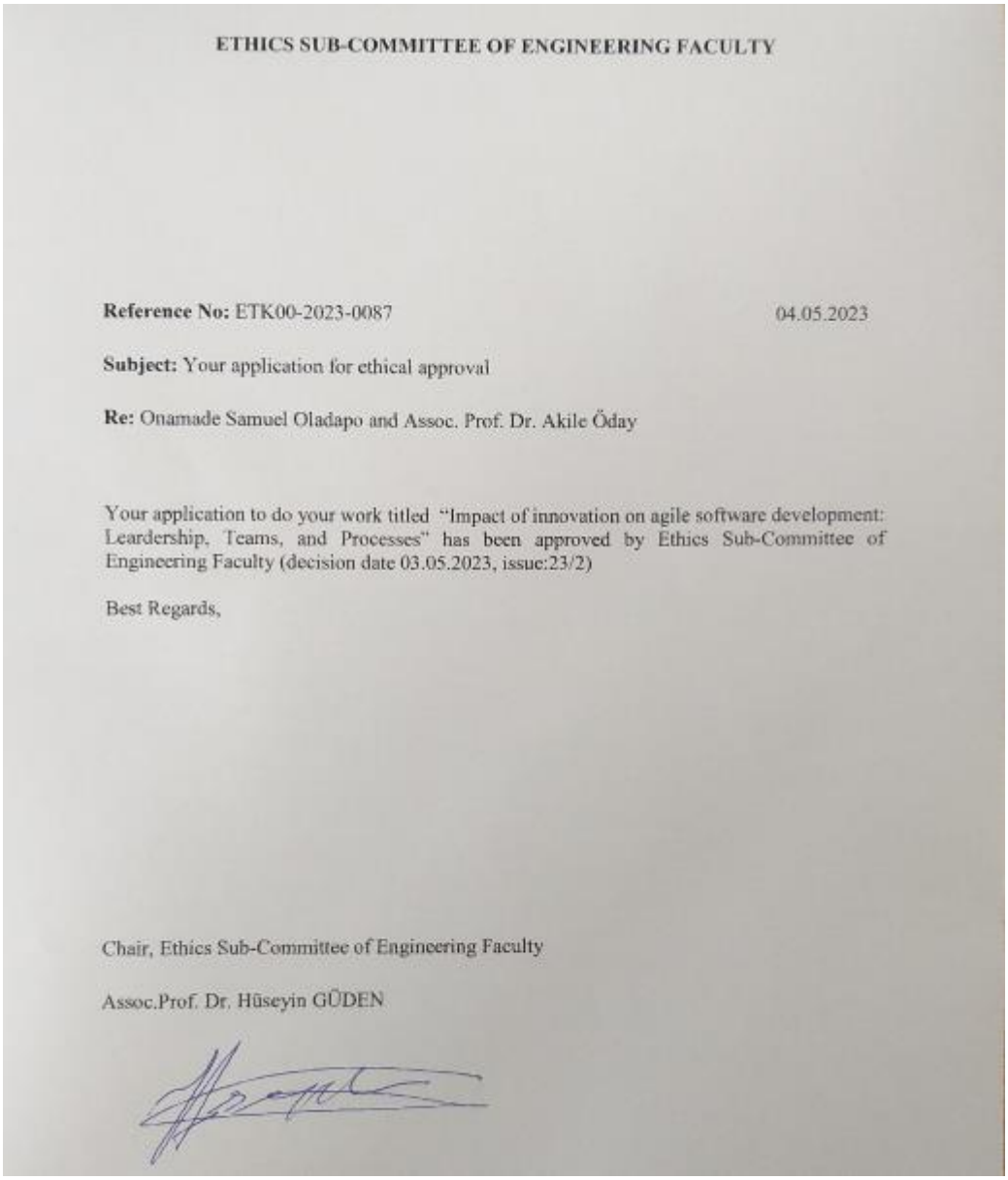
When you have sufficient data, please contact jzk@uxp.ie to generate the professional report.

If you want to test the setup, please use 999 as a password and put your name in one of the plaintext fields.

If you need to know at any time how much data you have gathered, contact jzk@uxp.ie for a progress report.

Best wishes for your study.

Appendix D: Ethics Committee Approval



ETHICS SUB-COMMITTEE OF ENGINEERING FACULTY

Reference No: ETK00-2023-0087 04.05.2023

Subject: Your application for ethical approval

Re: Onamade Samuel Oladapo and Assoc. Prof. Dr. Akile Öday

Your application to do your work titled "Impact of innovation on agile software development: Leardership, Teams, and Processes" has been approved by Ethics Sub-Committee of Engineering Faculty (decision date 03.05.2023, issue:23/2)

Best Regards,

Chair, Ethics Sub-Committee of Engineering Faculty

Assoc.Prof. Dr. Hüseyin GÜDEN

Appendix E: Participants Consent Form

Samuel Oladapo Onamade

MSc Information Technology

05338493359

21506213@emu.edu.tr

CONSENT FORM

Dear Participant,

Thank you for agreeing to participate in this survey. My name is Onamade Samuel Oladapo, an MSc student at the Eastern Mediterranean University, North Cyprus. I am carrying out a survey for my MSc thesis. The survey is about "IMPACT OF INNOVATION ON AGILE SOFTWARE DEVELOPMENT: LEADERSHIP, TEAMS, AND PROCESSES". It should take about 10 minutes to complete the survey.

Your participation in this survey is completely voluntary and you may stop the survey at any time. Neither your name nor any other identifying information will be recorded on the survey, and your responses will be kept completely anonymous. There is no known risk involved in this. Prospective data related to your participation will be used for research purposes only.

If you have any questions, please feel free to contact me (21506213@emu.edu.tr or 05338493359) or my advisor, Assoc. Prof. Akile Öday (akile.oday@emu.edu.tr or 630-1183)

Onamade Samuel Oladapo

M.Sc. Candidate

Department of Information Technology

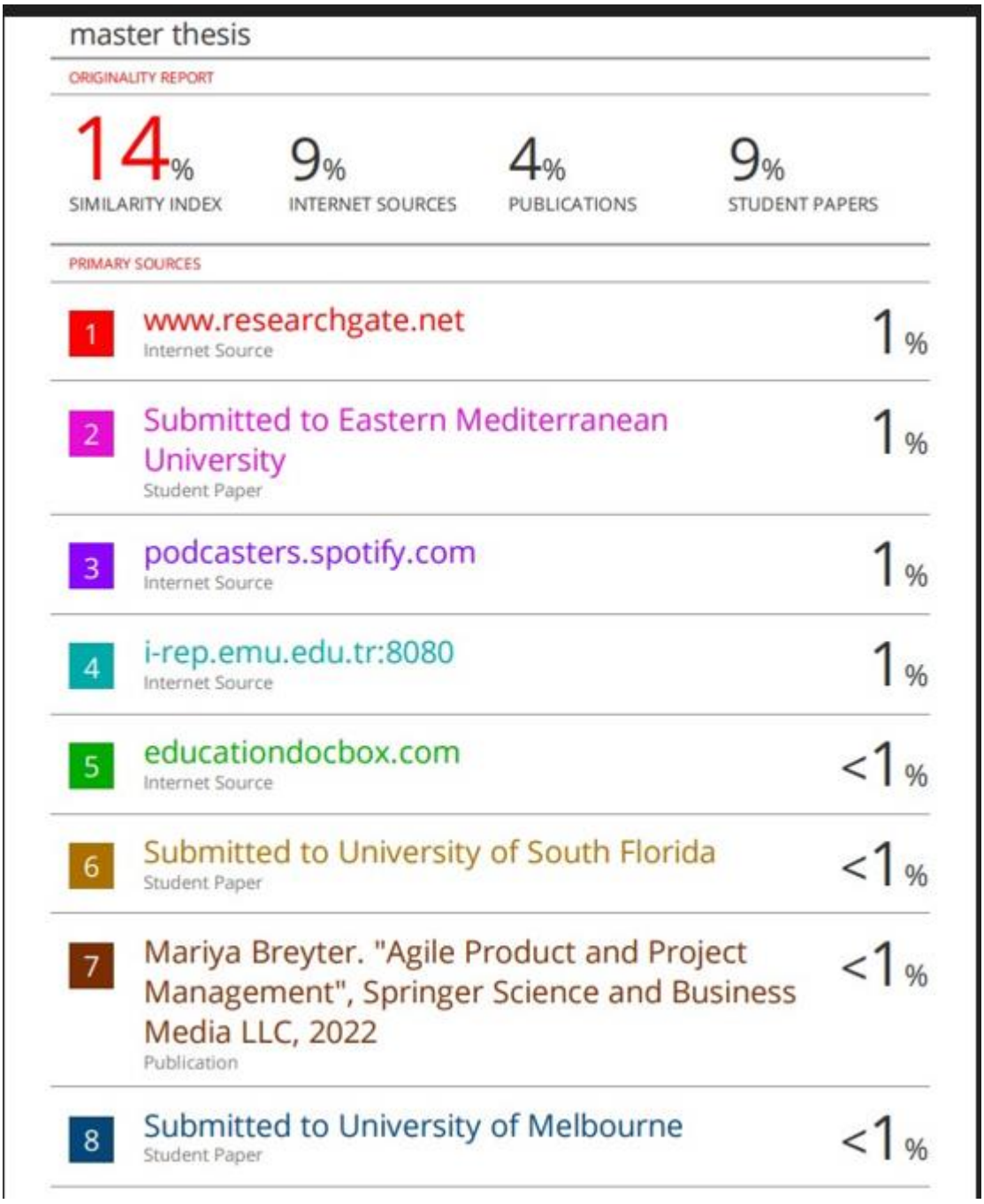
I have read and I understand the provided information and have had the opportunity to ask questions. I understand that my participation is voluntary and that I am free to withdraw at any time, without giving a reason. I understand that I will be given a copy of this consent form. I voluntarily agree to take part in this study.

13/04/2023

Date

Researcher Signature

Appendix F: Turnitin Uniqueness Result



master thesis

ORIGINALITY REPORT

14%	9%	4%	9%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.researchgate.net Internet Source	1%
2	Submitted to Eastern Mediterranean University Student Paper	1%
3	podcasters.spotify.com Internet Source	1%
4	i-rep.emu.edu.tr:8080 Internet Source	1%
5	educationdocbox.com Internet Source	<1%
6	Submitted to University of South Florida Student Paper	<1%
7	Mariya Breyter. "Agile Product and Project Management", Springer Science and Business Media LLC, 2022 Publication	<1%
8	Submitted to University of Melbourne Student Paper	<1%